UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

UNIVERSITÀ DEGLI STUDI DI TRIESTE

XXXVIII CICLO DEL DOTTORATO DI RICERCA IN APPLIED DATA SCIENCE AND ARTIFICIAL INTELLIGENCE

REPRESENTATION LEARNING FOR ROBUST AND ADAPTABLE ARTIFICIAL INTELLIGENCE

Settore scientifico-disciplinare: **INF-01**

DOTTORANDO
EMANUELE BALLARIN

COORDINATORE
PROF. FRANCESCO PAULI

SUPERVISORE DI TESI
PROF. LUCA BORTOLUSSI

CO-SUPERVISORE DI TESI
PROF. FABIO ANSELMI

ANNO ACCADEMICO 2024/2025

Università degli Studi di Trieste

Ph.D. in Applied Data Science & Artificial Intelligence

XXXVIII cycle

Representation learning for robust and adaptable artificial intelligence

Candidate
Emanuele Ballarin

Supervisors
Prof. Luca Bortolussi
Prof. Fabio Anselmi

Ingegno c'era nell'allenare congegni

ITALIAN PALINDROME

Abstract

The most remarkable trait of *deep learning* is the ability of deep artificial neural networks to solve any problem they are trained upon according to their own, learnt and bespoke, *internal representation* of it. This allows such models not to depend anymore on hand-crafted features specified beforehand, and allows modellers to reach for solutions they are not even able to foresee. Such advantage, however, does not come without a price as a lack of control on the inner workings of the model may hinder generalisation or give rise to bizarre idiosyncrasies. Additionally, these models are still far from having attained an *intuitive* understanding of, and swift adaptability to, new unforeseen problem variations — a quintessential trait of biological intelligence.

In this thesis, we will use the lens of *representation learning* to explore several failure modes of deep learning models — sharing a common lack of robustness, flexibility, and adaptation to the unexpected. We will also contextually develop semi-empirical methods to steer model training towards safer, more compliant, predictable, and adaptable behaviour — by making the role of the internal representation more explicit within the objective being optimised, or by performing specific interventions on it. Applications will include the development of models capable to withstand unforeseen adversarial perturbation of the inputs, to adapt in the multi-task learning of geometrically challenging tasks, and to optimise specific simulated quantum systems within a broad range of configurations.

We begin by considering the problem of *adversarial vulnerability*, *i.e.* the susceptibility of trained deep learning models to behave unexpectedly towards failure in response to legitimate inputs subject to a slight malicious perturbation. The main mitigations in such respect require either extending the training set with incrementally corrupted samples (adversarial training), or trying to revert the result of the perturbation before feeding inputs to the model (input purification[1]) — both with their own merits and neither definitive. We propose to merge the two approaches, in the context of image classification, by having a generative model perform input purification before feeding the result to an adversarially-trained classifier, conditionally on the internal representation of the very same classifier. Since such arrangement is able to produce several different input reconstructions, a custom aggregation function of the predicted classes is used to ensemble them robustly. Our method, evaluated by standard robustness benchmarks, is able to surpass the state-of-the-art for both either method considered alone, and the trivial merge of the two.

We then investigate a significantly different setting — an application from physics —

[1] Also known as input *sanitisation*.

that still involves a system lacking adaptability to variations. In particular, we focus on the propagation of radiation-induced energy-carrying *excitations* along a *quantum network*. The setup is not arbitrary, as it can be used to model, *e.g.*, the production of an electric current from sunlight in photovoltaic systems or biological compounds that enable photosynthesis-like processes. In such case, intrinsic properties of the system make it maximally responsive only to a very specific frequency of the radiation incident to it — inducing a significant loss of efficiency in the case of artificial systems of this kind. We first develop a fully-differentiable computational simulation of these systems, then identify few *local* components whose optimisation could enhance the efficiency of the system *w.r.t.* nonresonant radiation, and finally optimise those by gradient-based methods and algorithmic differentiation. Our key finding is that the introduction of driving terms at the *antenna/network* and *network/sink* sites — whose harmonic composition is to be learned online — allows for very fast and almost-optimal adaptation to incident radiation of arbitrary frequency, bears a net positive energy balance, and scales well as network size increases.

Along the same trajectory, and within the context of supervised *multi-task* learning, we investigate the interplay between representation structure and amenability to *very-few-shot* learning of new tasks. Indeed, the training of deep learning models requires copious amounts of data, compute, and statistical *trial-and-error*. However, most often the results of such process are oddly task-specific, brittle to slight variations in data distribution, and scarcely transferable to new related but unforeseen objectives. While approaches relying on transfer learning, continual learning, or fine-tuning do partially address such limitation, they still require a significant amount of data and ad-hoc precautions to prevent the perils of forgetting or overfitting. Identifying a crucial issue in the intermingling of structure learning and task resolution, we propose a framework for multi-task representation learning based on a non-linear generalisation of *Koopman operator learning*. After a standard training phase, the resulting model allows for test-time incremental adaptation to different similarly-grounded objectives — via the fitting of a minimal additional set of parameters which act linearly on their input. Such adaptation is fast, relies on closed-form parameter updates, and requires only a handful of iterations and the theoretical minimum of examples from the new task.

Finally, we study the emergence of sparse ensembles in the latent representation of small neural networks trained with the *Forward-Forward* algorithm — a recently introduced and purportedly more biologically-plausible optimisation scheme alternative to *Backpropagation*. In such case, indeed, we observe the formation of highly sparse representations organised in small neuron pools that selectively activate for specific input categories — a landmark feature of activation patterns in sensory cortices. However, we also show that such patterns can similarly emerge in models trained by Backpropagation, just driven by the same objective as Forward-Forward learning. This suggests that the loss function may play a role as crucial as the learning algorithm in the quest for biological plausibility of learning machines.

Contents

Abstract iii

Contents v

Scientific basis 1

1 Introduction 1
- 1.1 Contributions . . . 2
  - 1.1.1 Adversarial vulnerability & robustness (ch. 3) . . . 2
  - 1.1.2 Optimisation of complex simulated physical systems (ch. 4) . . . 2
  - 1.1.3 *Very-few-shot* multi-task learning (ch. 5) . . . 3
  - 1.1.4 Representations from *Forward-Forward* learning (ch. 6) . . . 4

2 Preliminaries 5
- 2.1 General preliminaries . . . 5
- 2.2 Preliminaries on *adversarial attacks* and *defences* (ch. 3) . . . 6
  - 2.2.1 Related works . . . 6
  - 2.2.2 Technical aspects . . . 7
- 2.3 Preliminaries on *continuous-time quantum walks* (ch. 4) . . . 9
  - 2.3.1 Technical aspects . . . 9
- 2.4 Preliminaries on *Koopman operator learning* (ch. 5) . . . 10
  - 2.4.1 Technical aspects . . . 10
- 2.5 Preliminaries on *Forward-Forward* and *neuronal ensembles* (ch. 6) . . . 11
  - 2.5.1 Related works . . . 11

3 Blending *adversarial training* and *representation-conditional purification* via *aggregation* improves adversarial robustness 15
- 3.1 Introduction . . . 15
- 3.2 Structure of CARSO . . . 17
  - 3.2.1 Architectural overview and principle of operation . . . 17
  - 3.2.2 A first-principles justification . . . 19
  - 3.2.3 Hierarchical input and internal representation encoding . . . 19
  - 3.2.4 Adversarially-balanced batches . . . 20
  - 3.2.5 Robust aggregation strategy . . . 20
- 3.3 Experimental assessment . . . 22
  - 3.3.1 Setup . . . 22

3.3.2 Results and discussion . . . 24
3.3.3 Experimental justification of the *robust aggregation strategy* . . . 26
3.3.4 Limitations and open problems . . . 27
3.4 Conclusion . . . 28

4 Driving enhanced exciton transfer by automatic differentiation 29
4.1 Introduction . . . 29
4.2 General settings and methods . . . 30
4.3 Analysis and results . . . 33
4.3.1 Nearest neighbour network . . . 33
4.3.2 Star Network . . . 35
4.3.3 FMO . . . 36
4.4 Discussion . . . 38

5 Towards very-few-shot structured supervised multi-task learning 41
5.1 Introduction . . . 41
5.2 The framework . . . 42
5.2.1 General setting . . . 42
5.2.2 Model structure . . . 43
5.2.3 Model training . . . 44
5.2.4 Test-time adaptation . . . 46
5.3 Preliminary experimental assessment . . . 46
5.3.1 Setup . . . 47
5.3.2 Results . . . 48
5.4 Conclusion . . . 51
5.4.1 Future developments . . . 52

6 Emergent representations in networks trained with the Forward-Forward algorithm 53
6.1 Introduction . . . 53
6.2 Methods . . . 54
6.2.1 Data . . . 56
6.2.2 Model trained with Forward-Forward (FF) . . . 56
6.2.3 Model trained with Backpropagation on the goodness objective (BP/FF) . . . 56
6.2.4 Model trained with Backpropagation on the cross-entropy loss (BP) 57
6.2.5 Analysis of representations . . . 57
6.3 Results . . . 58
6.3.1 Classification accuracy . . . 58
6.3.2 Forward-Forward elicits sparse neuronal ensembles . . . 58
6.3.3 Visually similar classes can elicit ensembles with shared neurons 60
6.3.4 Representations of unseen categories can elicit well-defined ensembles . . . 61
6.3.5 Distribution of excitatory and inhibitory connections . . . 62
6.4 Discussion and conclusions . . . 64

7 Conclusion 67

Bibliography 69

A Supplementary Material for Chapter 3 83
A.1 Justification of Adversarially-balanced batches . . . 83
A.2 Architectural details and hyperparameters . . . 84
A.2.1 Architectures . . . 84
A.2.2 Hyperparameters . . . 86
A.3 Ablation study on the need for adversarial training . . . 88
A.4 Further results and comparisons . . . 93

B Supplementary Material for Chapter 4 95
B.1 Appendix 1: Additional case study . . . 95

C Supplementary Material for Chapter 5 97
C.1 On the limitations of invertible linear latent dynamics in the presence of absorbing states . . . 97
C.1.1 Setup and Notation . . . 97
C.1.2 Case I: Scalar Latent Space ($d_k = 1$) . . . 98
C.1.3 Case II: Low-Dimensional Latent Space ($d_k \ll n$) . . . 99

D Supplementary Material for Chapter 6 101
D.1 Data . . . 101
D.2 Training details . . . 101
D.3 Statistical test of ensemble assignment . . . 102
D.4 Activation patterns in deeper layers . . . 102
D.5 Activation patterns in different models . . . 104
D.6 Further results on representations of unseen categories and their ensembles 105
D.7 Performance on unseen categories with linear probes . . . 106
D.8 Enforcing sparsity in the BP model . . . 106
D.9 Representation similarity . . . 108
D.9.1 Model comparison . . . 108
D.9.2 Layer comparison . . . 108
D.10 Forward-Forward with ℓ_1 goodness function . . . 109

Scientific basis

Published journal papers that constitute the scientific basis for the contributions described in this thesis are listed below:

- [1] E. Ballarin, A. Ansuini and L. Bortolussi. 'Blending adversarial training and representation-conditional purification via aggregation improves adversarial robustness'. In: *Transactions on Machine Learning Research*. 2025
- [2] E. Ballarin, D. A. Chisholm, A. Smirne, M. Paternostro, F. Anselmi and S. Donadi. 'Driving Enhanced Exciton Transfer by Automatic Differentiation'. In: *Machine Learning: Science and Technology* (2025)
- [3] N. Tosato, L. Basile, E. Ballarin, G. de Alteriis, A. Cazzaniga and A. Ansuini. 'Emergent representations in networks trained with the Forward-Forward algorithm'. In: *Transactions on Machine Learning Research*. 2025

Furthermore, this thesis describes preliminary results from still unpublished work done at University College London as part of an academic research visit under the local supervision of Prof. Carlo Ciliberto. A more formal and complete exposition of those results and their theoretical underpinnings will soon be available.

Finally, an additional conference workshop paper from the author – whose results have not been included in this thesis due to topic inhomogeneity – is the following:

- [4] E. Ballarin, F. Giacomarra, A. Mecchina and N. A. Pearson. 'Demystifying Generative AI: A Pedagogical Approach'. In: *Ital-AI Workshop on Generative AI for Education*. 2025

Code for the reproduction of the experiments contained in published papers is also available. In particular:

- Code for [1] is available at:
  `github.com/emaballarin/CARSO`
- Code for [2] is available at:
  `github.com/emaballarin/excitorch`
- Code for [3] is available at:
  `github.com/NiccoloTosato/EmergentRepresentations`

Chapter 1

Introduction

Artificial Intelligence (AI) is deeply permeating our society with transformative effects at every scale [5, 6]. From technological innovation to social relationships, from corporate organisation to entertainment, there is hardly any aspect it does not affect of how we work, access knowledge, communicate, or take care of ourselves. In fact, systems based on AI methods — *Deep Learning* (DL) in particular — have attained remarkable (and sometimes super-human) success in tasks usually associated with higher cognitive functions: notably, pattern recognition, decision-making, and generation of original content.

The enabling factors for such paradigmatic shift can be traced back to the convergence of a growing abundance of easily-accessible training data, the availability of ever more efficient hardware accelerators, and advances in artificial neural network research. Among the latter, in particular, the development of *deep* architectures — able to jointly learn a solution to the training task, and the most suitable featurisation of training data to make it possible [7]. Once bound to a pre-defined choice of input features, statistical learning techniques could now be applied directly to raw data, allowing them to discover an effective hierarchical representation just by error minimisation.

Given the impressive results and the widespread adoption of such technology, one may legitimately ask *where's the rub* — and indeed there is. What stellar benchmark results cannot tell is the fact that each of those top-performing models has been typically trained from scratch, with a significant time and data budget, on that very task-specific task alone. This usually applies also to the case of significantly similar (but still different) tasks. And while transfer, continual, and fine-tuning learning techniques do exist and mitigate the issue, particular care should be exercised during adaptation to a new task in order to balance the effects of overfitting and forgetting (which can be catastrophic at the worst).

Even though commonplace in the statistical learning community, such *modus operandi* is strikingly different from how learning happens in humans — where *fluid* intelligence, adaptability, and synthetic reuse of previous knowledge is paramount to the cognitive prowess of humankind [8, 9].

In this thesis, we will explore, through the lens of *representation learning*, several aspects that embody the *crystallised* nature of deep learning models. Contextually, we will

develop or investigate novel semi-empirical approaches to mitigate such pitfalls, within specific areas of application.

The rest of the document is structured as follows. In the next section 1.1, a general but more analytical overview of our contributions is provided — concluding the *Introduction*. In chapter 2, related works and elements of foundational background are provided — to better appreciate the subsequent chapters — either in the form of a concise review, or by pointing to relevant published material. Chapters 3, 4, 5, and 6 will analyse at much greater depth each of the specific scenarios and applications considered, including the novel methods or observations we propose. Chapter 7 summarises our findings and tries to find a common ground to move forward, towards a more robust and adaptable artificial intelligence. Appendices A, B, C, and D provide supplementary material for our results.

1.1 Contributions

The following section provides a summary of the main technical contributions of this thesis, with emphasis on the aspects of novelty. A minimal overview of the setting will be provided for context; the reader is however invited to refer to chapter 2 or to individual chapters for a more comprehensive analysis. Paragraphs will be named according to the topic of the contribution and the associated chapter.

1.1.1 Adversarial vulnerability & robustness (ch. 3)

It has been shown that deep learning models are susceptible to adversarial perturbations, *i.e.* the addition of potentially small, imperceptible, malicious noise to otherwise legitimate input data being ultimately able to significantly and catastrophically alter model behaviour *w.r.t.* reasonable expectations [10–12]. Approaches to the mitigation of such pitfall broadly belong to either *adversarial training* or *adversarial purification*. The former technique consists in enriching the training dataset with perturbed inputs (eventually associated with their pre-perturbation output); the latter aims at removing the effect of the perturbation before such input is classified by the original model. As of today, only adversarial training has proven to be significantly robust to attempts of tampering [13], while still leaving ample margin for improvement.

In such regard, we speculate that the full representation of a model under attack would provide more useful information to a defender compared to the input alone. In such sense we develop a hybrid defence where such representation is directly used to recover sanitised inputs and classify them robustly.

We assume an ℓ_∞ *white-box* threat model and test our method in image classification tasks against *state-of-the-art* adversarial training and purification approaches, giving the attacker the greatest advantage possible. Still, with only a slight decrease in *clean* accuracy, out technique results to be the most effective in comparison to either.

1.1.2 Optimisation of complex simulated physical systems (ch. 4)

Complex systems, *i.e.* physical systems with many non-trivially interacting parts, pervade the description of our world — both natural and artificial — and many technological

innovations rely on the hard task of obtaining a sufficiently precise control over them. Among those there are *photovoltaic systems* [14, 15] — where incident electromagnetic radiation (including, *e.g.*, light) elicits the production of an electric current: this is the case of compounds that allow photosynthesis-like processes in living organisms, or that of materials enabling non-thermal energy harvesting from sunlight [16]. Most of those systems, however, are maximally sensitive only to a narrowband of the emissive spectrum, a crucial limitation to their efficient use.

We investigate the propagation of quantum excitations through several simplified photovoltaic system models, using the formalism of *Markovian continuous time quantum walks* (CTQWs) [17] — and simulate their time evolution in a fully-differentiable fashion thanks to the use of an algorithmic differentiation [18] software framework. We then identify elements of the models where some form of external control can be enforced in an attempt to make the system optimally responsive to arbitrary incident frequency, and finally optimise dependant control parameters along the simulation by gradient-based methods — akin to the training of a deep neural network.

Our most effective intervention — the addition of two driving oscillators whose amplitudes and frequencies are to be learned online — allows fast adaptability of the system to even strongly off-resonant frequencies, while maintaining a net positive energy balance and being robust to initial condition variations.

1.1.3 *Very-few-shot* multi-task learning (ch. 5)

Despite the remarkable success of deep learning in a large variety of problems commonly associated with human cognition, the typical approach to its use relies on establishing a task (or set thereof) and training a model for it — should requirements mutate, training would need to be repeated. Such process is very often intensive in terms of time, computation, and training data needed, making it a challenging endeavour. Approaches to promote model adaptation to new tasks without model retraining do exist [19–23], but still require abundant data and *ad-hoc* precautions to balance overadaptation to new examples and forgetting of the old tasks [24]. This becomes especially evident when the optimal representations required to solve different tasks, considered alone, are incompatible or conflicting [25]. Furthermore — especially in the case of AI-powered devices and even on a single well-defined task — the deployment setup could be slightly different from training, requiring minor but crucial adaptations in order to ensure the expected behaviour.

As a novel approach to the problem, we develop a framework for multi-task representation learning geared towards discrete-time dynamical systems that extends *Koopman operator learning* [26–29]. In particular, each task — known during training, or unforeseen and thus requiring adaptation — induces a dynamics whose atomic transition is described by an affine operator. Such operators act on a task-agnostic embedding describing the system state, and are modulated by a task-agnostic nonlinear term that accounts for local variations of the dynamics — both to be learned jointly with tasks during training. Test-time adaptation to new tasks relies only on few fast closed-form fitting iterations, and requires the minimal number of examples.

Preliminary tests on challenging visual geometrically-grounded tasks show that the framework is able to learn a representation close to the theoretical optimum, while also allowing robust generalisation and swift adjustment to new tasks of the same kind. The same happens also when the state-space is partitioned across tasks, and variables unrelated to the task are introduced.

1.1.4 Representations from *Forward-Forward* learning (ch. 6)

Cortical areas of the brain are characterised by sparse activation patterns and the emergence of small pools of neurons that co-activate (only) in response to specific stimuli [30, 31]. Additionally, it has been shown that neurons or ensembles thereof are able to implement only a limited subset of mathematical or logical functions of their inputs. Such behaviour seems to be at odds with the particular implementation of modern deep learning architectures — implementing arbitrary nonlinearities and learning thanks to algorithmic differentiation and the *backpropagation* [32] algorithm [33]. For such reason, some alternatives to backpropagation have been proposed, in an attempt to make learning of NNs more biologically-plausible. Among those, the *Forward-Forward* algorithm [34], which relies on a contrastive-like [35] learning algorithm requiring only two forward passes (on *positive* and *negative* data) and local per-neuron activation norm maximisation/minimisation.

We investigate the structure of representations emerging in small neural networks tasked with image classification, under the drive of such learning procedure. In particular, we analyse their per-layer sparsity and excitatory/inhibitory balance, and look for the emergence of class-associated ensembles. Interestingly, we discover that all these properties [31, 36] — high sparsity, emerge of ensembles, and E/I balance — are aligned with the behaviour of representations found in sensory cortices.

We additionally try to isolate the source of such peculiar behaviour: whether it is caused by the learning algorithm, by the optimisation objective, or by the interaction of the two. Curiously, we find out that much of the observed sparsity can be obtained also by training an equivalent model by backpropagation — just on the same contrastive-like objective proposed. The choice of the norm type in such objective has also an influence on the result [37].

Chapter 2

Preliminaries

This thesis, as a whole, is predominantly concerned with the study of representations in deep artificial neural networks, and with the development of novel techniques to address some of their shortcomings. As such, it heavily relies on a large body of prior knowledge and technical contributions from the very same *machine learning* and *deep learning* communities it contributes back to. Additionally, the content of specific chapters inserts within further specialised areas of research, with their own idiosyncratic developments and conventions: it is the case, *e.g.*, of chapter 4, dedicated to an application overarching quantum information and the simulation of networked quantum systems.

Given the breadth of the subjects, a deep and detailed analysis of those preliminary topics would result being lengthy, desultory, and ultimately out of scope for this work. Nonetheless, references to essential bibliography and a succinct review of aspects deemed essential to describe our contributions will be contained in this chapter.

2.1 General preliminaries

For a general overview of the field of (classical) *machine learning*, its core methodology, and landmark techniques on which also the developments of *deep learning* build upon, the reader can refer to Bishop [38]. For a more modern and nuanced take, broadly linking the way of thinking typical of machine learning with probabilistic and Bayesian reasoning, and with optimal decision-making, Murphy [39] and Barber [40] provide a solid foundation. A renewed and expanded two-book version of the former is also available [41, 42].

As far as post-2012[1] deep learning is concerned, Goodfellow, Bengio and Courville [44] has now become the typical, although slightly outdated, reference on the topic. Prince [45] provides instead an updated, cutting-edge introduction. For a more technical, hands-on approach, the reader can refer to Zhang et al. [46].

Finally, for a deeper analysis of the phenomenological and mathematical foundations of deep learning, including formal and specific (somehow *niche*) aspects, Hertz, Palmer and

[1] The famous *AlexNet* paper [43] and convolutional neural network architecture are now conventionally considered to have started the *deep learning revolution* at large.

Krogh [47] provide an anticipatory seminal endeavour. A much more modern reference, focusing on a structured mathematical foundation of the field has been instead built by Petersen and Zech [48].

2.2 Preliminaries on *adversarial attacks* and *defences* (ch. 3)

2.2.1 Related works

Adversarial training as a defence

The idea of training a model on adversarially-generated examples as a way to make it more robust can be traced back to the very beginning of research on the vulnerabilities of deep learning. In their seminal work, Szegedy et al. [11] propose to perform training on a mixed collection of *clean* and adversarial data, generated beforehand.

The introduction of the *Fast Gradient Sign Method* (FGSM) [49] enables the efficient generation of adversarial examples along the training, with a single normalised gradient step. Its iterative generalisation called *Projected Gradient Descent* (PGD) [50] – discussed in subsection 2.2.2 – significantly improves the effectiveness of the adversarial examples produced, making it still the *de facto* standard for the synthesis of adversarial training inputs [51]. Further incremental improvements have also been developed, some focused specifically on robustness assessment (*e.g.* stepsize-adaptive variants, as by Croce and Hein [52]).

The most recent adversarial training protocols further rely on synthetic data to increase the numerosity of training datapoints [53–58], and adopt adjusted loss functions to balance robustness and accuracy [59] or generally foster the learning process [56]. The entire model architecture may also be tuned specifically for the sake of robustness enhancement [57]. At least some of such ingredients are often required to reach the current *state-of-the-art* in robust accuracy via adversarial training.

Purification as a defence

Among the first attempts of *purification-based* adversarial defence, Gu and Rigazio [60] investigate the use of denoising autoencoders [61] to recover examples free from adversarial perturbations computed against an image classifier. Despite its effectiveness in the denoising task, the method may indeed *increase* the vulnerability of the system when attacks are generated against it *end-to-end* [60], since the denoising process itself can fall victim of the attack. The proposed improvement adds a smoothness penalty to the reconstruction loss, partially mitigating such downside [60]. Similar in spirit, Liao et al. [62] tackle the issue by computing the reconstruction loss between the last-layers representations of the frozen-weights attacked classifier, respectively receiving, as input, the *clean* and the tentatively *denoised* example.

In the work by Samangouei, Kabkab and Chellappa [63], *Generative Adversarial Networks* (GANs) [64] learnt on *clean* data are used at inference time to find a plausible synthetic example – close to the perturbed input – belonging to the unperturbed data manifold.

Despite encouraging results, the delicate training process of GANs and the existence of known failure modes [65] limit the applicability of the method. More recently, a similar approach [66] employing *energy-based models* [67] failed to recover high-quality input reconstructions [68].

Purification approaches based on (conditional) variational autoencoders include the works by Hwang et al. [69] and Shi, Holtz and Mishne [70]. Very recently, a technique combining variational manifold learning with a test-time iterative purification procedure has also been proposed [71].

Finally, already-mentioned techniques relying on *score-* [72] and *diffusion-* based [68, 73] models have also been developed, with generally favourable results – often balanced in practice by longer training and inference times, and a much more fragile robustness assessment [73, 74].

2.2.2 Technical aspects

PGD adversarial training

The task of determining model parameters $\boldsymbol{\theta}^\star$ that are robust to adversarial perturbations of the inputs is cast in [50] as a *min-max* optimisation problem seeking to minimise *adversarial risk*, *i.e.*:

$$\boldsymbol{\theta}^\star \approx \hat{\boldsymbol{\theta}}^\star := \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[\max_{\boldsymbol{\delta}\in\mathbb{S}} \mathcal{L}\left(f\left(\boldsymbol{x}+\boldsymbol{\delta};\boldsymbol{\theta}\right), y\right) \right]$$

where $\mathcal{D}$ is the distribution on the examples $\boldsymbol{x}$ and the corresponding labels y, $f(\cdot;\boldsymbol{\theta})$ is a model with learnable parameters $\boldsymbol{\theta}$, $\mathcal{L}$ is a suitable loss function, and $\mathbb{S}$ is the set of allowed constrained perturbations. In the case of ℓ_p norm-bound perturbations of maximum magnitude ϵ, we can further specify $\mathbb{S} := \{\boldsymbol{\delta} \mid \|\boldsymbol{\delta}\|_p \leq \epsilon\}$.

The inner optimisation problem is solved, by Madry et al. [50], by *Projected Gradient Descent* (PGD), an iterative algorithm whose goal is the synthesis of an adversarial perturbation $\hat{\boldsymbol{\delta}} = \boldsymbol{\delta}^{(K)}$ after K *gradient ascent and projection* steps defined as:

$$\boldsymbol{\delta}^{(k+1)} \leftarrow \mathfrak{P}_{\mathbb{S}}\big(\boldsymbol{\delta}^{(k)} + \alpha \operatorname{sign}\left(\nabla_{\boldsymbol{\delta}^{(k)}} \mathcal{L}_{ce}(f(\boldsymbol{x}+\boldsymbol{\delta}^{(k)};\boldsymbol{\theta}), y)\right)\big)$$

where $\boldsymbol{\delta}^{(0)}$ is randomly sampled within $\mathbb{S}$, α is a hyperparameter (*step size*), $\mathcal{L}_{ce}$ is the cross-entropy function, and $\mathfrak{P}_{\mathbb{A}}$ is the Euclidean projection operator onto set $\mathbb{A}$, *i.e.*:

$$\mathfrak{P}_{\mathbb{A}}(\boldsymbol{a}) := \arg\min_{\boldsymbol{a}'\in\mathbb{A}} ||\boldsymbol{a}-\boldsymbol{a}'||_2 \ .$$

The outer optimisation is carried out by simply training $f(\cdot;\boldsymbol{\theta})$ on the examples found by PGD against the current model parameters – and their original pre-perturbation labels. The overall procedure just described constitutes PGD *adversarial training*. We additionally notice here that, when the number of iterations is fixed to 1 and $\delta^{(0)} = 0$, the resulting procedure is also called FGSM *adversarial training* [49].

In this work, we will use the shorthand notation ϵ_p to denote ℓ_p norm-bound perturbations of maximum magnitude ϵ.

(Conditional) variational autoencoders

Variational autoencoders (*VAEs*) [75, 76] allow to learn from data a generative distribution of the form $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x} \mid \boldsymbol{z})p(\boldsymbol{z})$, where the probability density $p(\boldsymbol{z})$ represents a prior over latent variable $\boldsymbol{z}$, and $p(\boldsymbol{x} \mid \boldsymbol{z})$ is the likelihood function, which can be used to sample data of interest $\boldsymbol{x}$, given $\boldsymbol{z}$.

Training is carried out by maximising a variational lower bound, $-\mathcal{L}_{\text{VAE}}(\boldsymbol{x})$, on the log-likelihood $\log p(\boldsymbol{x})$ – which is a proxy for the *Evidence Lower Bound* (*ELBO*) – *i.e.*:

$$-\mathcal{L}_{\text{VAE}}(\boldsymbol{x}) := \mathbb{E}_{q(\boldsymbol{z} \mid \boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z})] - \text{KL}(q(\boldsymbol{z} \mid \boldsymbol{x}) \| p(\boldsymbol{z}))$$

where $q(\boldsymbol{z} \mid \boldsymbol{x}) \approx p(\boldsymbol{z} \mid \boldsymbol{x})$ is an approximate posterior and $\text{KL}(\cdot\|\cdot)$ is the Kullback-Leibler divergence.

By parameterising the likelihood with a *decoder ANN* $p_{\boldsymbol{\theta}_\text{D}}(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}_\text{D}) \approx p(\boldsymbol{x} \mid \boldsymbol{z})$, and a possible variational posterior with an *encoder ANN* $q_{\boldsymbol{\theta}_\text{E}}(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}_\text{E}) \approx q(\boldsymbol{z} \mid \boldsymbol{x})$, the parameters $\boldsymbol{\theta}^\star_\text{D}$ of the generative model that best reproduces the data can be learnt – jointly with $\boldsymbol{\theta}^\star_\text{E}$ – as:

$$\begin{aligned}
&\boldsymbol{\theta}^\star_\text{E}, \boldsymbol{\theta}^\star_\text{D} := \\
&\quad \arg\min_{(\boldsymbol{\theta}_\text{E}, \boldsymbol{\theta}_\text{D})} \mathcal{L}_{\text{VAE}}(\boldsymbol{x}) = \\
&\quad \arg\min_{(\boldsymbol{\theta}_\text{E}, \boldsymbol{\theta}_\text{D})} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}} \left[-\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\theta}_\text{E}}(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}_\text{E})} \left[\log p_{\boldsymbol{\theta}_\text{D}}(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}_\text{D}) \right] + \text{KL}(q_{\boldsymbol{\theta}_\text{E}}(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}_\text{E}) \| p(\boldsymbol{z})) \right]
\end{aligned}$$

where $\mathcal{D}$ is the distribution over the (training) examples $\boldsymbol{x}$.

From a practical point of view, optimisation is based on the empirical evaluation of $\mathcal{L}_{\text{VAE}}(\boldsymbol{x}; \boldsymbol{\theta})$ on mini-batches of data, with the term $-\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\theta}_\text{E}}(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}_\text{E})} \left[\log p_{\boldsymbol{\theta}_\text{D}}(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}_\text{D}) \right]$ replaced by a *reconstruction cost*

$$\mathcal{L}_{\text{Reco}}(\boldsymbol{x}, \boldsymbol{x'}) \geq 0 \,|\, \mathcal{L}_{\text{Reco}}(\boldsymbol{x}, \boldsymbol{x'}) = 0 \iff x = x' \,.$$

The generation of new data according to the fitted model is achieved by sampling from

$$p_{\boldsymbol{\theta}^\star_\text{D}}(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}^\star_\text{D}) \Big|_{\boldsymbol{z} \sim p(\boldsymbol{z})}$$

i.e. decoding samples from the prior $p(\boldsymbol{z})$.

Conditional Variational Autoencoders [77, 78] extend *VAEs* by attaching a *conditioning tensor* $\boldsymbol{c}$ – expressing specific characteristics of each example – to both $\boldsymbol{x}$ and $\boldsymbol{z}$ during training. This allows the learning of a decoder model capable of conditional data generation. In such case, conditional sampling is achieved by:

$$\boldsymbol{x}_{\boldsymbol{c}_j} \sim p_{\boldsymbol{\theta}^\star_{\mathrm{D}}}(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{c}; \boldsymbol{\theta}^\star_{\mathrm{D}})\Big|_{\boldsymbol{z}\sim p(\boldsymbol{z});\ \boldsymbol{c}=\boldsymbol{c}_j}.$$

2.3 Preliminaries on *continuous-time quantum walks* (ch. 4)

As anticipated, chapter 4 is concerned with an application encompassing quantum information and the simulation of open quantum systems in continuous time. Given the peculiar shift in theoretical background required, we refer the reader to Griffiths and Schroeter [79], and to Nielsen and Chuang [80], for canonical introductions to – respectively – *quantum mechanics* and the more specific subfield of *quantum information*. Additionally, Quarteroni, Sacco and Saleri [81] can be referred to for an overview on numerical integration methods for (partial) differential equations, including the *Runge-Kutta* method we employ.

2.3.1 Technical aspects

In particular, in the chapter, we will be dealing with *Continuous Time Quantum Walks* over graphs. We refer as such to quantum systems whose unitary time-evolution operator is defined by:

$$U(t) := e^{-i\,\mathrm{H}\,t}$$

where $t \in \mathbb{R}$ represents time, i is the imaginary unit, and $\mathrm{H} = [\mathrm{H}_{ij}]$ is a Hamiltonian matrix such that, given the directed graph $\mathcal{G}$ associated with the CTQW,

$$\mathbb{C} \ni \mathrm{H}_{ij} \begin{cases} \in \mathbb{R}, & \text{if } i = j \\ \neq 0, & \text{if } \exists \text{ an edge in } \mathcal{G} \text{ linking vertices } i \text{ and } j \\ = 0, & \text{if } \nexists \text{ an edge in } \mathcal{G} \text{ linking vertices } i \text{ and } j \end{cases}$$

We describe the generic state of the quantum system as $|\Psi\rangle \in \mathcal{H}$, the element of a corresponding Hilbert space $\mathcal{H}$. With the Hamiltonian defined above, and calling $|\Psi_{t=0}\rangle$ the state of the system at time $t = 0$, it is possible to obtain the state of the system $|\Psi_{t=t^\star}\rangle$ at time $t^\star$ as $|\Psi_{t=t^\star}\rangle = U(t^\star)\ |\Psi_{t=0}\rangle$.

The canonical choice for a CTQW Hamiltonian, which we use, relies on the adjacency matrix $A = [a_{ij}]$ of the associated directed graph $\mathcal{G}$. In such case:

$$\mathbb{R} \ni \mathrm{H}_{ij} = \begin{cases} \kappa d_i, & \text{if } i = j \\ \kappa a_{ij}, & \text{otherwise} \end{cases}$$

where κ is an energy scale factor, and d_i is the degree of vertex i.

However, when models of this kind are utilised to describe complex or otherwise intractable phenomena whose site-specific energies are derived from observations or measured experimentally, such direct interpretation of the network Hamiltonian is not always as straightforward. This is the case, *e.g.*, of the *Fenna-Matthews-Olson complex* as described by Sgroi et al. [82].

A more detailed analysis of the formalism, together with a link with *discrete-time quantum walks* is offered by M N and Brun [83].

In our specific case, we are interested in investigating the behaviour of the system when in contact with an incident radiation of a given frequency, a *sink* able to absorb quantum excitations travelling through the system, and under the effect of eventual additional interventions. Thus, the system cannot be assumed to be closed. Describing the propagation of excitations through the different parts of the system as the result of a Markov process allows to describe time-evolution through a *Gorini–Lindblad*-type master equation [84, 85].

2.4 Preliminaries on *Koopman operator learning* (ch. 5)

2.4.1 Technical aspects

In this subsection, we briefly introduce *Koopman operator learning* via dynamics embedding [86, 87] — which constitutes the very foundation of the novel approach we will later describe.

Let us consider a dynamical system in discrete time, with a well-defined and fully-observable state $\boldsymbol{x} \in \mathbb{R}^N$, time-step $\Delta t > 0$, and subject to time-invariant dynamics. We call $\boldsymbol{x}_t$ the system state at a given time t and we are interested in predicting the state of the system at a later time $\boldsymbol{x}_{t+\Delta t}$. The key assumption of the method is that there exist at least one (potentially non-linear) change of coordinates $\alpha : \mathbb{R}^N \to \mathbb{R}^{d_k}$ for system state that renders the description of the dynamics linear across one time-step.

In such case, we want to find $\alpha : \mathbb{R}^N \to \mathbb{R}^{d_k}$ and $\mathrm{K} \in \mathbb{R}^{d_k \times d_k}$ (a matrix) such that:

$$\alpha(\boldsymbol{x}_{t+\Delta t}) = \mathrm{K}\, \alpha(\boldsymbol{x}_t);\ \forall t.$$

As a result, the state at an arbitrary future time $\alpha(\boldsymbol{x}_{t+n(\Delta t)})$ can be obtained as:

$$\alpha(\boldsymbol{x}_{t+n(\Delta t)}) = \mathrm{K}^n\, \alpha(\boldsymbol{x}_t).$$

Eventually, K could also be made an affine (instead of linear) operator acting on $\alpha(\boldsymbol{x}_t)$, with a similar time-evolution structure. The specific nature and parameterisation of α, as

well as the fitting procedures or optimisation schemes required in general to determine both α and K are beyond the scope of such short introductory exposition.

An aspect worth mentioning is that — for a given class of functions among which α has to be chosen, as well as for arbitrary αs (even universal approximators) with a fixed d_k — not all possible dynamical evolutions of $\boldsymbol{x}$ are always learnable in an exact fashion. This is the case *e.g.* of otherwise linearisable dynamical systems whose dynamics sharply changes at specific states.

2.5 Preliminaries on *Forward-Forward* and *neuronal ensembles* (ch. 6)

2.5.1 Related works

Forward-Forward

The Forward-Forward algorithm [34] is a recently proposed learning algorithm for artificial neural networks, whose main premise is the ability to overcome the notorious biological implausibility of the *Backpropagation* algorithm [32]. In fact, while the effectiveness of Backprop makes it the standard algorithm for training neural networks, it is based on biologically unrealistic assumptions that usually conflict with the local nature of mutual neuron interaction, such as the need to propagate information forwards and backwards through the network [37].
Forward-Forward owes its name to the fact that it replaces the backward pass with an additional forward pass. The two forward passes are executed on different data, named *positive* and *negative* data. During training, the objective of Forward-Forward is to maximise a so-called *goodness* function of the neural activations (*e.g.* the ℓ_p norm) on positive data and minimise it on negative data. In a simple image classification setting, such as the one we adopt in this chapter, one could encode a class label at the border of images, by one-hot encoding it with a white pixel (as shown in Figure 2.1). Following the definition from Hinton [34], positive data are those for which the encoded label matches the ground truth label, while the opposite holds for negative data. Layers are trained separately and sequentially, and learn to discriminate between positive and negative data by maximising and minimising their goodness, according to the data presented. Crucially, activations are normalised before being passed to the subsequent layer, to prevent layers from relying on the goodness computed by their predecessors. From the biological point of view, normalisation is known to be a form of *canonical neural computation* [88]. At test time, when a new unlabelled sample has to be categorised, many copies of the image are created, each with a different one-hot encoded label. These are then fed into the neural network to obtain a goodness score. Finally, the image gets classified in the category that produced the maximum goodness value.

In the original Hinton [34], satisfactory classification results are reported on the standard handwritten digit recognition dataset Mnist, with the definition of positive and negative data described above, and using the ℓ_2 or ℓ_1 norm of activations as goodness function. In a recent theoretical work, it has been analytically shown that, under somewhat mild assumptions, sparsity emerges in Forward-Forward layers [90] as a consequence of optim-

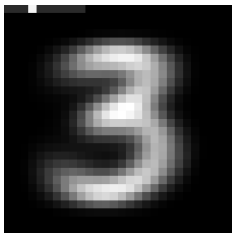

Figure 2.1: *Example of datapoint from the MNIST dataset [89], prepared for Forward-Forward learning, as described in Hinton [34]. Notice the 10-pixel-long stripe in the top-left corner whose 3rd pixel (corresponding to class 3) has higher luminance than the others.*

ising the Forward-Forward loss. While these formal results are derived for single linear layers considered alone, they offer a theoretical grounding for our experimental findings. Recent studies inspired by the Forward-Forward learning procedure have expanded its applicability across various architectures, notably achieving enhanced performance in Convolutional Neural Networks (CNNs) [91, 92]. Additionally, other works have proposed alternative goodness functions and explored the specific contributions of individual neurons to the classification process, shedding light on the interpretability and adaptability of the approach [93].

We illustrate properties of representations obtained in Forward-Forward networks, that are reminiscent of what is found the neocortex and hippocampus, where ensembles of a few number of units activate consistently in response to similar stimuli. We discuss properties of neuronal ensembles in the following section.

Neuronal ensembles

In Neuroscience, neuronal ensembles are defined as sparse groups of neurons that co-activate either spontaneously or in response to sensory stimuli. These ensembles, rather than individual neurons, have long been proposed as emergent functional units of cortical activity, playing critical roles in sensory processing, memory, and behaviour [30, 31, 94–99]. Recent reviews, such as Yuste, Cossart and Yaksi [31], provide a comprehensive overview of the concept and its implications.

The importance of ensembles has been increasingly corroborated by experimental studies, enabled by advances in techniques like calcium imaging, which allow for simultaneous recording of large-scale neural activity at single-cell resolution [100]. For example, Miller et al. [30] demonstrated that, during visual processing, cortical spiking activity is dominated by ensembles whose properties cannot be explained by the independent activity of individual neurons. These ensembles are activated both by sensory stimuli (*e.g.* visual inputs) and by spontaneous network activity, suggesting that they represent intrinsic functional building blocks of cortical responses. Notably, single neurons often participate in multiple ensembles, thereby enhancing the network's encoding potential [101, 102]. Further evidence from Yoshida and Ohki [103] showed that sparse ensembles in the primary visual cortex (V1) are elicited by visual stimuli. Images can be decoded reliably from the activity of a small subset of highly responsive neurons, with additional neurons either failing to improve or even degrading decoding performance. These

findings underscore the efficiency of sparse representations, likely facilitated by partially overlapping receptive fields. This arrangement enables robust and efficient encoding of visual information, making sparse ensembles an optimal strategy for downstream processing.

The presence and functionality of ensembles are not limited to specific species or sensory modalities. Studies in various animal models have revealed their role in diverse neural processes [104, 105], and recent findings suggest they may even contribute to conscious experience [106]. Moreover, ensembles have demonstrated remarkable stability over time. For instance, Pérez-Ortega, Alejandre-García and Yuste [107] showed that neuronal ensembles can persist for weeks, supporting their potential involvement in long-term representations of perceptual states or memories.

Technological advancements have also enabled not just the visualization but the direct stimulation of ensembles, allowing to "play the piano" with ensembles of neurons [99]. All-optical approaches, such as those described by Packer et al. [108] and Carrillo-Reid et al. [98], have shown that repeatedly stimulating specific groups of neurons in V1 can imprint ensembles that remain spontaneously active even after a day. These imprinted ensembles exhibit pattern completion, where activating a subset of neurons can recall the entire ensemble. Remarkably, this effect persists long after the initial stimulation, and experiments have demonstrated causal links between ensemble activation and behaviour [99].

Finally, the concept of neuronal ensembles has inspired computational models. For instance, Doi and Lewicki [109] demonstrated that sparse and redundant representations are optimal for encoding natural images, particularly when neurons are unreliable, a result corroborated by earlier studies [110, 111]. These computational frameworks align with biological observations, suggesting that sparse ensemble representations are both efficient and robust mechanisms for encoding sensory information.

Chapter 3

Blending *adversarial training* and *representation-conditional purification* via *aggregation* improves adversarial robustness

3.1 Introduction

Vulnerability to adversarial attacks [10, 11] – *i.e.* the presence of inputs, usually crafted on purpose, capable of catastrophically altering the behaviour of high-dimensional models [112] – constitutes a major hurdle towards ensuring the compliance of deep learning systems with the behaviour expected by modellers and users, and their adoption in safety-critical scenarios or tightly-regulated environments. This is particularly true for adversarially-*perturbed* inputs, where a norm-constrained perturbation – often hardly detectable by human inspection [12, 113] – is added to an otherwise legitimate input, with the intention of eliciting an anomalous response [114].

Given the widespread nature of the issue [115], and the serious concerns raised about the safety and reliability of models learnt from data in the lack of appropriate mitigations [116], adversarial attacks have been extensively studied. However, obtaining generally robust machine learning (*ML*) systems remains a longstanding issue, and a major open challenge.

Research in the field has been driven by two opposing yet complementary efforts. On the one hand, the study of *failure modes* in existing models and defences, with the goal of understanding their origin and developing stronger attacks with varying degrees of knowledge and control over the target system [11, 49, 117, 118]. On the other hand, the construction of increasingly capable defence mechanisms. Although alternatives have been explored [119–122], most of the latter is based on adequately leveraging *adversarial training* [49, 50, 53–57, 123–125], *i.e.* training a *ML* model on a dataset composed of (or enriched with) adversarially-perturbed inputs associated with their correct, *pre-perturbation* labels. In fact, adversarial training has been the only technique capable of consistently providing an acceptable level of defence [51], while still incrementally

improving up to the current *state-of-the-art* [56–58].

Another defensive approach is that of *adversarial purification* [70, 72], where a generative model is used – similarly to denoising – to recover a safe version of the input before classification is performed. Nonetheless, such attempts have generally fallen short of expectations due to inherent limitations of the generative models used in early attempts [68], or due to decreases in robust accuracy[1] when attacked *end-to-end* [60, 74] – resulting in subpar robustness if the defensive structure is known to the adversary [118]. More recently, the rise of diffusion-based generative models [126] and their use for purification have enabled more successful results of this kind [68, 73] – although at the cost of longer training and inference times, and a much brittler robustness evaluation [73, 74].

In this chapter, we design a novel adversarial defence for supervised image classification, dubbed Carso (*Counter-Adversarial Recall of Synthetic Observations*). The approach relies on an adversarially-trained classifier (called hereinafter simply *the classifier*), endowed with a stochastic generative model (called hereinafter *the purifier*). Upon classification of a potentially-perturbed input, the latter learns to generate – from the tensor[2] of (pre)activations registered at neuron level in the former – samples from a distribution of plausible, perturbation-free reconstructions. At inference time, some of these samples are classified by the very same *classifier*, and the original input is robustly labelled by aggregating its many outputs in the form of a normalised doubly-exponential logit product. This method – to the best of our knowledge the first attempt to organically merge the *adversarial training* and *purification* paradigms – avoids the vulnerability pitfalls typical of the mere stacking of a purifier and a classifier [60, 74], while still being able to take advantage of independent incremental improvements to adversarial training or generative modelling.

An empirical assessment[3] of the defence in the ℓ_∞ *white-box* setting is provided, using a *conditional* [77, 78] *variational autoencoder* [75, 76] as the purifier and existing *state-of-the-art* adversarially pre-trained models as classifiers. Such choices are meant to give existing approaches – and the *adversary* attacking our architecture *end-to-end* as part of the assessment – the strongest advantage possible. Yet, in all scenarios considered, Carso improves significantly the robustness of the pre-trained classifier – even against attacks specifically devised to fool stochastic defences like ours. Remarkably, with a modest *clean* accuracy penalty, our method improves by a significant margin the current *state-of-the-art* for Cifar-10 [130], Cifar-100 [130], and TinyImageNet-200 [131] ℓ_∞ robust classification accuracy against AutoAttack [52].

In summary, the chapter makes the following contributions:

- The description of Carso, a novel adversarial defence method synergistically blending *adversarial training* and *adversarial purification*, thanks to *representation-conditional* purification and a dedicated *robust aggregation* strategy;
- A collection of relevant technical details fundamental to its successful training and

[1] The *test set accuracy* of the frozen-weights trained classifier – computed on a dataset entirely composed of adversarially-perturbed examples generated against that specific model.

[2] Which we call *internal representation*.

[3] Implementation of the method and code for the experiments rely on *PyTorch* [127], `AdverTorch` [128], and `ebtorch` [129].

use, originally developed for the *purifier* being a *conditional variational autoencoder* – but in principle not bound to such setting;

- An experimental assessment of the method, against standardised benchmark adversarial attacks – showing higher robust accuracy *w.r.t.* to existing *state-of-the-art* adversarial training and purification approaches.

The rest of the chapter is structured as follows. In subsection 2.2.1 we provided an overview of selected contributions in the fields of *adversarial training* and *purification-based* defences – with focus on image classification. In subsection 2.2.2, an introduction was given to two integral parts of our experimental assessment: PGD adversarial training and conditional variational autoencoders. Section 3.2 is devoted to the intuition behind CARSO, its architectural description, and the relevant technical details that allow it to work effectively. Section 3.3 contains details about the experimental setup, results, comments, and limitations. Section 3.4 concludes the chapter and outlines directions of future development.

3.2 Structure of CARSO

The core ideas informing the design of our method are driven more by *first principles* rather than arising from specific contingent requirements. This section discusses such ideas, the architectural details of CARSO, and a group of technical aspects fundamental to its training and inference processes.

3.2.1 Architectural overview and principle of operation

From an architectural point of view, CARSO is essentially composed of two *ANN* models – a *classifier* and a *purifier* – operating in close synergy. The former is trained on a given classification task, whose inputs might be adversarially corrupted at inference time. The latter learns to generate samples from a distribution of potential input reconstructions, tentatively free from adversarial perturbations. Crucially, the *purifier* has only access to the internal representation of the *classifier* – and not even directly to the perturbed input – to perform its task.

During inference, for each input, the internal representation of the *classifier* is used by the *purifier* to synthesise a collection of tentatively unperturbed input reconstructions. Those are classified by the same *classifier*, and the resulting outputs are aggregated into a final *robust prediction*.

There are no specific requirements for the classifier, whose training is completely independent of the use of the model as part of CARSO. However, training it adversarially generally improves the robust accuracy of the overall system (see section A.3 for an ablation study on that matter), also allowing it to benefit from established adversarial training techniques.

The purifier is also independent of specific architectural choices, provided it is capable of stochastic conditional data generation at inference time, with the internal representation of the classifier used as conditioning.

In the rest of the chapter, we employ a *state-of-the-art* adversarially pre-trained WideResNet model as the classifier, and a purpose-built *conditional variational autoencoder* as the purifier, the latter operating decoder-only during inference. Such choice was driven by the deliberate intention to assess the adversarial robustness of our method in its worst-case scenario against a *white-box* attacker, and with the least advantage compared to existing approaches based solely on adversarial training.

In fact, the decoder of a conditional VAE allows for exact algorithmic differentiability [18] *w.r.t.* its conditioning set, thus averting the need for backward-pass approximation [132] in generating *end-to-end* adversarial attacks against the entire system, and preventing (un)intentional robustness inflation by gradient obfuscation [132]. The same cannot be said [73] for more capable and modern purification models, such as those based *e.g.* on diffusive processes, whose proper robustness assessment is still in the process of being thoroughly understood [74].

A downside of such choice is represented by the reduced effectiveness of the decoder in the synthesis of complex data, due to well-known model limitations. In fact, we experimentally observe a modest increase in reconstruction cost for non-perturbed inputs, which in turn may limit the *clean* accuracy of the entire system. Nevertheless, we defend the need for a fair and transparent robustness evaluation, such as the one provided by the use of a VAE-based purifier, in the evaluation of any novel architecture-agnostic adversarial defence.

A diagram of the whole architecture is shown in Figure 3.1, and its detailed principles of operation are recapped below. Additionally, an ablation study investigating the need for either the *classifier* or the *purifier* being trained on adversarially-perturbed inputs is provided in section A.3.

Training

At training time, adversarially-perturbed examples are generated against the *classifier*, and fed to it. The tensors containing the *classifier* (pre)activations across the network are then extracted. Finally, the conditional *VAE* serving as the *purifier* is trained on perturbation-free input reconstruction, conditional on the corresponding previously-extracted internal representations, and using pre-perturbation examples as targets.

Upon completion of the training process, the encoder network is discarded, as it will not be used for inference.

Inference

The example requiring classification is fed to the *classifier*. Its corresponding internal representation is extracted and used to condition the generative process described by the decoder of the *VAE*. Stochastic latent variables are repeatedly sampled from the original priors, which are given by an *i.i.d.* multivariate Standard Normal distribution. Each element in the resulting set of reconstructed inputs is classified by the same *classifier*, and the individually predicted class logits are aggregated. The result of such aggregation constitutes the robust prediction of the input class.

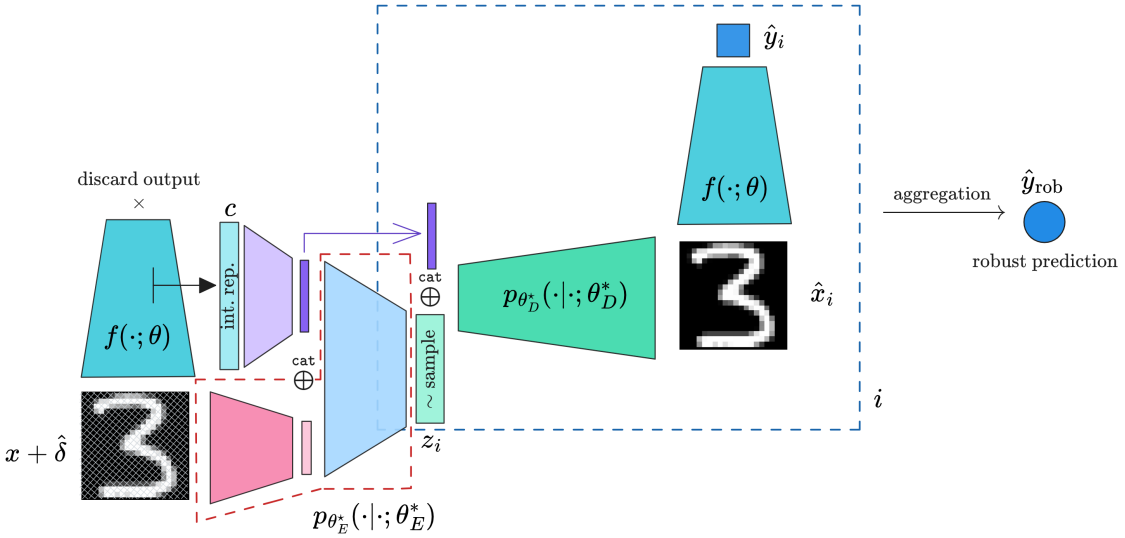

Figure 3.1: *Schematic representation of the CARSO architecture used in the experimental phase of this work. The subnetwork bordered by the red dashed line is used only during the training of the purifier. The subnetwork bordered by the blue dashed line is re-evaluated on different random samples $\boldsymbol{z}_i$ and the resulting individual $\hat{y}_i$ are aggregated into $\hat{y}_{rob}$. The classifier $f(\cdot;\boldsymbol{\theta})$ is always kept frozen; the remaining network is trained on $\mathcal{L}_{VAE}(\boldsymbol{x}, \hat{\boldsymbol{x}})$. More precise details on the functioning of the networks are provided in subsection 3.2.1.*

Remarkably, the only link between the initial potentially-perturbed input and the resulting purified reconstructions (and thus the predicted class) happens through the *internal representation* of the classifier, which serves as a *featurisation* of the original input. The whole process is exactly differentiable *end-to-end*, and the only potential hurdle to the generation of adversarial attacks against the entire system is the stochastic nature of the decoding – which is easily tackled by *Expectation over Transformation* [133].

3.2.2 A first-principles justification

If we consider a trained *ANN* classifier, subject to a successful adversarial attack by means of a slightly perturbed example, we observe that – both in terms of ℓ_p magnitude and human perception [58] – a small variation on the input side of the network is amplified to a significant amount on the output side, thanks to the layerwise processing by the model. Given the deterministic nature of such processing at inference time, we speculate that the collection of (pre)activation values within the network, along the forward pass, constitutes a richer characterisation of such an amplification process compared to the knowledge of the input alone. Indeed, as we do, it is possible to learn a direct mapping from such featurisation of the input, to a distribution of possible perturbation-free input reconstructions – in a way that takes advantage of such characterisation.

3.2.3 Hierarchical input and internal representation encoding

Training a conditional VAE requires [77] that the conditioning set $\boldsymbol{c}$ is concatenated to the input $\boldsymbol{x}$ before encoding occurs, and to the sample of latent variables $\boldsymbol{z}$ right

before decoding. The same is also true, with the suitable adjustments, for any conditional generative approach where the target and the conditioning set must be processed jointly.

In order to ensure the usability and scalability of CARSO across the widest range of input data and classifier models, we propose to perform such processing in a hierarchical and partially disjoint fashion between the input and the conditioning set. In principle, the encoding of $\boldsymbol{x}$ and $\boldsymbol{c}$ can be performed by two different and independent subnetworks, until some form of joint processing must occur. This allows to retain the overall architectural structure of the purifier, while having finer-grained control over the inductive biases [134] deemed the most suitable for the respective variables.

In the experimental phase of our work, we encode the two variables independently. The input is compressed by a multilayer convolutional neural network (CNN). The internal representation – which in our case is composed of differently sized multi-channel *images* – is processed *layer by layer* by independent multilayer CNNs (responsible for encoding local information), whose flattened outputs are finally concatenated and compressed by a fully-connected layer (modelling inter-layer correlations in the representation). The resulting compressed input and conditioning set are then further concatenated and jointly encoded by a fully-connected network (FCN).

In order to use the VAE decoder at inference time, the compression machinery for the conditioning set must be preserved after training, and used to encode the internal representations extracted. The input encoder may be discarded instead.

3.2.4 Adversarially-balanced batches

Training the purifier in representation-conditional input reconstruction requires having access to adversarially-perturbed examples generated against the classifier, and to the corresponding clean data. Specifically, we use as input a mixture of *clean* and adversarially *perturbed* examples, and the clean input as the target.

Within each epoch, the *training set* of interest is shuffled [135, 136], and only a fixed fraction of each resulting batch is adversarially perturbed. Calling ϵ the maximum ℓ_p perturbation norm bound for the threat model against which the *classifier* was adversarially pre-trained, the portion of perturbed examples is generated by an even split of $\text{FGSM}_{\epsilon/2}$, $\text{PGD}_{\epsilon/2}$, FGSM_{ϵ}, and PGD_{ϵ} attacks.

Any smaller subset of attack types and strengths, or a detailedly unbalanced batch composition, experimentally results in a worse-performing purification model. More details justifying such choice are provided in section A.1.

3.2.5 Robust aggregation strategy

At inference time, many different input reconstructions are classified by the *classifier*, and the respective outputs concur to the settlement of a *robust prediction*.

Calling l_i^α the output logit associated with class $i \in \{1, \dots, C\}$ in the prediction by the classifier on sample $\alpha \in \{1, \dots, N\}$, we adopt the following novel aggregation strategy:

$$P_i := \frac{1}{Z} \prod_{\alpha=1}^{N} e^{e^{l_i^\alpha}}$$

with P_i being the aggregated probability of membership in class i, Z a normalisation constant such that $\sum_{i=1}^{C} P_i = 1$, and e Euler's number.

Such choice produces a *robust prediction* much harder to overtake in the event that an adversary selectively targets a specific input reconstruction.

The rationale leading to the choice of such specific *robust aggregation strategy* is an attempt to answer to the following question: 'How is it possible to aggregate the results of an ensemble of classifiers in a way such that it is hard to tilt the balance of the ensemble by attacking only a few of its members?'. The same reasoning can be extended to the *reciprocal* problem we are trying to solve here, where different input reconstructions obtained from the same potentially perturbed input are classified by the same model (the *classifier*).

Heuristic analysis

Far from providing a satisfactory answer, we can analyse the behaviour of our aggregation strategy as the logit associated with a given model and class varies across its domain, under the effect of adversarial interventions. Comparison with existing *probability averaging* and *logit averaging* aggregation strategies [137] should provide a heuristic justification of our choice.

We recall *logit averaging* aggregation

$$P_i := \frac{1}{Z} e^{\frac{1}{N}\sum_{\alpha=1}^{N} l_i^\alpha} = \frac{1}{Z} \prod_{\alpha=1}^{N} e^{\frac{1}{N} l_i^\alpha} = \frac{1}{Z} \left(\prod_{\alpha=1}^{N} e^{l_i^\alpha} \right)^{\frac{1}{N}}$$

and *probability averaging* aggregation

$$P_i := \frac{1}{Z} \sum_{\alpha=1}^{N} \frac{e^{l_i^\alpha}}{\sum_{j=1}^{C} e^{l_j^\alpha}} = \frac{1}{Z} \sum_{\alpha=1}^{N} e^{l_i^\alpha} \frac{1}{Q^\alpha}$$

where $Q^\alpha = \sum_{j=1}^{C} e^{l_j^\alpha}$.

Additionally, since $l_i^\alpha \in \mathbb{R}, \forall l_i^\alpha$, $\lim_{x \to -\infty} e^x = 0$ and $e^0 = 1$, we can observe that $e^{l_i^\alpha} > 0$ and $e^{e^{l_i^\alpha}} > 1, \forall l_i^\alpha$.

Now, we consider a given class $i^\star$ and the *classifier* prediction on a given input reconstruction $\alpha^\star$, and study the potential effect of an adversary acting on $l_{i^\star}^{\alpha^\star}$. This adversarial intervention can be framed in two complementary scenarios: either the class $i^\star$ is correct and the adversary aims to decrease its membership probability, or the class $i^\star$ is incorrect and the adversary aims to increase its membership probability. In any case, the adversary should comply with the ϵ_∞-boundedness of its perturbation on the input.

Logit averaging

In the former scenario, the product of $e^{l_i^\alpha}$ terms can be arbitrarily deflated (up to zero) by lowering the $l_{i^\star}^{\alpha^\star}$ logit only. In the latter scenario, the logit can be arbitrarily inflated, and such effect is only partially suppressed by normalisation by Z (a sum of ${}^1/_N$-exponentiated terms, $N \geq 1$).

Probability averaging

In the former scenario, although the effect of the deflation of a single logit is bounded by $e^{l_{i^\star}^{\alpha^\star}} > 0$, two attack strategies are possible: either decreasing the value of $l_{i^\star}^{\alpha^\star}$ or increasing the value of $Q^{\alpha^\star}$, giving rise to complex combined effects. In the latter scenario, the reciprocal is possible, *i.e.* either inflating $l_{i^\star}^{\alpha^\star}$ or deflating $Q^{\alpha^\star}$. Normalisation has no effect in both cases.

Ours

In the former scenario, the effect of logit deflation on a single product term is bounded by $e^{e^{l_{i^\star}^{\alpha^\star}}} > 1$, thus exerting only a minimal collateral effect on the product, through a decrease of Z. This effectively prevents *aggregation takeover* by logit deflation. Similarly to *logit averaging*, in the latter scenario, the logit can be arbitrarily inflated. However, in this case, the effect of normalisation by Z is much stronger, given its increased magnitude (addends are not ${}^1/_N$-exponentiated, $N \geq 1$).

From such a comparison, our aggregation strategy is the only one that strongly prevents *adversarial takeover* by *logit deflation*, while still defending well against perturbations targeting *logit inflation*.

The experimental assessment of our aggregation strategy, in the specific scenarios considered and against the more common *probability averaging* and *logit averaging* strategies, is deferred to after the description of our experimental setup. As such, it is contained in subsection 3.3.3.

3.3 Experimental assessment

Experimental evaluation of our method is carried out in terms of *robust* and *clean* image classification accuracy within three different scenarios (*a*, *b*, and *c*), determined by different classification tasks. The *white-box* threat model with a fixed ℓ_∞ norm bound is assumed throughout, as it generally constitutes the most demanding setup for adversarial defences.

3.3.1 Setup

Data

The CIFAR-10 [130] dataset is used in *scenario (a)*, the CIFAR-100 [130] dataset is used in *scenario (b)*, whereas the TINYIMAGENET-200 [131] dataset is used in *scenario (c)*.

Architectures

A WideResNet-28-10 model is used as the *classifier*, adversarially pre-trained on the respective dataset – the only difference between scenarios being the size of the inputs, and the number of output logits: 10 in *scenario (a)*, 100 in *scenario (b)*, and 200 in *scenario (c)*.

The purifier is composed of a conditional VAE, processing inputs and internal representations in a partially disjoint fashion, as explained in subsection 3.2.3. The input is compressed by a two-layer CNN; the internal representation is instead processed layer-wise by independent CNNs (three-layered in *scenarios (a)* and *(b)*, four-layered in *scenario (c)*) whose outputs are then concatenated and compressed by a fully-connected layer. A final two-layer FCN jointly encodes the compressed input and conditioning set, after the concatenation of the two. A six-layer deconvolutional network is used as the decoder.

More precise details on all architectures are given in section A.2.

Outer minimisation

In all scenarios, *classifiers* are acquired as pre-trained models, using publicly available weights provided by the respective authors. Therefore, in *scenarios (a)* and *(b)*, the *classifier* is trained according to Cui et al. [56]; in *scenario (c)*, according to Wang et al. [55].

The *purifier* is trained on the *VAE* loss, using *summed pixel-wise channel-wise* binary cross-entropy as the reconstruction cost. Optimisation is performed by RAdam+Lookahead [138, 139] with a learning rate schedule that presents a linear warm-up, a plateau phase, and a linear annealing [140]. To promote the learning of meaningful reconstructions during the initial phases of training, the *KL divergence* term in the VAE loss is suppressed for an initial number of epochs. Afterwards, it is linearly modulated up to its actual value, along a fixed number of epochs (β *increase*) [141]. The initial and final epochs of such modulation are reported in Table A.11.

Additional scenario-specific details are provided in section A.2.

Inner minimisation

$\epsilon_\infty = {}^8\!/_{255}$ is set as the perturbation norm bound, as customary in the empirical ℓ-norm adversarial robustness community [142].

Adversarial examples against the *purifier* are obtained, as explained in subsection 3.2.4, by $\text{FGSM}_{\epsilon/2}$, $\text{PGD}_{\epsilon/2}$, FGSM_{ϵ}, and PGD_{ϵ}, in a *class-untargeted* fashion on the cross-entropy loss. In the case of PGD, gradient ascent with a step size of $\alpha = 0.01$ is used.

The complete details and hyperparameters of the attacks are described in section A.2.

Evaluation

In each scenario, we report the *clean* and *robust* test-set accuracy – the latter by means of AutoAttack [52] – of the *classifier* alone, and that of the corresponding Carso architecture.

For the *classifier* alone, the *standard* version of AutoAttack (*AA*) is used: *i.e.*, the worst-case accuracy on a mixture of AutoPgd on the cross-entropy loss [52] with 100 steps, AutoPgd on the *difference of logits ratio* loss [52] with 100 steps, Fab [143] with 100 steps, and the *black-box* Square attack [144] with 5000 queries.

In the evaluation of the Carso architecture, the number of reconstructed samples per input is set to 8, the logits are aggregated as explained in subsection 3.2.5, and the output class is finally selected as the $\arg\max$ of the aggregation. Due to the stochastic nature of the *purifier*, robust accuracy is assessed by a version of AutoAttack suitable for stochastic defences (*randAA*) – composed of AutoPgd on the cross-entropy and *difference of logits ratio* losses, across 20 *Expectation over Transformation* (EoT) [133] iterations with 100 gradient ascent steps each. In the specific case of *scenario (a)*, we also assess our method by the Pgd+EoT pipeline proposed by Lee and Kim [74], as explained in subsection 3.3.2.

Computational infrastructure

All experiments were performed on an *NVIDIA DGX A100* system. Training in *scenarios (a)* and *(c)* was run on 8 *NVIDIA A100* GPUs with 40 GB of dedicated memory each; in *scenario (b)* 4 of such devices were used. Elapsed training time for the purifier in all scenarios is reported in Table 3.1.

Table 3.1: *Elapsed running time for training the purifier in the different scenarios considered.*

Scenario	*(a)*	*(b)*	*(c)*
Elapsed training time	159 min	138 min	213 min

3.3.2 Results and discussion

An analysis of the experimental results is provided in the subsection that follows, whereas their systematic exposition is given in Table 3.2. Results obtained by using deliberately worse-performing pretrained *classifiers*, as well as a broader comparison with existing adversarial defences from literature, are provided in section A.4.

Scenario (a)

Comparing the robust accuracy of the *classifier* model used in *scenario (a)* [56] with that resulting from the inclusion of the same model in the Carso architecture, we observe a $+8.4\%$ increase. This is counterbalanced by a -5.6% clean accuracy decrease. The same version of Carso further provides a $+2.42$ robustness increase *w.r.t.* the current best AT-trained model [58] that employs a $\sim 4\times$ larger WideResNet-96-16 model.

In addition, our method provides a remarkable $+9.72\%$ increase in robust accuracy *w.r.t.* to the best adversarial purification approach [145], a diffusion-based purifier. However, the comparison is not as straightforward. In fact, the original paper [145] reports a robust accuracy of 78.12% using AutoAttack on the gradients obtained via the adjoint method [68]. As noted in Lee and Kim [74], such evaluation (which uses the version of

Table 3.2: *Clean (results in italic) and adversarial (results in upright) accuracy for the different models and datasets used in the respective scenarios. The following abbreviations are used:* `Scen`*: scenario considered;* `AT/Cl`*: clean accuracy for the adversarially-pretrained model used as the classifier, when considered alone;* `C/Cl`*: clean accuracy for the* Carso *architecture;* `AT/AA`*: robust accuracy (by the means of* AutoAttack*) for the adversarially-pretrained model used as the classifier, when considered alone;* `C/randAA`*: robust accuracy for the* Carso *architecture, when attacked end-to-end by* AutoAttack *for randomised defences;* `Best AT/AA`*: best robust accuracy result for the respective dataset (by the means of* AutoAttack*), obtained by adversarial training alone (any model);* `Best P/AA`*: best robust accuracy result for the respective dataset (by the means of* AutoAttack*), obtained by adversarial purification (any model). Robust accuracies in round brackets are obtained using the* Pgd+EoT *[74] pipeline, developed for diffusion-based purifiers. The best clean and robust accuracies per dataset are shown in* ***bold****. The clean accuracies for the models referred to in the* `Best` *columns are shown in Table A.14 (in section A.4).*

Scen.	Dataset	AT/Cl	C/Cl	AT/AA	C/randAA (Pgd+EoT)	Best AT/AA	Best P/AA (Pgd+EoT)
(a)	Cifar-10	***0.9216***	*0.8686*	0.6773	**0.7613** (0.7689)	0.7371	0.7812 (0.6641)
(b)	Cifar-100	***0.7385***	*0.6806*	0.3918	**0.6665**	0.4267	0.4609
(c)	TinyImageNet-200	***0.6519***	*0.5632*	0.3130	**0.5356**	0.3130	

AutoAttack that is unsuitable for stochastic defences) leads to a large overestimation of the robustness of diffusive purifiers. As suggested in Lee and Kim [74], Lin et al. [145] re-evaluate the robust accuracy according to a more suitable pipeline (Pgd+EoT, whose hyperparameters are shown in Table A.9), obtaining a much lower robust accuracy of 66.41%. Consequently, we repeat the same evaluation for Carso and compare the worst-case robustness among the two. In line with typical AT methods, and unlike diffusive purification, the robustness of Carso assessed by means of *randAA* remains lower *w.r.t.* that achieved by Pgd+EoT.

Scenario (b)

Moving to *scenario (b)*, Carso achieves a robust accuracy increase of $+27.47\%$ *w.r.t.* the *classifier* alone [56], balanced by a -5.79% decrease in clean accuracy. Our approach also improves upon the robust accuracy of the best AT-trained model [55] (WideResNet-70-16) by $+23.98\%$. In the absence of a reliable robustness evaluation by means of Pgd+EoT for the best purification-based method [145], we still obtain a $+20.25\%$ increase in robust accuracy upon its (largely overestimated) AA result.

Scenario (c)

In *scenario (c)*, Carso improves upon the *classifier* alone [55] (which is also the best AT-based approach for TinyImageNet-200) by $+22.26\%$. A significant clean accuracy toll is imposed by the relative complexity of the dataset, *i.e.* -8.87%. In this setting, we

lack any additional purification-based methods.

Assessing the impact of *gradient obfuscation*

Although the architecture of Carso is algorithmically differentiable *end-to-end* – and the integrated diagnostics of the *randAA* routines identified no pitfalls during the assessment – we additionally guard against the eventual gradient obfuscation [132] induced by our method by repeating the evaluation at $\epsilon_\infty = 0.95$, verifying that the resulting robust accuracy stays below random chance [146]. Results are shown in Table 3.3.

Table 3.3: *Robust classification accuracy against AutoAttack, for $\epsilon_\infty = 0.95$, as a way to assess the (lack of) impact of gradient obfuscation on robust accuracy evaluation.*

Scenario	*(a)*	*(b)*	*(c)*
$\epsilon_\infty = 0.95$ *acc.*	<0.047	<0.010	≈0.0

3.3.3 Experimental justification of the *robust aggregation strategy*

To further corroborate the choice of the specific aggregation function described in subsection 3.2.5, we repeat the assessment of Carso under the same conditions described in section 3.3, the only difference being the use of the alternative aggregation functions analysed in subsection 3.2.5 (*i.e. logit* and *probability averaging* aggregation). Results, in terms of both *clean* and *robust* accuracy, are shown in Table 3.4.

Table 3.4: *Clean (results in italic) and adversarial (results in upright) accuracy for alternative aggregation strategies used within Carso. The following abbreviations are used:* `Scen`*: scenario considered;* `C/Cl (L.A.)`*: clean accuracy of Carso with logit average aggregation;* `C/randAA (L.A.)`*: robust accuracy of Carso with logit average aggregation, assessed by means of AutoAttack for stochastic defences;* `C/Cl (P.A.)`*: clean accuracy of Carso with probability average aggregation;* `C/randAA (P.A.)`*: robust accuracy of Carso with probability average aggregation, assessed by means of AutoAttack for stochastic defences;* `C/Cl`*: clean accuracy of Carso with our proposed aggregation;* `C/randAA`*: robust accuracy of Carso with our proposed aggregation, assessed by means of AutoAttack for stochastic defences. Results from the last two columns mirror those of Table 3.2.*

Scen.	Dataset	C/Cl (L.A.)	C/randAA (L.A.)	C/Cl (P.A.)	C/randAA (P.A.)	C/Cl	C/randAA
(a)	Cifar-10	*0.8688*	0.0086	*0.8688*	0.0092	*0.8686*	**0.7613**
(b)	Cifar-100	*0.6808*	0.0436	*0.6807*	0.0439	*0.6806*	**0.6665**

In Table 3.5, we additionally provide the same *clean* and *robust* accuracy assessment for the naive and non algorithmically-differentiable *majority voting* aggregation strategy. In such regard, it is important to remark that the non-differentiability of majority voting results in the vast portion ($> 99\%$) of gradient samples used by AutoAttack being either zero or *not-a-number*. Thus, the result of robustness assessment has to be considered

Table 3.5: *Clean (results in italic) and adversarial (results in upright) accuracy resulting from the use of the majority vote aggregation strategy within Carso. The following abbreviations are used:* `Scen`*: scenario considered;* `C/Cl (M.V.)`*: clean accuracy of Carso with majority vote aggregation;* `C/randAA (M.V.)`*: robust accuracy of Carso with majority vote aggregation, assessed by means of AutoAttack for stochastic defences. Almost the entirety of gradient samples computed by AutoAttack has been deemed unreliable by integrated diagnostics, and the robust accuracy results must be considered untrustworthy.*

Scen.	Dataset	C/Cl (M.V.)	C/randAA (M.V.)
(a)	Cifar-10	*0.8691*	0.8602
(b)	Cifar-100	*0.6805*	0.6698

unreliable – and not the mark of exceptional robustness – as almost exclusively the effect of gradient obfuscation.

As we can see, the use of alternative aggregation strategies leads to minimal variations in the *clean* accuracy attained, whereas the corresponding *robust* accuracy sharply decreases – in the case of Cifar-10 even below random chance – as the attacks become increasingly effective (or is unreliable, as it is the case for *majority voting*). Such results strongly corroborate the use of the *normalised doubly-exponential logit product* proposed as the aggregation strategy in subsection 3.2.5 and prove its central role in the overall adversarial robustness and reliability of the method.

3.3.4 Limitations and open problems

In line with recent research aiming at the development of robust defences against multiple perturbations [147, 148], our method produces a decrease in *clean* accuracy *w.r.t.* the original model on which it is built upon – especially in *scenario (c)* as the complexity of the classification task increases. This phenomenon is partially dependent on the choice of a VAE as the generative purification model, a requirement for the fairest evaluation possible in terms of robustness.

Yet, the issue remains open: is it possible to devise a Carso-like architecture capable of the same – if not better – robust behaviour, which is also competitively accurate on clean inputs? Potential avenues for future research may involve the development of Carso-like architectures in which representation-conditional data generation is obtained by means of diffusion or score-based models. Alternatively, incremental developments aimed at improving the cross-talk between the purifier and the final classifier may be pursued.

Additionally, the scalability of Carso could be strongly improved by determining whether the internal representation used in conditional data generation may be restricted to a smaller subset of layers, while still maintaining the general robustness of the method.

Finally, a thorough investigation of the *normalised doubly-exponential logit product* aggregation strategy needs to be undertaken in order to shed some light on the specific mechanisms that lead to the much improved defensive capabilities of the system.

3.4 Conclusion

In this chapter, we presented a novel adversarial defence mechanism tightly integrating input purification, and classification by an adversarially-trained model – in the form of representation-conditional data purification, followed by a specific logit aggregation. Our method is able to improve upon the current *state-of-the-art* in Cifar-10, Cifar-100, and TinyImageNet ℓ_∞ robust classification, *w.r.t.* both *adversarial training* and *purification* approaches alone.

Such results suggest a new synergistic strategy to achieve adversarial robustness in visual tasks and motivate future research on the application of the same design principles to different models and types of data.

Chapter 4

Driving enhanced exciton transfer by automatic differentiation

4.1 Introduction

Complex systems are composed of several interacting components with a behaviour that is not immediately predictable from the characteristics of its parts, and often exhibits emergent phenomena. They include, among others, social [149, 150] and economic structures [151, 152] as well as biological complexes [153, 154], and are currently the subject of intense study: either to understand the emergence of new phenomena or as tools to analyse real-world complex scenarios. Complex systems are nowadays receiving much attention and study also in the quantum context [14]. In fact, it is now possible to engineer and manipulate multipartite quantum systems in many experimental settings, reaching sizes where their complexity becomes significant and practically relevant [15, 155]. Complexity has been thus identified as a precious resource for several quantum tasks, ranging from communication to computation and metrological ones [156–159].

One paradigmatic example of complex quantum dynamics is that of continuous-time quantum walks (CTQWs) [17], where a single quantum walker flows throughout a physical system, represented by a complex network. CTQWs have been used to model the excitation transfer in bio-molecule complexes [160–165], offering insight into the intricate interplay between environmental effects, the network structure, and surviving quantum features in determining the performance of the transfer process. In particular, because of the dissipative role of the environment, fast excitation transfer is needed to maximise its efficiency.

In this chapter, we investigate excitation transfer in complex networks with the purpose of optimising transfer speed [166–168]. Inspired by the excitation transfer process in a photovoltaic system, our goal is to describe, using a simple model, the absorption of a photon, its conversion into an excitation, and subsequent transfer along a network. We focus in particular on the case where two of the system components are subject to an external driving, and make use of algorithmic differentiation and gradient-based optimisation techniques to find the optimal driving.

We find that such a relatively simple degree of control can greatly enhance the efficiency of the process. We also find that such a novel approach can be coupled with previously studied methods [82, 169] to yield better results compared to already optimised protocols. Furthermore, the inclusion of an absorption mechanism (the antenna) allows us to have a more general picture compared to the scenario typically considered in the literature [169].

The engineering of protocols to enhance the transport efficiency in quantum networks can be challenging due to the complex nature of the system. Moreover, while other control strategies, such as changing the environmental dissipation or engineering the network couplings, have been proven to be effective, their physical implementation would be far from trivial or even unfeasible. On the other hand, enacting a local control only on two of the system nodes would prove much more feasible at the experimental level. Additionally, as we are implementing control on a fixed number of systems, the number of parameters to optimise is constant regardless of the size of the network, while the complexity of alternative optimisation strategies may scale unfavourably with the network size.

This establishes a viable pathway to optimisation with the feature of being more computationally scalable than other strategies used in the past [170–173], making it a valuable tool for future optimisation procedures in similar settings.

4.2 General settings and methods

We aim to describe an energy-transfer process that involves some radiation incident on an antenna absorbing the incoming energy and converts it into an excitation transmitted across a network [174]. The N^{th} site of the latter is assumed to be connected to a *sink-like* system, where the excitation is stored. In modelling such elements, we must strike a delicate balance between accuracy and simplicity to ensure the possibility of grasping an intuition of the physics underpinning the overall process.

A sketch of the arrangement we consider, where both the antenna and the sink are modelled as two-level systems, is shown in Figure 4.1. The incoming radiation is considered as a single mode, represented by a harmonic oscillator initially in its first excited state. This oscillator interacts through a hopping interaction with the antenna and excites it. The antenna is then connected to the first site of a network with N sites. For the network, we choose to work in the subspace of a single excitation, which significantly reduces the dimension of the Hilbert space (which is $N+1$, compared to 2^N, for a network comprised by N two-level systems). The choice of the single excitation subspace is customary in the CTQWs literature [17, 82, 160, 161, 163, 169], and justified whenever there is low rate of absorption, which is the case in light harvesting systems. We also include non-unitary terms in the network to account for possible local dephasing. Finally, the last site of the network is connected to a two level system through non-unitary dynamics. This serves as the sink because, in transport phenomena, the excitation is typically absorbed upon reaching its destination. Thus we model the interaction with the sink as an irreversible decay channel, meaning that once the excitation reaches the sink, it cannot return to the network.

The Bohr frequency of the antenna and the energy of the last site of the network will generally be time-dependent. The reason for maintaining such a time dependence is that, as we will discuss in detail, one possible way to increase energy transfer along the network is to introduce suitable time-dependent drivings.

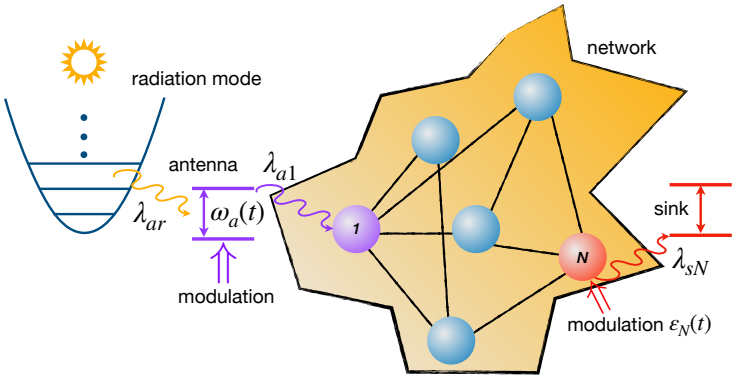

Figure 4.1: *Schematic of the model. One mode of the radiation interacts with the antenna (subscript 'a'), exciting its ground state. The excitation then jumps to the first site of the network and travels through it, up to the last site (N), which is connected to a sink (subscript 's').*

Accordingly, the model is described by a Lindblad Master Equation [84, 85] of the form:

$$\frac{d\hat{\rho}(t)}{dt} = -\frac{i}{\hbar}\left[\hat{H}, \hat{\rho}(t)\right] + L_N[\hat{\rho}(t)] + L_S[\hat{\rho}(t)],$$

where the Hamiltonian is a sum of several contributions:

$$\hat{H} = \hat{H}_R + \hat{H}_A(t) + \hat{H}_N(t) + \hat{H}_{AR} + \hat{H}_{A1}.$$

The Hamiltonian of the single radiation mode with frequency ω_r is $\hat{H}_R = \hbar\omega_r \hat{a}^\dagger \hat{a}$. The Hamiltonian of the antenna is that of a simple two-level system, i.e.

$$\hat{H}_A(t) = \hbar\omega_a(t)\left(|e_a\rangle\langle e_a| - |g_a\rangle\langle g_a|\right),$$

where $|e_a\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|g_a\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ are the excited and ground state.

We take the frequency $\omega_a(t)$ to be the time-dependent function

$$\omega_a(t) = \omega_a + \sum_{i=1}^{R} A_i \sin(\nu_i\, t + \phi_i)$$

with ω_a the Bohr frequency of the $|e_a\rangle \leftrightarrow |g_a\rangle$ transition in the absence of any modulation, and A_i, ν_i, ϕ_i real parameters that enter the harmonic-based decomposition of the modulation strategy that we will aim at optimising.

The Hamiltonian of the network is given by

$$\hat{H}_N = \sum_{i=1}^{N-1} \varepsilon_i |i\rangle\langle i| + \varepsilon_N(t)|N\rangle\langle N| + \sum_{i\neq j=1}^{N} V_{ij}\left(|i\rangle\langle j| + |j\rangle\langle i|\right),$$

where ε_i are the energies of the sites and V_{ij} the couplings of the interaction between sites i and j. The last site N, which is connected to the sink, has an energy which can be time-dependent to allow the introduction of a driving, which similarly to Eq. Figure 4.2 takes the form

$$\varepsilon_N(t) = \varepsilon_N + \sum_{i=1}^{R} B_i \sin(\mu_i t + \theta_i),$$

with ε_N, B_i, μ_i, and θ_i real parameters. Note that the Hilbert space of the network, in addition to the N states representing its sites, contains also a vacuum state $|0\rangle$ with energy $\varepsilon_0 = 0$ that does not interact with any other site of the network. This site essentially represents the absence of excitations in the network, and it only interacts with the antenna. The reason for its introduction is that, contrary to what is typically done where one assumes that the excitation is in the first site of the network at time $t = 0$, here we are also modelling the absorption of the excitation. Hence, we need a state representing the absence of excitation in the network, which we model by adding an additional site that does not interact with the others.

The radiation-antenna and antenna-network interactions are described by

$$\hat{H}_{AR} = \lambda_{ar}|e_a\rangle\langle g_a| \otimes \hat{a} + h.c. \quad \text{and} \quad \hat{H}_{A1} = \lambda_{a1}|e_a, 0\rangle\langle g_a, 1| + h.c.,$$

respectively. Such hopping Hamiltonians rule the coherent transfer of excitations between the radiation mode and the antenna (at rate λ_{ar}), and between the antenna and the first site of the network (at rate λ_{a1}).

Going back to Eq.displaymathFigure 4.2, we introduce the incoherent term

$$L_S[\hat{\rho}] = \lambda_{sN}\left(P_{g_s,N}|e_s, 0\rangle\langle e_s, 0| - \frac{1}{2}\{|g_s, N\rangle\langle g_s, N|, \hat{\rho}\}\right),$$

with $P_{g_s,N} = \langle g_s, N|\hat{\rho}|g_s, N\rangle$ the population of state $|g_s, N\rangle$ of the sink-site N compound. This describes the one-way mechanism through which an excitation populating site N is transferred to the sink at a rate λ_{sN}. We also have the local dephasing mechanism

$$L_N[\hat{\rho}] = \lambda_N\left(\sum_{j=1}^{N}|j\rangle\langle j|\hat{\rho}|j\rangle\langle j| - \hat{\rho}\right)$$

with $|j\rangle$ representing the state where site j of the network is populated, and λ_N the dephasing rate, assumed for simplicity to be the same for all sites.

In the next section, we will study several ways to improve excitation transfer from the radiation to the sink. In all the case-studies, we will take as initial state

$$\hat{\rho}(0) = |1_r\rangle\langle 1_r| \otimes |g_a\rangle\langle g_a| \otimes |0\rangle\langle 0| \otimes |g_s\rangle\langle g_s|,$$

i.e. the oscillator is in its first excited state and all other elements of the model are in their ground state.

We evolve this state, using a fourth-order Runge-Kutta method (see [175], p. 215), according to Eq. Figure 4.2 for different networks. Then, we optimise different sets of parameters in order to maximise the probability that the excitation reaches the sink in the shortest time.

4.3 Analysis and results

The model introduced in the previous section depends on many potentially tunable parameters. In what follows, we will optimise subsets of these parameters (keeping the remaining fixed), to maximise the time-integrated probability that an excitation reaches the sink within a given time.

We focus on three choices for the parameters to be optimised. In the first case, we optimise the drivings $\omega_a(t)$ and $\varepsilon_N(t)$; in the second case, the couplings λ_{ar} and λ_{a1}; in the last case, the energies of the network ε_i. In each of those cases, we compare the setting in which selected parameters are optimised, with the corresponding unoptimised scenario.

Parameter optimisation is carried out using gradient-based methods, relying on automatic differentiation [18] provided by the Python library *PyTorch* [127] and using the *Adam* optimiser [176]. Specifically, the learning rate of the optimiser has been chosen in a case-dependent fashion, according to the empirical complexity of the loss landscape [177].

In detail, the parameters are optimised in such a way to maximise the time-integrated probability for the excitation to reach the sink, i.e.

$$I_P(T_L) := \int_0^{T_L} p_{sink}(t)\, dt,$$

where $p_{sink}(t) = \mathrm{Tr}[|e_s\rangle\langle e_s|\hat{\rho}(t)]$ with $\hat{\rho}(t)$ the total density matrix at time t, $|e_s\rangle\langle e_s|$ the projector on the excited state of the sink and T_L the time of evolution used as part of the optimisation of the parameters. The rationale for such a choice is that by maximising $I_P(T_L)$ instead of the final probability $p_{sink}(T_L)$, we can identify a set of parameters that not only maximises $p_{sink}(T_L)$, but also favours solutions where $p_{sink}(t)$ increases significantly at earlier times, i.e., those where the excitation reaches the sink more quickly.

We perform our analyses on three different types of network: one describing a nearest neighbour (NN) interaction, a star network (SN) where one of the sites is connected to all the others, and one modelling the Fenna–Matthews–Olson (FMO) complex [16, 178].

4.3.1 Nearest neighbour network

We start by considering the case of a NN network, which has the Hamiltonian:

$$H_N = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0.5 & 1 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0.5 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & 0.5 & 1 \\ 0 & 0 & 0 & \dots & 0 & 1 & 0.5 \end{pmatrix}.$$

The remaining parameters used in the simulation are summarised in Table 4.1. The coupling strengths λ_{ar} and λ_{a1} were set equal to ω_a to match the network's energy scale. Network dimensions were chosen as $N = 4, 8$ to balance complexity and computational feasibility. The dephasing strength was set to $\lambda_N = 0.1\omega_a$ to remain in the weak coupling

$\omega_a T_L$	$\omega_a T$	ω_r/ω_a	N	λ_{ar}/ω_a	λ_{a1}/ω_a	λ_N/ω_a	λ_{sN}/ω_a
30	1200	{0.264, 15}	{4, 8}	1	1	0.1	0.05

Table 4.1: *Values of model parameters used to run the simulations; times are in units of ω_a^{-1} and frequencies in units of ω_a. T_L is the evolution time used within the optimisation process; after optimisation, we test the performance of the learned system by evolving it further up to time T; ω_r is the frequency of the radiation, ω_a the Bohr frequency of the antenna (when there is no driving), N the number of sites in the network, λ_{ar} and λ_{a1} the values of the couplings (when not optimised), λ_N the value of the dephasing constant and λ_{sN} the value of the sink constant.*

regime, ensuring the validity of the Lindblad (Markovian) model. The sink interaction strength $\lambda_{sN} = 0.05\omega_a$ was chosen to keep its influence on excitation transfer minimal. The value $\omega_r = 0.264\omega_a$ was optimized for maximal exciton transfer while keeping network parameters fixed (resonant condition). To test our strategies, we also considered $\omega_r = 15\omega_a$, significantly detuned from resonance. We set $\hbar = 1$, and therefore the parameters with the dimension of an energy are expressed in units of ω_a; times are expressed in units of ω_a^{-1}. We choose as the initial state the one specified in Eq. Figure 4.2.

All simulations are run for a total time of $T = 1200$, which is significantly longer than the evolution used within the optimisation process ($T_L = 30$). In fact, we verified that a further increase in T_L does not result in any significant improvement in the integral of sink probability. The coupling to the sink λ_{sN} is taken to be an order of magnitude smaller than all other couplings to limit the resulting Zeno effect, which would significantly influence the flow of excitation along the network.

We start by considering a network with $N = 4$ sites and an incident radiation with a frequency of $\omega_r = 0.264$. This baseline value is chosen as it maximises the value of $I_P(T_L)$ in Eq. section 4.3 for the parameters shown in Table 4.1.

Figure 4.2(A) shows the probability of the excitation reaching the sink as a function of time, comparing the case without any optimisation (grey curve) to the cases where the driving parameters (red curve), the couplings (yellow curve) and the network energies (blue curve) are optimised. We see that none of the three optimisation strategies is particularly effective compared to the baseline, because $\omega_r = 0.264$ is already optimal. One might argue that, by considering more complex drivings where more terms ($R > 1$) are accounted for in the series in Eqs. (4.2) and (4.2) the performance could potentially improve.

We found out that this is not the case and that, interestingly, increasing the number of terms in the drivings up to $R = 7$ (corresponding to a total of 21 real parameters to learn for each driving term) the network after optimisation performs worse than the original. This behaviour is probably due to the significant increase of the complexity of the solutions landscape and therefore to a more difficult optimisation problem.

We now turn to a more interesting case by selecting a different frequency for the incident radiation, $\omega_r = 15$. This particular value is chosen to be significantly different from the

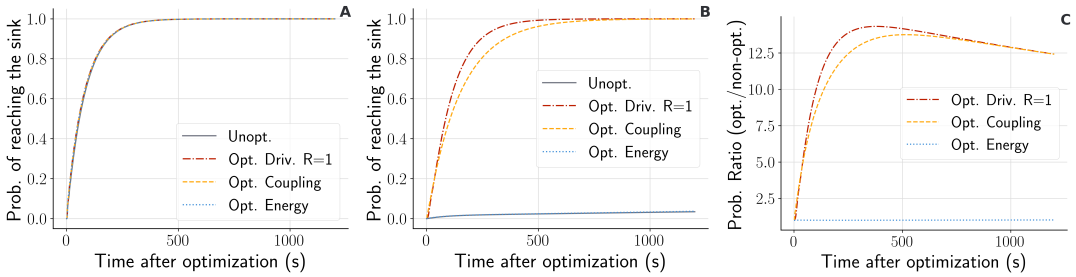

Figure 4.2: *Comparison among unoptimised network, optimised driving (with $R = 1$ in Eqs.displaymathFigure 4.2 and Figure 4.2), optimised coupling, and optimised energies for a NN network of $N = 4$ sites. (A) Probability of reaching the sink when the frequency of the mode is $\omega_r = 0.264$ (resonant case): the improvement w.r.t. the unoptimised network is minimal with all the three methods. (B) Probability of reaching the sink when the frequency of the mode is $\omega_r = 15$ (off-resonant case): the improvement is relevant when optimising the couplings or the drivings. (C) Ratio between the probability of reaching the sink with the three optimisation strategies and the probability of reaching the sink without optimisation, in the off-resonant case ($\omega_r = 15$).*

baseline at $\omega_r = 0.264$, at which the network is known to perform well. By using an incident photon with significantly higher energy, we expect it to be far off-resonance, making the optimisation of tunable parameters, or the introduction of external driving, way more effective. Figure 4.2(B) shows that this is the case. In fact, the probability of reaching the sink is increased by more than an order of magnitude when the drivings or the couplings are learnt (see Figure 4.2(C)). On the other hand, optimising the network energies is not so effective, as shown by the dotted blue line in Figure 4.2(C).

To further test the strategy based on introducing and optimising drivings, we studied how it changes by increasing the number of terms in the drivings $R = 1, 2, 7$ or by considering network of different sizes $N = 4, 6, 8$. As one can see in Figure B.1(A) of Appendix 1, no sensible improvement is reached by increasing R. Also, as shown in Figure B.1(B) of Appendix 1, the conclusions we draw for the case $N = 4$ are qualitatively confirmed for larger networks, showing that driving optimisation is effective over larger networks.

Moreover, we tested all three methods under increasing levels of noise. In Figure B.1(C) of Appendix 1, we repeated the same analysis as in Figure 4.2(B), but with $\lambda_N = 1$ (instead of 0.1), finding no significant differences. We then focused on optimising only the driving, for increasing levels of noise ($\lambda_N = 0.1, 1, 100$). The results presented in Figure 4.3 demonstrate that driving optimisation remains highly effective. They also reveal an interesting phenomenon: increasing noise levels in an unoptimised system may lead to a higher integrated sink probability. This appears to be an instance of noise-assisted transport [179].

4.3.2 Star Network

In this section we consider a different type of network, called *star network* (SN), where one of the sites (the second, in our case) is connected to all the others. The goal of such an analysis is to check whether the conclusions reached in the previous scenario also

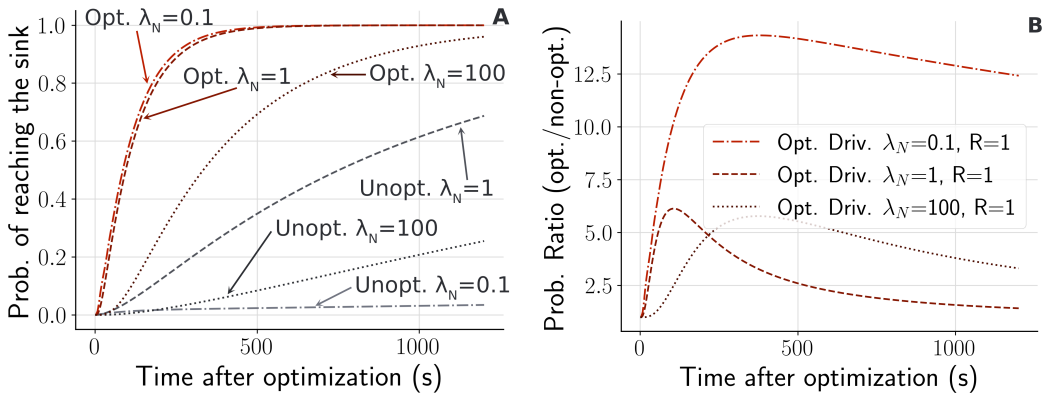

Figure 4.3: *(A) Probability of reaching the sink for unoptimised network (`Unopt.`) and one where the drivings where optimised (`Opt.`) ($\omega_r = 15$ and $R = 1$ in Eqs.displaymathFigure 4.2 and Figure 4.2). The simulation were run for different values of the dephasing parameter $\lambda_N = 0.1, 1, 100$. (B) Ratio between the probability of reaching the sink with the driving optimisation and the probability of reaching the sink without optimisation for the three values of λ_N (being $\omega_r = 15$ and $R = 1$ kept constant).*

extend to other network setups.

The Hamiltonian of the star network is:

$$H_{star} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.5 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0.5 \end{pmatrix}.$$

Figure 4.4(A) illustrates that, unlike in the NN network, more than one driving term is required for optimal results, i.e. $R > 1$. However, the number of terms remains relatively small, as we observe a significant improvement with just $R = 2$.

In Figure 4.4(B) and (C) we compare the performance of the unoptimised network against those reached by optimising energies, coupling, and drivings (with $R = 2$). The results confirm the conclusions obtained for the NN network: optimising drivings or couplings is a better strategy than optimising the energies of the network.

4.3.3 FMO

In this section, we examine our final and more structured network, based on the Hamiltonian of the FMO complex [16, 178]. While a proper description of the FMO dynamics would require, among other things, properly accounting for non-Markovian effects [180, 181], we simply use it as an example of a more realistic network

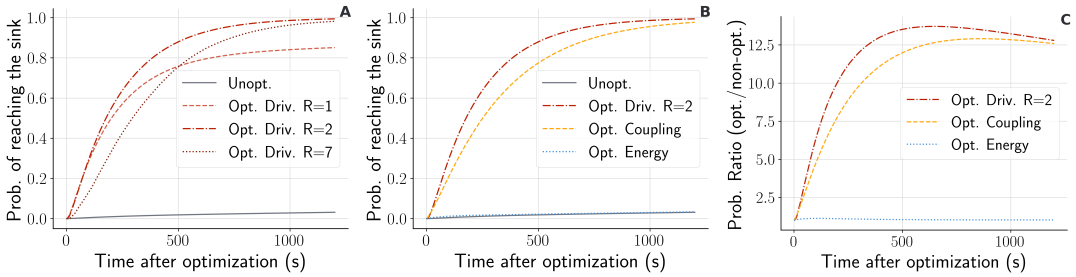

Figure 4.4: *Comparison among unoptimised network, optimised driving, optimised coupling, and optimised energies for a SN network of $N = 8$ sites in the off-resonant case $\omega_r = 15$. (A) Probability of reaching the sink for the unoptimised network (grey continuous line) and when increasingly expressive drivings ($R = 1, 2, 7$, see Eqs.displaymathFigure 4.2 and Figure 4.2) are introduced and optimised (red lines). (B) Probability of reaching the sink for an unoptimised network, optimised driving with $R = 2$, optimised coupling, and optimised energies. The line corresponding to an optimised driving with $R = 2$ is the same as from (A). (C) Ratio between the probability of reaching the sink with the three optimisation strategies and the probability of reaching the sink without optimisation.*

The network Hamiltonian is that found in [82], where the energies are already optimised (see Eq. (A1) of Appendix 1 in [82]):

$$H_N = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 65.7 & -104.1 & 5.1 & -4.3 & 4.7 & -15.1 & -7.8 \\ 0 & -104.1 & -11.1 & 32.6 & 7.1 & 5.4 & 8.3 & 0.8 \\ 0 & 5.1 & 32.6 & -56.1 & -46.8 & 1.0 & -8.1 & 5.1 \\ 0 & -4.3 & 7.1 & -46.8 & -36.2 & -70.7 & -14.7 & -61.5 \\ 0 & 4.7 & 5.4 & 1.0 & -70.7 & -30.6 & 89.7 & -2.5 \\ 0 & -15.1 & 8.3 & -8.1 & -14.7 & 89.7 & 55.7 & 32.7 \\ 0 & -7.8 & 0.8 & 5.1 & -61.5 & -2.5 & 32.7 & 4.2 \end{pmatrix}.$$

Most of the conclusions drawn for the previous networks also apply to the FMO. For instance, when we compare the performance of the unoptimised system to cases where we optimised energy, coupling, or driving at the frequency that already maximises the probability of reaching the sink (in this case, $\omega_r = 0.242$), we observe no significant improvements (data not shown).

Additionally, Figure 4.5(A) shows that the driving with one term ($R = 1$) is not very effective whereas just adding one term ($R = 2$) greatly improves the performance.

In Figure 4.5(B) and (C) we compare the performance of the unoptimised network against those reached by optimising energies, couplings and drivings (with $R = 2$). The improvement is even more remarkable when one considers that the Hamiltonian of the network H_N in Eq. subsection 4.3.3 was explicitly optimised along the diagonal to maximise exciton transfer in [82].

Also, from the same figures we see that couplings optimisation, differently from the other networks, performs significantly worse than driving optimisation with $R = 2$. This

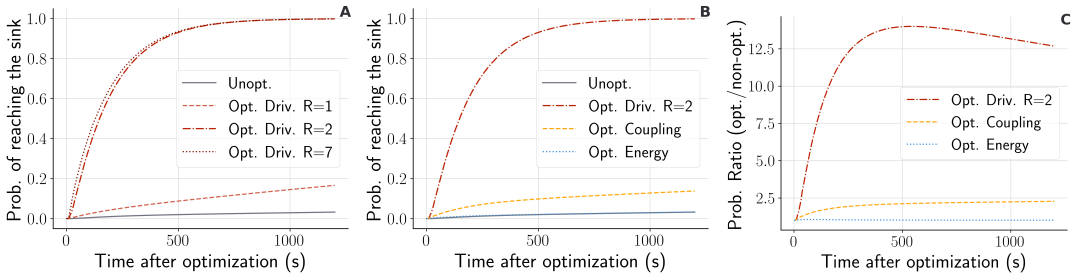

Figure 4.5: *Comparison among unoptimised network, optimised driving, optimised coupling, and optimised energies for the FMO network in Eq. subsection 4.3.3 in the off-resonant case $\omega_r = 15$. (A) Probability of reaching the sink for the unoptimised network (grey continuos line) and when increasingly expressive drivings ($R = 1, 2, 7$, see in Eqs.displaymathFigure 4.2 and Figure 4.2) are introduced and optimised (red lines). (B) Probability of reaching the sink for an unoptimised network, optimised driving with $R = 2$, optimised coupling, and optimised energies. The line corresponding to an optimised driving with $R = 2$ is the same as from (A). (C) Ratio between the probability of reaching the sink with the three optimisation strategies and the probability of reaching the sink without optimisation.*

analysis suggests that the optimisation of the drivings can be more effective than that of the couplings, especially when complex networks are considered.

4.4 Discussion

In this chapter, we investigated the excitation-transfer process across different types of network. In particular we explicitly modelled the radiation-antenna interaction at the absorption stage, and focused on the possible effect of driving on both the antenna and the target site. Using state-of-the-art automatic differentiation tools and gradient-based optimisation algorithms, we determined the parameters of the system that maximise the efficiency of the transport, as quantified by the time-integrated probability of reaching the sink.

Our analysis shows that optimising the coherent couplings between, respectively, the antenna and the incoming radiation – and the antenna and the first site of the network – is effective for simpler networks like NN and SN, but not for more complex networks such as the one associated with the FMO. On the other hand, introducing and optimising external drivings leads to a significant enhancement in excitation transfer across all network types and dimensions. Importantly, a small number of driving terms is sufficient to achieve near-optimal efficiency in all considered scenarios: indeed the increase in excitation transfer efficiency in the off-resonant case amounts to more than an order of magnitude compared to the undriven case. This could be relevant in the context of energy harvesting, where a tunable device, capable of efficiently absorbing excitations in a range of several frequencies, is desirable. A deeper understanding of the physical reasons for which learning the couplings is less efficient than introducing and learning optimal drivings will be the focus of future research. Moreover, we expect the driving of the local energies to be more experimentally feasible (e.g. by the modulation of an external field) than acting on the couplings, which might be a more challenging task,

also depending on the specific experimental platform one has in mind – which is not available yet, and we hope this work will help to establish.

Our analysis concludes that significant enhancement in energy transfer within complex networks can be achieved by applying simple driving modulation at both the start and end of the network. This approach proves especially effective under strong off-resonant conditions between the incoming light and the system, and the practicality of implementing such drivings makes our findings valuable and broadly applicable for optimising energy transfer processes. Our study paves the way for further research in this direction, particularly to assess the robustness of our strategy across different regimes. A relevant direction in this respect is the incorporation of memory effects in the dynamics, due for example to comparable time scales between system and environmental evolutions – as commonly encountered in the energy transfer in photosynthetic processes. Moreover, a complete characterization of the practical utility of our strategy will require evaluating the energetic cost of the driving that enhances the excitation transfer, compared to the energetic cost of competing strategies.

Chapter 5

Towards very-few-shot structured supervised multi-task learning

5.1 Introduction

Among the very central pursuits of Artificial Intelligence there is the attempt to replicate behaviours traditionally associated with higher human cognitive functions, such as the ability to extract meaningful features from sensory inputs and use them to solve a given task from examples or by *trial and error*. In such regard, the achievements of deep learning have been astounding — with *state of the art* models often being capable of surpassing human ability in visual and audio-centric tasks, as well as in the solution of intricate or exploration-heavy tabletop and video games, and able to drive complex dexterous actuation of robots in rugged unfriendly environments.

Nonetheless, a large unfilled gap still remains between humans and deep learning models — even those that can claim indisputably better performance against their human counterparts — *w.r.t.* how tasks are actually learnt and evaluated. Indeed, deep learning systems are almost always purpose-built with a specific goal or task in mind — although potentially broad — and trained or validated with data curated for such purpose. Should requirements or tasks change, training would need to be repeated. On the other hand, human beings usually learn many tasks at once, and many more along their lifetime — without forgetting those previously mastered — and they do so from a stream of percepts mostly composed of information not strictly relevant to each specific task, often without direct supervision [23]. Even more strikingly, should a new task require to be learned or objectives slightly change, humans are innately able to rapidly recover some structure from previous experiences or knowledge, swiftly adapting to the unknown.

Attempts to close such gap, allowing trained machine learning models to integrate new examples towards the adaptation to a new task do exist and broadly belong to three categories: (1) *transfer learning* [19] approaches typically re-use early layers of a neural network model trained on given tasks, while re-initialising and re-training (the) last layer(s) on data pertaining to the new task(s); (2) *fine-tuning* [20–22] approaches instead simply carry on the training of the model with data pertaining to the new task(s), eventually after an adequate adjustments to the loss function or hyperparameters;

(3) *continual/lifelong learning* [23, 182] techniques try to devise algorithms that allow the learning of several tasks in a sequence, balancing the effect of adaptation to a new task with retention of previously-learned ones. However, despite some success, all those methods require a still significant amount of curated data, and none is able to actively promote the same kind of structure consolidation and later reuse that makes active learning so effective in humans. Frontier large language models may represent the closest incarnation of such idea of *in-context learning* [183] — still being greatly reliant on sheer pre-training efforts in terms of time and compute, and on fine-tuning for knowledge update.

In this chapter, we develop a novel framework for supervised multi-task representation learning that allows fast *very-few-shot* test-time adaptation to new unforeseen tasks that bear some structural similarity to those seen during training. Our approach is based on a nonlinear generalisation of Koopman operator learning [26–28] and is geared towards modelling the input/output mapping associated with a given task as a specific discrete-time dynamics of a dynamical system. Training occurs driven by a multi-objective gradient-based optimisation scheme, whereas test-time adaptation only requires the fitting in closed form of one affine operator per additional task.

Preliminary tests on challenging geometrically-grounded visual tasks with a clear analytical optimal representation and solution show promising results in terms of adaptation accuracy, speed, and data efficiency. Future developments towards imitation learning in more complex settings are finally outlined.

5.2 The framework

In this section we introduce our novel framework for supervised multi-task learning and fast test-time adaptation to new tasks. Although originally geared towards *next-state prediction* for discrete-time dynamical systems, its application to arbitrary setups with generally weak assumptions would be easily achieved.

5.2.1 General setting

Let us consider a typical setting in supervised multi-task learning, where several different *tasks* τ_i are required to be *learned*, from examples, by the same model. In particular, we have access to training sets composed of inputs $\boldsymbol{x}_{ij} \in \mathbb{R}^n$ and outputs $\boldsymbol{x}'_{ij} \in \mathbb{R}^n$ such that:

$$\boldsymbol{x}'_{ij} = \tau_i(\boldsymbol{x}_{ij});\ \forall i \in \mathcal{I},\ j \in \mathcal{J}_i$$

with $\mathcal{I}$, $\mathcal{J}_i$ appropriate sets of indices. In the specific case, $\mathcal{I}$ would identify indices associated with the different tasks, whereas $\mathcal{J}_i$ is associated with the plurality of examples in the respective training set (thus the dependence on index i).

We remark here that the same vector space $\mathbb{R}^n$ can be assumed to contain both inputs and outputs across tasks without loss of generality — as it is always possible, *e.g.*, to pad all vectors to the maximum size among them.

5.2.2 Model structure

In such setting, we can formally consider each task $\tau_i : \mathbb{R}^n \to \mathbb{R}^n$ as a different transition operator acting on the state ($\boldsymbol{x}_{ij}$) of the same discrete-time dynamical system — where it simply induces a different dynamical evolution (to state $\boldsymbol{x}'_{ij}$) — akin to the formalism introduced in subsection 2.4.1 for a single task.

Thus, we propose to estimate $\boldsymbol{x}'_{ij}$ as

$$\boldsymbol{x}'_{ij} \approx \hat{\boldsymbol{x}}'_{ij} \coloneqq \beta_{\boldsymbol{\theta}_\beta}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij}) + T_i \varphi_{\boldsymbol{\theta}_\varphi}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij})))$$

where $\alpha_{\boldsymbol{\theta}_\alpha} : \mathbb{R}^n \to \mathbb{R}^{d_e}$, $\beta_{\boldsymbol{\theta}_\beta} : \mathbb{R}^{d_e} \to \mathbb{R}^n$, and $\varphi_{\boldsymbol{\theta}_\varphi} : \mathbb{R}^{d_e} \to \mathbb{R}^{d_d}$, are task-independent parametric sub-models (*e.g.* artificial neural networks) with learnable parameters $\boldsymbol{\theta}_\alpha$, $\boldsymbol{\theta}_\beta$, and $\boldsymbol{\theta}_\varphi$ respectively. $T_i \in \mathbb{R}^{d_d \times d_e}$ are instead task-dependent affine operators. d_e, $d_d \in \mathbb{N}$ are treated as hyperparameters.

In such context, model $\alpha_{\boldsymbol{\theta}_\alpha}$ acts on system state as a (potentially nonlinear) embedding-generating function, whereas $\beta_{\boldsymbol{\theta}_\beta}$ represents the reciprocal *de-embedding* function. The dynamics can then be described — within the embedding space — as

$$\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}'_{ij}) = \alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij}) + T_i \varphi_{\boldsymbol{\theta}_\varphi}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij}))$$

or, with a rearrangement of terms:

$$\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}'_{ij}) - \alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij}) = T_i \varphi_{\boldsymbol{\theta}_\varphi}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij}))$$

whose structure bears some strong similarity with *vanilla Koopman operator learning*. Two crucial differences, nonetheless, are also present. In particular, they are described as follows.

- The task-dependent affine operator (T_i) does not act directly on an embedding of the state before the transition occurs ($\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij})$), but on a further **transformation of it** ($\varphi_{\boldsymbol{\theta}_\varphi}$).
- The result of the action of such operator is not the corresponding embedding for system state after the transition ($\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}'_{ij})$), or a function thereof. It is instead the **difference** of those embeddings across the transition ($\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}'_{ij}) - \alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij})$).

In our intentions, such departure from the standard Koopman operator learning setup is not arbitrary. Indeed, we want — at least in principle — to endow each specific term with an equally specific *role* in modelling *transitions/tasks* and the corresponding states before and after they occur (*i.e.*, the corresponding inputs and outputs).

In particular, the role of $\alpha_{\boldsymbol{\theta}_\alpha}$ would be similar to that of the Koopman embedding operator — in that it should be able to equip system states with a suitable metric, capable of capturing a *global* geometrical structure of system states, independent of specific tasks, over which (at least *typically*) all induced transitions are described with minimal error. In such sense, with $\varphi_{\boldsymbol{\theta}_\varphi}$ being the identity and an adequate choice of dimensions, the method would not be much different or more expressive than *vanilla Koopman*. On

the other hand, the effect of $\varphi_{\boldsymbol{\theta}_\varphi}$ would indeed describe a *nonlinear modulation* of such *typical dynamics*, capturing *local* state-dependent deviations from it. A more formal justification for the need of such nonlinear modulation term is provided in section C.1, where a crucial limitation of invertible linear latent dynamical evolution is uncovered in a simple multi-task scenario.

5.2.3 Model training

Model training happens under the drive of loss minimisation by gradient-based methods, as customary with most contemporary deep learning models. In principle, one could simply fit all learnable parameters with an *end-to-end* reconstruction loss of the type $\mathcal{L} \coloneqq \left\langle \mathcal{D}(\hat{\boldsymbol{x}'}_{ij}; \boldsymbol{x}'_{ij}) \right\rangle_{i,j}$ where $\mathcal{D}$ denotes any distance or divergence between estimated and true target states, and $\langle\cdot\rangle_{i,j}$ denotes the averaging over all available training examples. However, in order to stay as close as possible to the original Koopman operator learning theory — thus allowing to eventually benefit from more refined and advantageous derived fitting procedures, such as that described by [28] — we decompose the *end-to-end* reconstruction objective into two distinct sub-objectives, to be optimised in a multi-objective fashion as the linear combination of the two. Specifically, we employ a loss of the type:

$$\mathcal{L} \coloneqq c_{\text{emb}}\, \mathcal{L}_{\text{emb}} + c_{\text{dyn}}\, \mathcal{L}_{\text{dyn}}$$

where

$$\mathcal{L}_{\text{emb}} \coloneqq {}^1\!/_2\, \left\langle \mathcal{D}(\beta_{\boldsymbol{\theta}_\beta}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij})), \boldsymbol{x}_{ij}) \right\rangle_{ij} + {}^1\!/_2\, \left\langle \mathcal{D}(\beta_{\boldsymbol{\theta}_\beta}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}'_{ij})), \boldsymbol{x}'_{ij}) \right\rangle_{ij}$$

and

$$\mathcal{L}_{\text{dyn}} \coloneqq \left\langle \frac{\|\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}'_{ij}) - \alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij}) - T_i \varphi_{\boldsymbol{\theta}_\varphi}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij}))\|_2}{\min\left(\|\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}'_{ij}) - \alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij})\|_2,\ \varepsilon\right)} \right\rangle_{ij}$$

with $\varepsilon > 0$ in $\mathcal{L}_{\text{dyn}}$ being a numerical stability constant to avoid fraction blow-up.

Here, the role of the $\mathcal{L}_{\text{emb}}$ *inversion loss* is that of preventing representation collapse and promoting the learning of an invertible embedding sub-model amenable to the description of the many dynamical evolutions induced on the system state by the different tasks. On the other hand, the $\mathcal{L}_{\text{dyn}}$ loss promotes the joint learning of each task-specific linear evolution operator, together with the — still global — state-dependent nonlinear modulation term. The norm-term at the denominator of $\mathcal{L}_{\text{dyn}}$ allows metric-aware minimisation in the embedding space, while the additional freedom on multiplicative coefficients c_{emb} and c_{dyn} enables a more controllable training.

Regularisation and *generalisation-promoting* interventions

As with more traditional machine learning models, also our framework might be susceptible to training-time overfitting, and as such it requires specific interventions to

ensure proper model regularisation and foster generalisation. Such aspect is clearly not specific to the multi-task setting, or to Koopman-like task modelling — on the other hand, countermeasures to prevent such pitfalls should better be. We recommend three different interventions in such case: one specifically designed to prevent the learning of brittle embeddings by our model (or general Koopman-style operator learning), the other two belonging to standard regularisation techniques. They are listed below.

Robust embedding inversion
Model training, as we outlined, does not directly rely on the *end-to-end* error on states to update its parameters. However, in the case of next-state prediction, having such error as close to zero as possible is all that matters to determine training success. To foster the development of an adequately-structured, yet spaced-out, representation in embedding space — and thus more robust to slight encoding and dynamics errors — $\mathcal{L}_{\text{emb}}$ can be amended by introducing a deliberate but bounded random perturbation after state encoding and before decoding. The resulting loss function would thus read as:

$$\mathcal{L}_{\text{emb}} \coloneqq 1/2 \ \left\langle \mathcal{D}(\beta_{\boldsymbol{\theta}_\beta}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij}) + \boldsymbol{\delta}_\epsilon), \boldsymbol{x}_{ij}) \right\rangle_{ij} + 1/2 \ \left\langle \mathcal{D}(\beta_{\boldsymbol{\theta}_\beta}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}'_{ij}) + \boldsymbol{\delta}_\epsilon{}'), \boldsymbol{x}'_{ij}) \right\rangle_{ij}$$

with $\boldsymbol{\delta}_\epsilon,\ \boldsymbol{\delta}_\epsilon{}' \sim \mathcal{N}(0, \epsilon)$. The value of ϵ should be treated as a hyperparameter. Additionally, as a side-effect, such *generalisation-enhancing* technique further prevents the risk of representation collapse.

Relaxed ϵ-insensitive embedding inversion
The appropriate tuning of coefficients c_{emb} and c_{dyn} controls the relative relevance of the respective loss terms. However, this exerts little control over the respective gradient flows and the local shape of the loss landscape. As such, it may happen that the resulting optimisation favours the minimisation of the $c_{\text{emb}}\mathcal{L}_{\text{emb}}$ term nonetheless, even well beyond a tolerable error. In turn, this may result in an overcommitment of model expressivity towards increasingly diminishing returns, and ultimately hinder test-time adaptability. Forcefully zeroing the loss when sufficiently close to zero may mitigate the issue. Traditionally, this could be done by further amending $\mathcal{L}_{\text{emb}}$ as $\tilde{\mathcal{L}}_{\text{emb}} = \max(\mathcal{L}_{\text{emb}} - \epsilon_{\text{ins}},\ 0)$ [184]. A continuous relaxation of such function can also be used, such as:

$$\tilde{\mathcal{L}}_{\text{emb}} = \text{Softplus}(\mathcal{L}_{\text{emb}} - \epsilon_{\text{ins}})$$

where $\text{Softplus}(z) \coloneqq \frac{1}{b}\ln(1 + e^{bz})$, with $\ln$ being the Natural Logarithm, e Euler's Number, and b a tunable smoothness parameters. Even though b could be treated as a hyperparameter, its tuning is not particularly fruitful in this setting and the usual value of $b = 1$ can be used. We explicitly remark here that, as provided by its formal description, such regularisation intervention is compatible with the generalisation-promoting technique introduced earlier.

Rank constraint on T_i
Finally — especially in the case when operators are chosen to be particularly high-dimensional — T_is could undesirably absorb part of the modelling role assigned in our intentions to $\alpha_{\boldsymbol{\theta}_\alpha}$ and $\varphi_{\boldsymbol{\theta}_\varphi}$. This could lead to a sub-optimal structure being learnt by

the latter, ultimately compromising test-time adaptation capabilities. In such sense, it is possible to constrain the *rank* of T_i to a fixed value $\mathbb{N} \ni r < \min(d_d,\ d_e)$. The most straightforward way to achieve such goal is to decompose T_i according to the matrix factorisation

$$\mathbb{R}^{d_d \times d_e} \ni \mathrm{T}_i = \mathrm{T}'_i \, \mathrm{T}''_i$$

where $\mathrm{T}'_i \in \mathbb{R}^{d_d \times r}$ and $\mathrm{T}''_i \in \mathbb{R}^{r \times d_e}$. Again, such intervention is in no way incompatible with the two described above.

We also provide here some heuristic suggestions for the choice of d_d and d_e, calling r the rank of T_is (regardless of whether it is a result of rank constraints or not) and k the total number of tasks (*i.e.* including both training-time tasks and the expectation of new tasks for which adaptation is required at test-time). In particular — unless justified by optimisation-related requirements — it is recommended not to exceed $d_e \leq d_d \leq k \times r$.

5.2.4 Test-time adaptation

The test-time scenario is almost identical to that of model training, with only two crucial differences. *I.e.*, the tasks for which examples are available are generally (but not strictly required to be) different from those considered during training — and the numerosity of such example pairs is strongly limited.

In such case, we propose to use the model obtained from the training phase, with learnable parameters $\boldsymbol{\theta}_\alpha$, $\boldsymbol{\theta}_\beta$, and $\boldsymbol{\theta}_\varphi$ kept frozen. For each new task for which adaptation is required (*i.e.* that has to be learned at test-time), new matrices T_i are allocated, and fitted **each**, **separately** and **analytically**, on the minimisation of mean square error loss in the embedding space, *i.e.*

$$\mathcal{L}^i_{\text{adapt}} \coloneqq \left\langle \left\| \alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}'_{ij}) - \alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij}) - T_i \varphi_{\boldsymbol{\theta}_\varphi}(\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x}_{ij})) \right\|_2^2 \right\rangle_j$$

Since the only term left learnable is the affine operator T_i, the test-time adaptation problem is reduced to the solution of system of linear equations. As far as the specific fitting procedure is concerned, the framework does not impose any strict requirement. For the rank-unconstrained setting, we suggest the use of *Ridge-regularised* least squares (RRLS) regression [185], whereas for the rank-constrained scenario with the specific factorisation described in section 5.2.3 a more advanced approach should be used in order to preserve the structure of T_i. In detail, *alternating least squares* [186, 187] (ALS) with a fixed number of iterations can be utilised for the fitting of each matrix factor (*i.e.* T'_i and T''_i), eventually employing as initial guess the SVD decomposition of a rank-unconstrained matrix fitted by RRLS on test data.

5.3 Preliminary experimental assessment

In this section we describe preliminary results from the testing of our framework, at the moment limited to a specific setup with challenging geometrical properties of the

inputs and outputs, yet simple and with clear analytical solutions for the structure of αs and T_is able to capture typical behaviour. The goal of our tests is threefold: (1) to ensure that our framework, as described in section 5.2, is able to learn — during training — the adequate structure we know to be required to solve the tasks and allow for optimal test-time generalisability; (2) to verify whether the test-time adapted model is able to correctly perform next-state prediction on previously unknown tasks, within a reasonable tolerance; (3) to investigate whether this remains true when local variations to typical dynamics are added, or when states are made to contain features irrelevant to dynamical evolution.

5.3.1 Setup

Our experimental setup is inspired by the idea of different tasks being *moves* performed on a chessboard by a chess-like piece. Such arrangement is intuitively easy to deal with, given the appropriate representation, as any chess-like move on the board can be optimally described as a specific translation on a flat Cartesian 2D grid. Our intuition, however, is so keen on this scenario also because it is probably not the first time we see a chessboard with moving pieces on it; nor it is the first time we model simple transformations within a system of coordinates. On the other hand, the success of a deep learning model — and its ability to generalise to new tasks — will inevitably depend on whether it is able to develop such *intuition* during its training alone, and *use* it appropriately when test-time adaptation occurs.

In the following subsection, we will build and describe our setup more detailedly — incrementally, adding one element of complexity at a time.

State representation

We begin by considering a finite square chessboard of size $L \times L$ such that exactly one piece can be found on it at any time. The piece is able to move according to the same rules as the *King* of chess.[1] In order to generate examples, an initial square is picked uniformly at random from the entire board, then a random move is sampled — still uniformly at random — among all those which are allowed from that initial position. Lastly, the initial/final position-pair is added to the example-set of a *task* corresponding to the given move direction (thus producing datasets for 8 different tasks in total).

The system state for both input and output, however, is provided to the model in the form of a flattened square 1-channel image, where all pixels bear the value of 0 (*i.e.* are black-coloured) except for the pixel corresponding to the position of the piece, which carries the (normalised) value of 1 (*i.e.* is white-coloured). The result is a vector of size L^2.

Such use of flattened images as states — instead of coordinates — makes the resulting problem geometrically much more challenging. Indeed, all of such states are mutually orthogonal in the standard basis and equidistant one another. Even though the described dynamics is simple in the aforementioned Cartesian representation, the model needs

[1] *I.e.* a move is allowed to any horizontally, vertically, or diagonally adjacent square — exactly once at a time.

to learn such representation (or, potentially, another equivalent to it) exclusively from examples.

Atypical dynamics: border behaviour

Assuming the model is able to perfectly recover a Cartesian representation of the chessboard, each of the 8 moves would be representable as a fixed translation vector, regardless of the specific *input/initial state* to which it is applied. We refer to it as *typical dynamical evolution*. To make such dynamics more challenging, we introduce some discontinuous task-state interdependence.

Specifically — and differently from traditional chessboard games — we allow moves to be performed also *against* board edges. In such case, we consider two different and mutually-exclusive possibilities: in the *sticky edges* case, the piece moved against an edge (regardless of the task or the specific edge) remains in its initial position; in the *loopy edges* case, the move is performed in periodic boundary conditions — giving rise to a toroidal topology.

Splitting focus

From talking in a noisy environment, to singing and playing an instrument at the same time, the ability to direct or split focus according to circumstances is crucial to many activities. In order to capture such peculiar aspect within our simple setup, we further extend our state representation. Specifically, we concatenate to our already-defined L^2-scalars long chessboard-state vector one further real-valued scalar $f_j \in [0, 1]$. When generating each initial state, a value for f_j is sampled uniformly from the $\{0, \dots, {}^{i}/_{F}, \dots, 1\}$ F-equipartition of the $[0, 1]$ interval. Such scalar is preserved by dynamical evolution and simply copied to the final state. Interestingly, since the different possible values for f_j are naturally ordered, and not bound to just $\{0, 1\}$, such change to system state breaks the natural symmetry induced by equidistance and orthogonality among states — and it does so in a completely uninformative fashion *w.r.t.* the underlying chessboard dynamics.

Partitioned observability

Finally, to complicate our task even further, we introduce some spurious correlation between chessboard states and the values of f_j. In particular, we consider — among all possible initial chessboard-states and all possible values of f_j — the most numerous set of the two. Then, the values of such set are associated uniformly at random and without repetition to the values of the other set. If *e.g.* $L = 8$ and $F = 4$, of all possible 64 chessboard states only approximately 16 fixed states would ever carry each of the 4 distinct f_j values.

5.3.2 Results

We experimentally test our framework, described in section 5.2, in the setup just outlined.

Multi-task instance

In particular, we consider a chessboard with $L = 16$, thus producing states composed of 257 scalars — where f_js are to be chosen from 100 distinct equispaced values. Given such disproportion between the numerosity of the two sets, we allow the $c_{\text{emb}}\ \mathcal{L}_{\text{emb}}$ loss term to be rewritten as:

$$c_{\text{emb}}^{\text{chess}}\ \mathcal{L}_{\text{emb}}^{\text{chess}} + c_{\text{emb}}^{\text{f}}\ \mathcal{L}_{\text{emb}}^{\text{f}}$$

in order to more carefully control the strength of each reconstruction term separately.

We choose 3 tasks as *training tasks*, whereas all the others are then fitted, each separately, during test-time adaptation. Both the *sticky* and *loopy* edges variants are considered. Specifically, we choose tasks $\uparrow$ (*North*), $\swarrow$ (*South-West*) and $\searrow$ (*South-East*) as training tasks — since they allow to cover all possible edges with the minimum number of tasks, while being the most spread-out one another.

Model instance and hyperparameters

We define our model and associated hyperparameters by choosing $\alpha_{\boldsymbol{\theta}_\alpha}$ to be a learnable affine transformation with output size $d_e = 4$, $\varphi_{\boldsymbol{\theta}_\varphi}$ to be a *2-layers multi-later perceptron* with ReLU nonlinearity, output size $d_d = 5$ and hidden size 10, and constrain T_is to have rank 1 via matrix factorisation. $\beta_{\boldsymbol{\theta}_\beta}$ is finally chosen to be a *2-layers MLP* with ReLU nonlinearity and hidden size 260.

We further choose distance $\mathcal{D}$ — instrumental in defining losses $\mathcal{L}_{\text{emb}}^{\text{chess}}$ and $\mathcal{L}_{\text{emb}}^{\text{f}}$ — to be pixel-averaged cross-entropy. Mixing coefficients are set to be $c_{\text{dyn}} = 225$, $c_{\text{emb}}^{\text{chess}} = 1$ and $c_{\text{emb}}^{\text{f}} = 1$. Regularisation is ensured by setting $\epsilon = 0.25$ and $\epsilon_{\text{ins}} = 0.0085$.

Training-time optimisation is carried out on shuffled batches of size 256, for 125 epochs, by the RAdam [138] optimiser with a constant learning rate of 0.002. Test-time adaptation is instead performed on 10 randomly-harvested examples per task, with a Ridge regularisation weight of 10^{-7} and 75 ALS iterations. Remarkably, the specific number of 10 examples is the minimum for which the resulting rank-constrained model is not underdetermined on test data.

Results

After completion of the training process, the resulting model appears to have converged, with training losses (averaged over the entire training-set) of $\mathcal{L}_{\text{dyn}} < 10^{-5}$, $\mathcal{L}_{\text{emb}}^{\text{chess}} < 10^{-3}$ and $\mathcal{L}_{\text{emb}}^{\text{f}} < 10^{-7}$.

Considering instead per-task *end-to-end* next-state prediction loss — *i.e.*

$$\mathcal{L}_{\text{e2e}(i)} \coloneqq \left\langle \mathcal{D}(\hat{\boldsymbol{x}_{ij}}', \boldsymbol{x}_{ij}') \right\rangle_j ,$$

with $\mathcal{D}$ being pixel-averaged cross-entropy — still averaged over the training set, we obtain $\mathcal{L}_{\text{e2e}(i)}^{\text{chess}} < 10^{-2}$ and $\mathcal{L}_{\text{e2e}(i)}^{\text{f}} < 10^{-6}$ independently of the specific task considered.

Moving on to test-time adaptation, we report in Table 5.1 *end-to-end* next-state prediction losses, split by border-type scenario and portion of state (*i.e.* the *chessboard* or the

additional scalar f_j) being considered, computed on the test set. For each entry, we show the average and associated 3σ Gaussian confidence intervals, across tasks and 5 distinct re-samplings of the data being fit during the adaptation phase — the two sources of variation producing effects of similar magnitude.

Table 5.1: *Average test-set end-to-end next-state prediction losses, decoupled by state fragment being predicted, in the sticky borders and loopy borders scenarios. Associated 3σ Gaussian confidence intervals represent variability of both intra-task variation and data re-sampling across 5 runs.*

	sticky	***loopy***
$\mathcal{L}^{\text{chess}}_{\text{e2e}(i)}$ *(avg.)*	2.3 ± 0.7	1.0 ± 0.2
$\mathcal{L}^{\text{f}}_{\text{e2e}(i)}$ *(avg.)*	$(1.1 \pm 0.7) \times 10^{-5}$	$(2.3 \pm 0.9) \times 10^{-5}$

Such results — although clearly partial, and preliminary in nature — are still significant to validate our original intuition that drove the development of the framework we presented. Indeed, not only the resulting system is able to successfully learn training tasks with very low error in *end-to-end* next-state prediction — even in the presence of discontinuously state-dependent non-tipicalities, confounding variables, or spurious correlations. Most remarkably, it is also able to learn a representation structure that unlocks swift test-time adaptation to new similar task, with minimal additional data and computational expenditure.

To further validate that the learned representation is indeed what we expect it to be, we finally plot the representation $\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x})$ associated with each of the $F \times L^2 = 25600$ possible states $\boldsymbol{x}$ — as shown in Figure 5.1 after 4D $\rightarrow$ 3D dimensionality reduction by principal component analysis and colouring plotted points according to the value of f_j along the *viridis* palette [188].

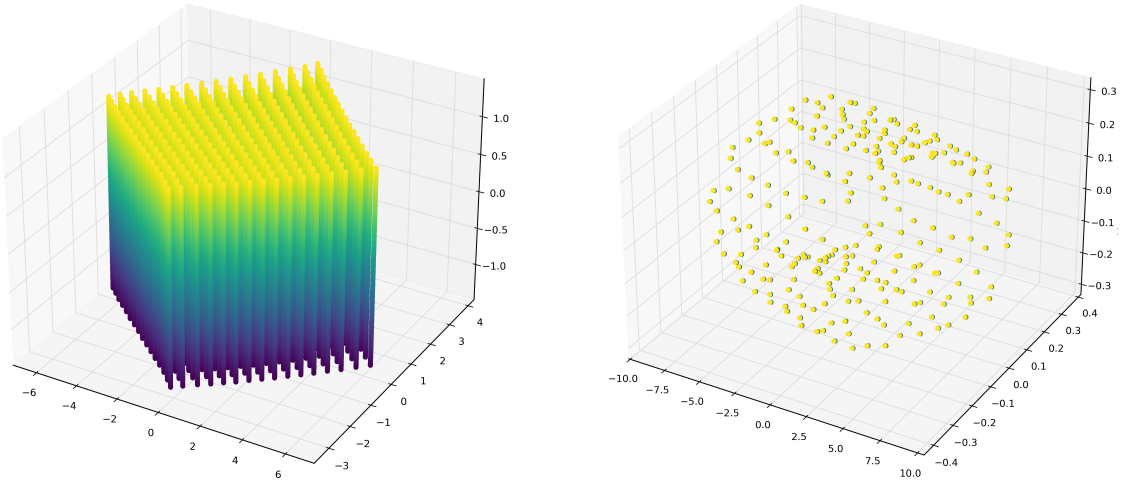

Figure 5.1: *Principal component analysis (4D $\rightarrow$ 3D) of learned representation $\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x})$ associated with each possible system state $\boldsymbol{x}$ in the scenario considered — with sticky (left) or loopy (right) borders. Colour-coding follows the value of f_j along the viridis palette.*

As it is possible to see — somehow also to our surprise — the resulting representations closely match our intuitive understanding of the experimental setup: *i.e.* a planar 2D Cartesian grid-like structure for the chessboard-state in the *sticky* setting, and a cylindrical one in the *loopy* setting. In the *sticky* case, such description adequately (and only) captures typical dynamics — leaving the handling of border behaviour to φs. In the *loopy* scenario, instead, both typical dynamics and part of border dynamics can be jointly described on the cylinder — accounting for the much lower and more concentrated reconstruction error for the chessboard-states in such scenario.

Furthermore, in the *sticky borders* setting, the representation associated with states carrying different values of f_j form vertical structures where f_js are ordered in increasing (or decreasing) order — mirroring the natural ordering induced by Euclidean distance among states. Such phenomenon is not visibly present in the *loopy* setting, where — apart from a higher reconstruction error for f_is — the cylindrical representation already fills the first 3 principal components. Appearing as compressed *dots* in the figure, groups of chessboard-states with different f_is will likely develop along the non-visible fourth dimension of α. This concludes our assessment of the method.

For completeness, Figure 5.2 shows the evolution of the learned representations $\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x})$ as training progresses.

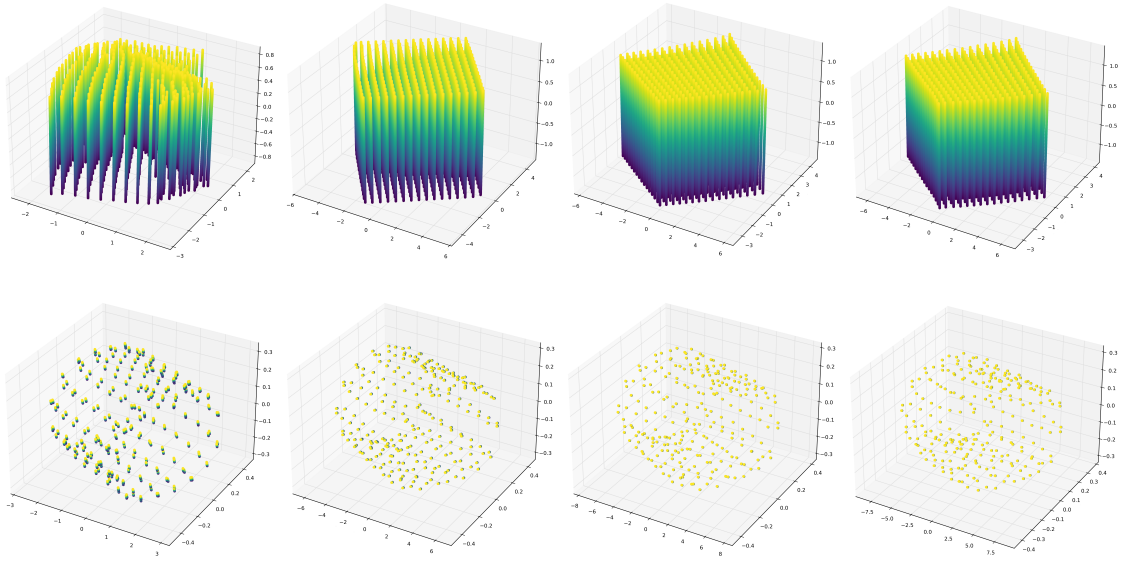

Figure 5.2: *Evolution of learned representations $\alpha_{\boldsymbol{\theta}_\alpha}(\boldsymbol{x})$ along training, in the sticky borders (top) and loopy borders (bottom) scenarios. Left-to-right: epoch 5, 35, 65, 95. Each graphical representation follows the same conventions as Figure 5.1.*

5.4 Conclusion

In this chapter, we introduced and preliminarily evaluated a novel framework that allows effective supervised multi-task learning and subsequent *very-few-shot* test-time adaptation with minimal additional data and compute. The method, based on a nonlinear generalisation of Koopman operator learning, describes input/output pairs as states

of a dynamical system subject to task-dependent dynamical evolution — and tries to effectively decouple, in such light, a global geometrical grounding of *typical* dynamical evolution from local, state-dependent deviations from it. On top of such structure, actual differences across tasks are simply described by linear operators, which can easily be fitted at test-time.

Our assessment, though still preliminary, corroborates the effectiveness of our framework — being able to capture a structure close to ideal when dealing with problems that can be easily described only in the adequate representation — and showcases its test-time adaptation ability. The same continues to be true when the problem is made increasingly more complex by the additions of deliberate confounders.

5.4.1 Future developments

When such framework was originally devised, however, we did not have only chessboards and chess-pieces in mind. Instead, we were motivated by all those scenarios — still hard for contemporary machine learning — where fast test-time adaptation is paramount, and where effective solutions require a carefully-tuned interplay between structure and learnability. As a motivating example, we can think about the *reality gap* in robot control — where a robot, whose control policy has been learned via simulated physics, finally faces the real world and needs to adapt to minor but crucial discrepancies, as fast as possible. Or we can think of well-functioning vehicles or appliances, whose reliable operation must be ensured even in the event of damage or part failure.

Groundwork in such sense is already underway. In particular, we are considering the problem of next-state prediction for simulated systems whose dynamics is induced by a previously-learned or expert-driven policy — and adaptation must occur *w.r.t.* slight but insidious variations in system-wide physical constants.

Chapter 6

Emergent representations in networks trained with the Forward-Forward algorithm

6.1 Introduction

Deep Learning is a highly effective approach to artificial intelligence, with tremendous implications for science, technology, culture, and society. At its core, there is the Backpropagation (Backprop) algorithm [32], which efficiently computes the gradients necessary to optimise the learnable parameters of an artificial neural network. Backprop, however, lacks biological plausibility [33] – leading to many attempts to address the issue. One of the most recent approaches, the Forward-Forward algorithm [34], eliminates the need to store neural activities and propagate error derivatives along the network.
In a standard classification context, the application of Forward-Forward requires the designation of positive and negative data. For example, to classify images, one could assign positive (or negative) data to those images having their correct (or incorrect, respectively) class label embedded via one-hot encoding at the border (as shown in Figure 6.1, Panel **A**). The Forward-Forward algorithm then learns to discriminate between positive and negative data by optimising a goodness function (*e.g.*, the ℓ_2 norm of the activations), akin to contrastive learning [35]. Satisfactory results have been observed [34] for classification tasks on MNIST [189], a standard benchmark dataset. This work takes a step beyond performance evaluation, delving into the structure of the hidden representations learned by the Forward-Forward algorithm, uncovering their spontaneously sparse nature and drawing parallels to neural *ensembles* observed in the brain [30, 31].

We organise this chapter as follows. In subsection 2.5.1 we set the stage by providing a brief overview of the Forward-Forward algorithm and of neuronal ensembles. Then, section 6.2 is dedicated to the description of the models and datasets investigated and the methods used to analyse representations. Our analysis of the Forward-Forward representations begins in section 6.3, where we present our key findings. Specifically, in subsection 6.3.2, we show that the Forward-Forward algorithm spontaneously learns sparse representations, organised into artificial ensembles, *i.e.* small sets of highly spe-

cialised neurons that consistently co-activate for data in a given class. In subsection 6.3.3, we demonstrate that these ensembles can overlap, with individual units contributing to multiple ensembles when visual features are shared. Further, subsection 6.3.4 reveals that ensembles can arise on previously unseen categories, indicating a robust generalization of this representational mechanism. Notably, these ensembles can share units with those associated with seen categories, demonstrating effective integration of new information with concepts learned during training. Finally, in subsection 6.3.5, we examine the structure of the weights and show that the observed sparsity and ensemble formation arise from suppression mechanisms, analogous to the inhibitory processes mediated by biological neurons [36]. These findings are particularly striking because the Forward-Forward algorithm achieves these properties without requiring explicit regularisation to induce sparsity. We observe that, although optimising the cross-entropy loss for the same classification task does not appear to produce the sparse ensembles we observe, the phenomenon may not solely be due to the use of the Forward-Forward algorithm. In fact, similar results are obtained by optimising the same goodness function of Forward-Forward, with Backprop instead. This suggests that more focus should be put on the purpose and biological meaning of the loss function rather than the training algorithm [37]. We discuss our results in section 6.4.

In summary, our main results are as follows:

- The Forward-Forward algorithm yields sparse representations composed of small groups of highly specific units, which we refer to as ensembles by analogy with those observed in the cortex.
- Ensembles can emerge in a zero-shot manner for classes held out during training and they can share units across visually related categories.
- The emergence of sparsity and the formation of ensembles are not unique to Forward-Forward optimisation, as they can also be observed in networks trained with Backpropagation on the same objective function.

6.2 Methods

In this chapter, we investigate and compare the representations produced by three models: [1]

- A classifier in the style of that used by Hinton [34], trained with Forward-Forward (**FF**);
- A classifier identical to the above, but trained end-to-end with Backprop to optimise the same goodness function (**BP/FF**);
- A classifier trained with Backprop on the categorical cross-entropy loss, as customary (**BP**).

Such different scenarios are described individually in subsection 6.2.2, subsection 6.2.3 and subsection 6.2.4, respectively.

[1] From this point on, we use the term *model* to refer to the combination of network architecture and optimisation algorithm.

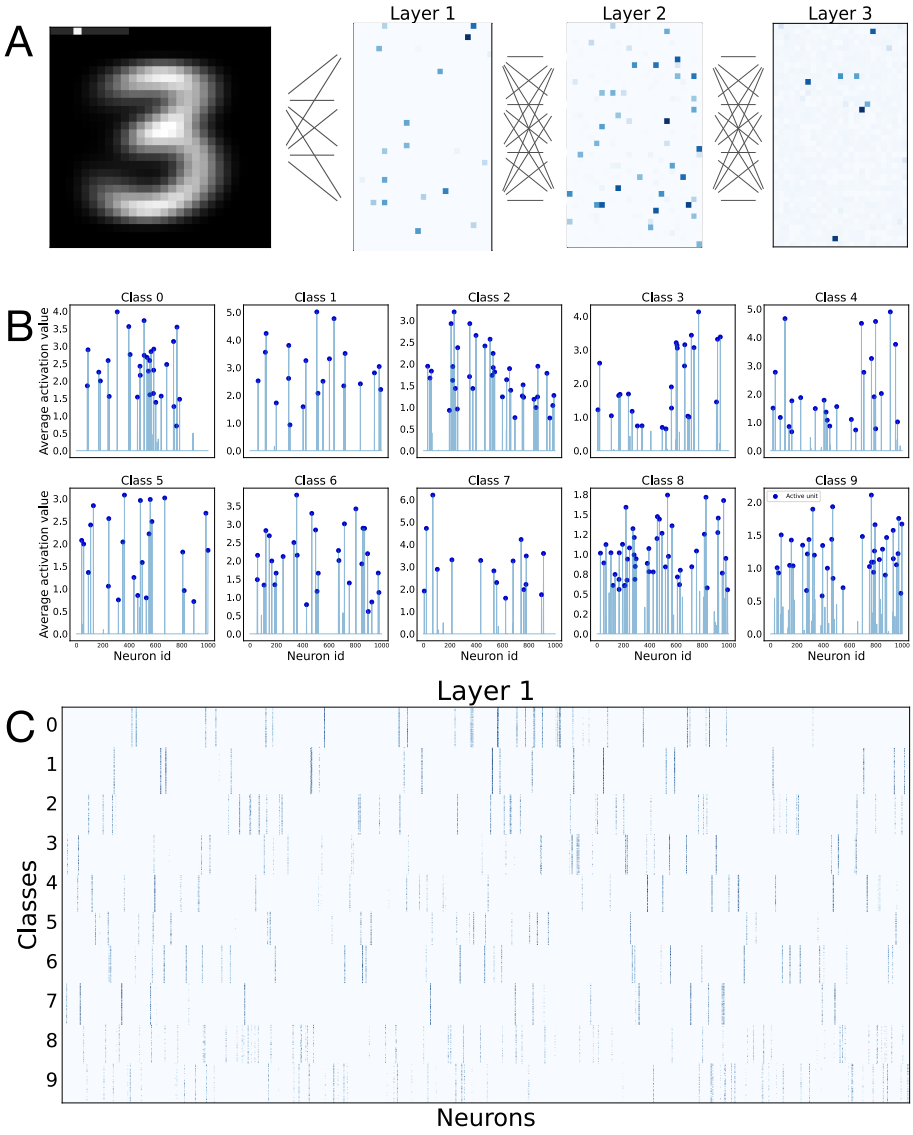

Figure 6.1: *Activation patterns in a Multi-Layer Perceptron trained with the Forward-Forward algorithm, on the MNIST dataset.*
*Panel **A** Examples of activation patterns in response to a positive input (class label embedded as a one-hot encoding on the top left corner of the image). Images show the activation value for network units, arranged as a matrix only for the sake of clarity; darker squares represent more active neurons.*
*Panel **B** Activation value of each neuron in the first hidden layer (Layer 1), averaged on all images of a given class. Neuron index on the x axis; average activation on the y axis. Blue dots indicate units that are considered active according to the leave-one-out (LOO) method described in subsection 6.2.5.*
*Panel **C** Activation map for neurons in Layer 1 for all images, grouped by class. A blue dot in position (x, y) indicates that neuron x is activated by input y; colour scale represents the intensity of such activation. Horizontal bands mark different categories; blue vertical stripes mark active, category-specific neurons. Each input category activates consistently a specific sets of neurons (ensemble).*

6.2.1 Data

The datasets we use to train and test the models described so far are MNIST [189], FASHIONMNIST [190], SVHN [191] and CIFAR-10 [192]. Details on these datasets are provided in the section D.1.

6.2.2 Model trained with Forward-Forward (FF)

Our **FF** model is inspired by the architecture proposed by Hinton [34] – and likewise trained according to the Forward-Forward algorithm. It consists of three fully-connected layers, each composed by 1000 units in the case of MNIST and FASHIONMNIST, and 3072 units in the case of SVHN and CIFAR-10. Each linear layer is followed by elementwise ReLU non-linearities. Both during training and inference, the layer-wise ℓ_2 norm is used as the goodness function of choice; correspondingly, ℓ_2 normalisation is performed between subsequent layers. Additional results obtained using the ℓ_1 norm as a goodness function are presented in section D.10.
To define positive and negative data, a one-hot-encoded class vector is embedded at the top-left corner of images. Prior to such embedding, these pixels are set to black colour. Then, in the case of positive data, the pixel corresponding to the true class is switched to the maximum value elsewhere observed in the image, while in the case of negative examples such value is randomly assigned to one of the other pixels of the embedding vector.
During training, the weights are optimised by minimising the loss function $L = \log(1 + e^{G_{neg} - G_{pos}})$, where G_{neg} and G_{pos} are, respectively, the goodness value for negative and positive data. At inference time, for each layer, the goodness values corresponding to every possible label are converted into a probability using softmax. By performing this step for each layer, they can contribute equally to the prediction.

For comprehensive details on the training procedures of the models discussed in this section and the next two sections, we direct the reader to section D.2.

6.2.3 Model trained with Backpropagation on the goodness objective (BP/FF)

The architecture of the **FF** model, while designed to be optimised using the Forward-Forward algorithm, can be trained seamlessly with Backprop on the same goodness maximisation/minimisation objective. Indeed, keeping the definition of positive and negative data introduced for **FF**, one could simply use Backprop to optimise the goodness-based loss from the Forward-Forward algorithm.
In detail, positive and negative data are fed to the network during the forward step, and the overall goodness of the internal representation is evaluated. The backward pass is then executed, and parameters are optimised to achieve the same goal as the **FF** model. It is worth pointing out that, in this case, the goodness is maximised globally instead of layer-by-layer (*i.e.*, locally).

6.2.4 Model trained with Backpropagation on the cross-entropy loss (BP)

The **FF** and **BP/FF** models are also compared to a standard neural classifier, serving as a baseline. For such purpose, a multi-layer perceptron is employed. The model shares the same number of layers, layerwise neuron count, and non-linear activation function choice with **FF** and **BP/FF**. The only architectural difference between the **BP** model and the other two is the addition of a final softmax layer, to suitably shape and scale the output for the classification task. The model is trained end-to-end with Backprop on the categorical cross-entropy loss.

6.2.5 Analysis of representations

For each model described, we analyse the internal representation emerging at each layer. We limit our analysis to data belonging to the test set (*i.e.* not seen during training) and correctly classified by the respective model. However, the main results of our analysis, concerning sparse and ensemble-like representations, extend without any modification to training data. Concretely, the representation of a single image is a n-dimensional vector composed by the activations (after the ReLU non-linearity) of all the units in the layer. For each layer, we extract a representation matrix X of size (M, n), where M is the total number of test images (correctly classified) and n is the number of neurons in the layer considered.

Sparsity

For each representation vector x we assign a *sparsity* measure following the notion of sparsity introduced in Hoyer [193]:

$$S(x) = \frac{\sqrt{n} - \frac{\|x\|_1}{\|x\|_2}}{\sqrt{n} - 1}$$

With this definition, when $S(x) = 1$ the vector x contains only one non-zero component representing the case of an extreme sparsity. The other limiting case is the one in which all the components of x are equal in magnitude, in this case $S(x) = 0$. The sparsity function S interpolates smoothly between these two extremes. The sparsity of a layer representation is obtained by averaging the sparsity of its component vectors $S = \frac{1}{M}\sum_{i=1}^{M} S(x_i)$.

Ensembles

To detect the emergence of category-specific ensembles, within each model and dataset combination, we adopt the following method. The idea is that a neuron should be considered active and part of an ensemble if it activates consistently and specifically when the network receives input data that belongs to that category.
We start by defining a category-specific representation matrix X_c, of shape (M_c, n), where M_c is the number of correctly classified test images of the given category. Then, we compute the average activation of each hidden unit across all samples: $\overline{x_{j,c}} = \frac{1}{M_c}\sum_{i=1}^{M_c}(X_c)_{ij}$; and the leave-one-out average of the averages $\text{LOO}_{j,c} = \frac{1}{n-1}\sum_{i\neq j}\overline{x_{j,c}}$. We then classify

a neuron i as active (*i.e.* part of an ensemble) if $\overline{x_{i,c}} > 2 \cdot \text{LOO}_{i,c}$. We also perform a significance test for these comparisons through a permutation test (see section D.3 and Table 6.3). Examples of average activation profiles and of ensembles are reported in Figure 6.1.

The output of the ensemble computation is a set of active units for each category: $\mathcal{E}^c = \left\{e_1^c, e_2^c, \dots e_{n_c}^c\right\}, \forall c \in \{1, 2, \dots, C\}$, where n_c is the number of active units for category c. Once the ensembles are defined, it is possible to look at units that are shared across categories c and c' by considering $\mathcal{E}^c \cap \mathcal{E}^{c'}$. The size of the shared units is naturally measured by $\mid \mathcal{E}^c \cap \mathcal{E}^{c'} \mid$.
We can also measure the similarity between two ensembles using the Jaccard similarity index (intersection over union): $J(\mathcal{E}^c, \mathcal{E}^{c'}) = \frac{|\mathcal{E}^c \cap \mathcal{E}^{c'}|}{|\mathcal{E}^c \cup \mathcal{E}^{c'}|}$. As an example, for two ensembles composed of 50 units with a substantial ensemble overlap of 30% (15 units), the similarity J is ≈ 0.18. Examples of shared units are reported in Figure 6.3 (see Table 6.3 for typical ensemble sizes in our setting).

When the sparsity S of a representation is low, ensembles are typically ill-defined as too many neurons are significantly active simultaneously and the notion of active unit tends to blur. To set a threshold, we will consider values of S below 0.5 as non-sparse, and in these cases, we do not define ensembles out of the representation.

6.3 Results

In this section, we describe our findings for the three models introduced, on the MNIST, FASHIONMNIST, SVHN and CIFAR-10 datasets. In particular, we focus on the properties of representations obtained within the **FF** model, *i.e.* a model trained with Forward-Forward on its natural goodness objective. Such properties, such as the emergence of category-specific ensembles and the presence of shared units across them, establish a link between neural networks trained with the Forward-Forward algorithm and biological cortical networks described in section 2.5.1.

6.3.1 Classification accuracy

Before we present the main results of our work, we evaluate the performances of our models on the classification tasks at hand. Table 6.1 contains results in terms of test set classification accuracy for all models we employed – **FF**, **BP/FF** and **BP** – on MNIST, FASHIONMNIST, SVHN and CIFAR-10. While some of these accuracy values are far from the state-of-the-art (*i.e.*, respectively, 0.997 [194], 0.931 [190], 0.860 [195] and approximately 0.7 [196], for fully-connected networks), they are a solid ground on which to build our subsequent investigations. Training details and hyperparameters for all models are reported in section D.2.

6.3.2 Forward-Forward elicits sparse neuronal ensembles

The **FF** and **BP/FF** models – based on the original Forward-Forward network architecture, and trained according to the goodness objective (subsection 6.2.2 and subsection 6.2.3) – exhibit typically high sparsity levels in their representations, in clear contrast with **BP**

Table 6.1: *Test-set classification accuracy for the models considered in our investigation. Results expressed as mean ± std. dev. over 10 runs with independent randomised weight initialisation.*

Dataset	**FF**	**BP/FF**	**BP**
MNIST	0.94 ± 0.008	0.969 ± 0.001	0.982 ± 0.001
FASHIONMNIST	0.849 ± 0.002	0.877 ± 0.002	0.892 ± 0.004
SVHN	0.716 ± 0.002	0.799 ± 0.004	0.793 ± 0.145
CIFAR-10	0.484 ± 0.004	0.521 ± 0.006	0.564 ± 0.004

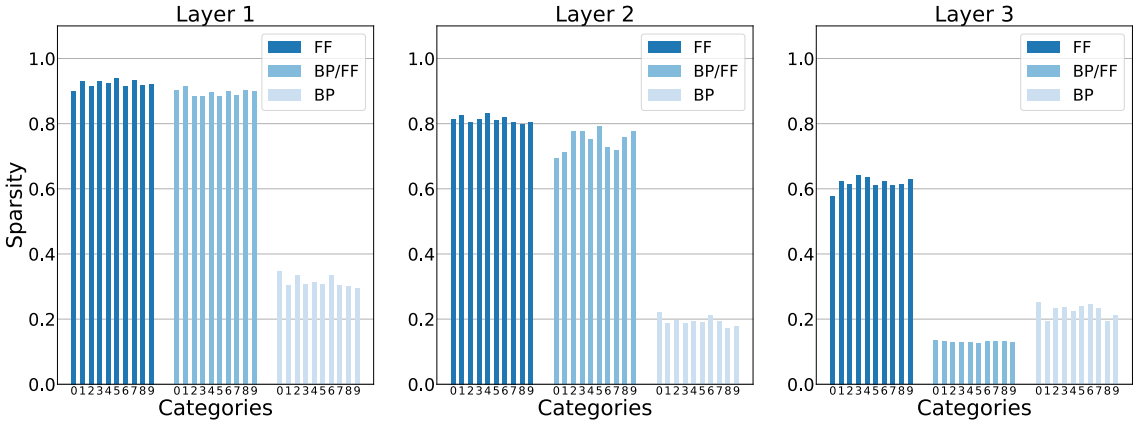

Figure 6.2: *Sparsity of category-specific representations. We report the sparsity of representations - computed as described in subsection 6.2.5 - for the three models **FF**, **BP/FF** and **BP** on the MNIST dataset. Sparsity values are the average over 10 runs.*

(see Figure 6.2 and Table 6.2). While sparsity does not spontaneously arise in **BP**, it can be enforced by means of ℓ_1 regularisation of the activations [197]. An analysis of this setting is presented in section D.8.

Table 6.2: *Average sparsity for all combinations of model, dataset and layer, according to the definition given in subsection 6.2.5. Results are expressed as mean ± std. dev. computed over 10 runs with independent random weights initialisation.*

Model	Layer	MNIST	FASHIONMNIST	SVHN	CIFAR-10
FF	1	0.922 ± 0.001	0.85 ± 0.002	0.83 ± 0.001	0.77 ± 0.001
	2	0.813 ± 0.019	0.605 ± 0.015	0.706 ± 0.001	0.728 ± 0.002
	3	0.618 ± 0.074	0.628 ± 0.013	0.489 ± 0.004	0.566 ± 0.002
BP/FF	1	0.895 ± 0.005	0.81 ± 0.007	0.783 ± 0.003	0.753 ± 0.004
	2	0.747 ± 0.013	0.851 ± 0.007	0.95 ± 0.003	0.932 ± 0.003
	3	0.131 ± 0.011	0.065 ± 0.009	0.133 ± 0.011	0.135 ± 0.009
BP	1	0.315 ± 0.003	0.352 ± 0.003	0.47 ± 0.02	0.478 ± 0.016
	2	0.193 ± 0.004	0.241 ± 0.005	0.524 ± 0.212	0.3 ± 0.18
	3	0.225 ± 0.006	0.248 ± 0.006	0.232 ± 0.106	0.164 ± 0.006

When the sparsity level is sufficiently high ($S > 0.5$) we are able to identify small sets

of neurons (ensembles) that consistently co-activate across all the samples of the same class, similar to what has been observed in cortical representations [30, 36, 95].

Figure 6.1 (Panels **B**, **C**) shows an example of average neuron activations for each class in Layer 1 of the **FF** model trained on Mnist, and showcases the emergence of sparse, category-specific, ensembles (see the section D.4 for a similar visualisation for Layers 2 and 3 of the same model and section D.5 for Layer 1 in all the models). These representations typically activate only a small fraction of units: ensembles consisting of just a few percent of the neurons in a layer are commonly observed, whether working with simpler datasets (e.g., Mnist, FashionMnist) or more complex ones (Svhn, Cifar-10), with a slight tendency of the **FF** model to create larger ensembles in the latter case (Table 6.3).

Table 6.3: *Average fraction of units taking part in ensembles, for all combinations of dataset and layers, in the **FF** and **BP/FF** models. Ensemble sizes are averaged across all categories, divided by the number of neurons in a layer, and then expressed in* $\%$*. Ensembles are defined according to the LOO method presented in subsection 6.2.5. Results expressed as mean* $\pm$ *std. dev. . In the third layer of **BP/FF**, as well as in **BP**, the representation is non-sparse. With (*) we marked the condition in which* $\approx 2.6\%$ *of the units have a* p*-value larger than* 0.05*, see section D.3 for details.*

Model	Layer	Mnist	FashionMnist	Svhn	Cifar-10
FF	1	3.69 ± 0.09	5.02 ± 0.14	10.3 ± 0.15	16.08 ± 0.09
	2	5.31 ± 0.35	18.46 ± 0.66	21.28 ± 0.23	21.2 ± 0.3
	3	1.36 ± 0.36	20.59 ± 0.63	4.48 ± 0.52	4.86 ± 0.51
BP/FF	1	8.58 ± 0.23	13.24 ± 0.31	15.07 ± 0.16	13.3 ± 0.13
	2	13.18 ± 0.67	8.45 ± 0.47	5.08 ± 0.19	$5.55 \pm 0.28(*)$
	3	-	-	-	-

Overall, these findings show that networks trained with the Forward-Forward objective produce highly sparse representations, characterised by *ensembles*, *i.e.* small groups of neurons with category-specific activation patterns.

6.3.3 Visually similar classes can elicit ensembles with shared neurons

Drawing a parallel with a phenomenon observed in Neuroscience [103], related categories can be expected to share units of their ensembles. This is indeed what we observe, as shown in Figure 6.3. Results are reported for FashionMnist, where different classes of clothes or shoes may contain a common share of visual features. In this regard, we observe a clear tendency to share units between similar classes – *e.g.* across representations of `pullover`, `coat` and `shirt`.

In the following section, we provide evidence that a unit can be shared across two ensembles even if one of these refers to an unseen category (*i.e.*, excluded from the training set but whose representation, extracted at test time, generates an ensemble), as we show in Figure 6.4 (Panel **C**).

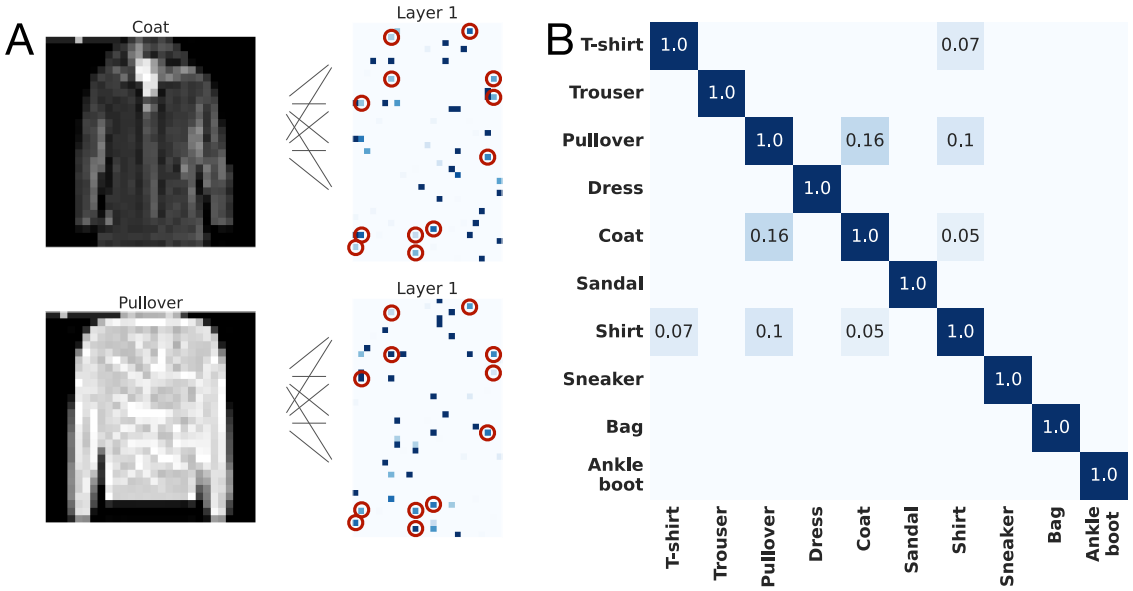

Figure 6.3: *Visually similar classes in FASHIONMNIST can elicit ensembles with shared neurons. Panel **A** The ensembles elicited in the first hidden layer of **FF** by two example inputs. Red circles indicate the active units which are shared between the two categories. Panel **B** Element i, j of the matrix indicates how many units are shared between the ensembles of category i and category j (normalised by the ensemble sizes), by using the Jaccard similarity index: $J(\mathcal{E}^i, \mathcal{E}^j) = \frac{|\mathcal{E}^i \cap \mathcal{E}^j|}{|\mathcal{E}^i \cup \mathcal{E}^j|}$. The results are referred to a single training run.*

These findings indicate that ensembles relative to visually related classes can partially overlap.

6.3.4 Representations of unseen categories can elicit well-defined ensembles

We investigate the ability of a trained **FF** model to respond to unseen categories with a coherent activation pattern which is typical of the ensembles we found on the categories seen at training time. To this end, we repeatedly train **FF** on FASHIONMNIST, removing one category at a time. Then, we extract the representation of the missing category, and verify if an ensemble is formed. We find that in all the ten cases, this is indeed the case, and the new ensemble share the same characteristics of the ones emerging for seen categories, apparently with the only exception of a lower average activation of their constituent units (see Figure 6.4 for one example, and the section D.6 for a more detailed account).

In several cases, we also find that the ensembles of unseen categories share units with the ensembles of seen categories, when endowed with similar visual features (Figure 6.4, Panel **C**). A more extensive exploration of these cases is also reported in the section D.6.

These results show that ensemble-like structures can arise in a zero-shot setting on data categories held out of the training set.

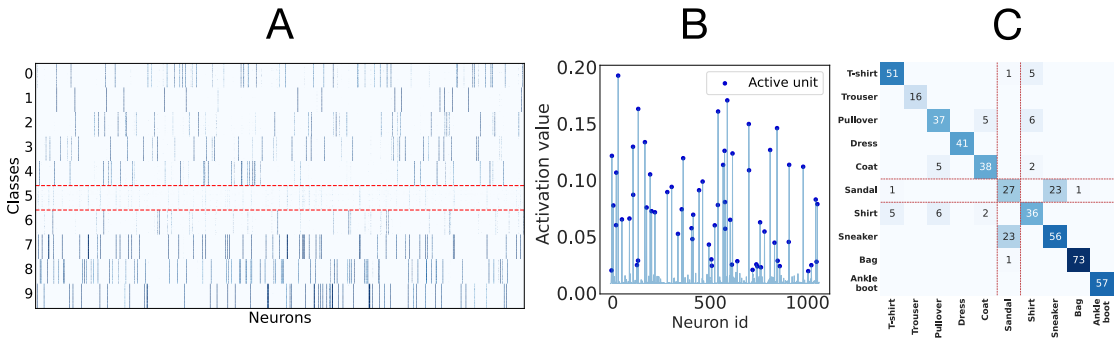

Figure 6.4: *The representations of an unseen category form an ensemble in* ***FF*** *trained on* FASHIONMNIST.
Panel ***A*** *Activation patterns in response to the different categories in the first hidden layer. The unseen category (*`Sandal`*), surrounded by red lines, produces a relatively weaker but well-defined ensemble-like activation pattern.*
Panel ***B*** *Activation value of each neuron, averaged on all images of the unseen category. Neuron index on the x axis; average activation on the y axis. Blue dots indicate units that are considered active according to the method described in subsection 6.2.5.*
Panel ***C*** *Ensembles of unseen categories can share units with the ensembles of the other categories. Element i, j of the matrix indicates how many units are shared between the ensembles of category i and category j: $\mid \mathcal{E}^i \cap \mathcal{E}^j \mid$. The results are referred to a single training run.*

6.3.5 Distribution of excitatory and inhibitory connections

As we observed in subsection 6.3.2, **FF** and **BP/FF** have comparable sparsity levels and, when they are defined, the ensembles have comparable sizes. A more fine-grained inspection of the representations learned by different models at different layers can be found in section D.9. The presence of sparse ensembles suggests that a strong inhibition mechanism is at work, leaving only a few neurons active for each data sample. Inhibition in these architectures is the result of an interplay between the sign and magnitude of the weights and the those of the biases. Therefore it is natural to wonder whether **FF** and **BP/FF** are similar also in this interplay between weights and biases. We find that the answer is *no*: the **FF** and **BP/FF** are indeed two profoundly different models that create sparse representations and ensembles with different mechanisms.

To show this, we consider for each neuron i in a layer with width n the fraction of its positive weights *w.r.t.* the total number of its input connections (ϱ_i^+ in the following), and its bias β_i. From these neuron-level quantities we construct their layer averages: $\varrho^+ = \frac{1}{n}\sum_i \varrho_i^+$ and $\beta = \frac{1}{n}\sum_i \beta_i$. The neuron's weights are strongly imbalanced towards inhibition when $\varrho_i^+ \approx 0$ and, *viceversa*, strongly imbalanced towards excitation when $\varrho_i^+ \approx 1$; when $\varrho_i^+ \approx 0.5$ we will say that the neuron's weights are almost perfectly balanced; similar considerations hold for the biases, where a large and negative β_i means strong inhibition for the i-th unit.

Focusing on the second hidden layer, we observe macroscopic differences in the empirical distribution of ϱ_i^+ among the three models (see Figure 6.5), with 1) a dominance of positive weights in the case of **FF**, 2) a bimodal distribution of ϱ_i^+ in **BP/FF**, with two populations

of imbalanced units in opposite directions, and 3) a unimodal and approximately balanced distribution for all the neurons in the **BP** model.

In **FF**, we find that the bias distribution is strongly imbalanced towards inhibition with a *mean* $\pm$ *std.dev.* value of -1.66 ± 1.534. On the contrary, **BP/FF** and and **BP** show a substantial balance (-0.017 ± 0.015 and 0.003 ± 0.018, respectively). Therefore, although the weights of the **FF** model appear imbalanced towards excitation, the average bias β is very large and negative, and this might explain why, for this model, we observe such high sparsity values. Conversely, the **BP/FF** model has substantially zero bias (within very small fluctuations), therefore the inhibitory mechanism at work here does not rely upon a negative bias, but on the weights' configuration. From these results, we conclude that not only the training objective (*i.e.* goodness-based vs. categorical cross-entropy minimisation), but the specific training protocols (**FF** vs **BP/FF**) are determinant in shaping a different interplay between excitation and inhibition, even when similar sparsity levels are observed.

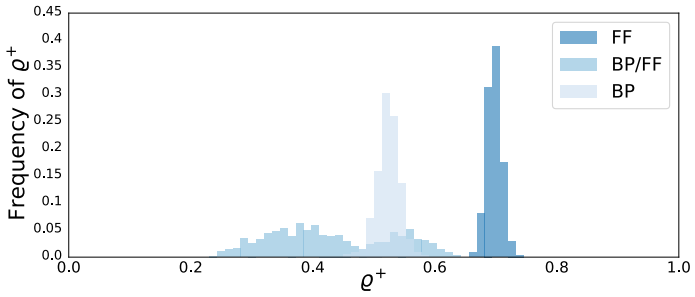

Figure 6.5: *Distribution of ϱ_i^+ in Layer 2 (Mnist dataset). In **FF**, the distribution is imbalanced, with most of the population of neurons having $\approx 65-75\%$ of excitatory weights. In **BP/FF** the distribution is bimodal with two populations of neurons: one inmbalanced towards excitation (right mode) and the other towards inhibition (left mode). The **BP** model is almost perfectly balanced between excitation and inhibition.*

We report the summary statistics ϱ^+, β for all the combinations of models, layers and datasets in Table 6.4 and Table 6.5. In all settings, we observe a similar picture, with the inhibition mechanisms dominated by negative biases in **FF** and negative weights in **BP/FF**.

In summary, these findings illustrate that although **FF** and **BP/FF** learn similar representations, they achieve sparsity by relying on fundamentally different neuron inhibition strategies.

Table 6.4: *Average fraction of positive weights (ϱ^+) for each combination of models, datasets and layers. Results are expressed as mean $\pm$ std. dev. over a single training run.*

Dataset	Layer	**FF**	**BP/FF**	**BP**
Mnist	1	0.661 ± 0.031	0.534 ± 0.028	0.486 ± 0.022
	2	0.688 ± 0.014	0.445 ± 0.099	0.523 ± 0.019
	3	0.882 ± 0.078	0.535 ± 0.071	0.52 ± 0.021
FashionMnist	1	0.457 ± 0.099	0.509 ± 0.018	0.491 ± 0.021
	2	0.602 ± 0.074	0.423 ± 0.065	0.52 ± 0.02
	3	0.416 ± 0.191	0.433 ± 0.019	0.52 ± 0.02
Svhn	1	0.487 ± 0.062	0.499 ± 0.009	0.499 ± 0.006
	2	0.521 ± 0.033	0.427 ± 0.042	0.504 ± 0.011
	3	0.583 ± 0.096	0.422 ± 0.031	0.522 ± 0.023
Cifar-10	1	0.493 ± 0.027	0.502 ± 0.011	0.501 ± 0.009
	2	0.489 ± 0.044	0.43 ± 0.038	0.513 ± 0.012
	3	0.612 ± 0.046	0.408 ± 0.034	0.523 ± 0.02

6.4 Discussion and conclusions

In many brain circuits, only a small fraction of neurons is active under specific sensory or behavioural conditions. It well established, indeed, that both sensory cortex and the hippocampus exhibit markedly sparse activity. The exact percentages can vary with species, types of stimuli, brain state, measurement technique, and brain area. For example, in rodent primary visual cortex, only about $10 - 20\%$ of excitatory neurons respond significantly to visual stimuli of varied complexity, from simple ones, such as oriented gratings to complex ones, like movies [30, 198]. In the auditory cortex, the fraction of neurons activated by a particular sound can be as low as $5 - 15\%$ [199, 200]. Lastly, hippocampal recordings during animal exploration show that only $20 - 40\%$ of neurons become active in different environments [201]. This sparse activity is often organised into *ensembles*, small groups of neurons that consistently co-activate in response to sensory stimuli or during spontaneous activity, and they have been proposed as functional building blocks of sensory processing, memory, and behaviour [30, 31, 94–99]. We discussed the notion of ensemble in section 2.5.1.

The main finding of our work is that artificial neural networks trained with the Forward-Forward algorithm can elicit sparse representations that share intriguing analogies with the neuronal ensembles found in real brains. We started our investigation by collecting and analysing representations from **FF** networks trained on Mnist, FashionMnist, Svhn, and Cifar-10 and defined, separately for each category, subsets of units (artificial ensembles) that prominently and consistently activated in response to data in such category (subsection 6.3.2). These category-specific ensembles turn out to be composed of a few active units, which is consistent with the aforementioned experimental findings on the sensory cortex and the hippocampus. Furthermore, when image categories are characterised by a certain degree of visual similarity, the corresponding ensembles often

Table 6.5: *Average bias (β) for each combination of models, datasets and layers. Results expressed as mean $\pm$ std. dev. over a single training run.*

Dataset	Layer	**FF**	**BP/FF**	**BP**
Mnist	1	-1.57 ± 2.029	-0.007 ± 0.019	0.001 ± 0.021
	2	-1.66 ± 1.534	-0.017 ± 0.015	0.003 ± 0.018
	3	-0.313 ± 0.151	-0.071 ± 0.02	0.002 ± 0.018
FashionMnist	1	-0.694 ± 0.706	-0.007 ± 0.018	0.003 ± 0.021
	2	-1.858 ± 0.489	-0.011 ± 0.018	0.003 ± 0.019
	3	-1.437 ± 1.213	-0.028 ± 0.009	0.003 ± 0.019
Svhn	1	-0.662 ± 0.219	-0.033 ± 0.012	-0.004 ± 0.013
	2	-0.959 ± 0.089	-0.005 ± 0.011	0.003 ± 0.012
	3	-0.962 ± 0.271	-0.001 ± 0.016	0.003 ± 0.011
Cifar-10	1	-0.402 ± 0.1	-0.022 ± 0.008	-0.001 ± 0.016
	2	-0.703 ± 0.132	-0.006 ± 0.01	0.003 ± 0.016
	3	-0.873 ± 0.076	-0.004 ± 0.013	0.002 ± 0.013

share one or more units (Figure 6.3, Panel **B**). The fact that single units can appear in multiple ensembles for different categories parallels the idea of mixed selectivity neurons. Mixed selectivity refers to neurons that respond in complex ways to combinations of characteristics or stimuli, thus increasing the dimensionality of population activity and allowing for flexible behaviour [101, 102]. Neurons that participate in more than one ensemble can be conceptually viewed as exhibiting mixed selectivity, since they contribute to multiple learned representations simultaneously.

We then tested the ability of trained **FF** models to cope with new data, and observed that activations in response to an unseen input category form, in many cases, a new ensemble with sparsity characteristics similar to those formed for other classes during training (Figure 6.4, Panels **A** and **B**). We also noticed that the ensembles of unseen categories often show a high level of similarity and integration with the ensembles of the categories of data encountered during training, realised through the sharing of units (see Figure 6.4, Panel **C** and also the results in section D.6). Beyond this qualitative similarity with the ensembles formed in response to seen categories, we showed, by training linear probes on representations, that the information content in these activation patterns is almost as high as that of seen categories Table D.2. While the **BP** model typically achieves higher decoding performance in this task, it does so by relying on a dense coding scheme. These findings suggest that the ensembles generated by **FF** in response to new data can support zero-shot classification tasks, which is particularly relevant in view of the importance of zero/few-shot learning in human and animal cognitive performance [202].

While absent in the **BP** model, the existence of ensembles composed of a few units is not unique to **FF**. It was indeed observed also in **BP/FF** (subsection 6.3.2), where the ensembles turned out to be of comparable size. This similarity in how representations are organised in **FF** and **BP/FF** is also backed by quantitative analysis with established representation similarity metrics section D.9. However, despite their similarity at representation level,

the **FF** and **BP/FF** models are profoundly diverse, as demonstrated by the different interplay between inhibition and excitation in these models (see subsection 6.3.5). We observed in this regard that the excitatory/inhibitory (E/I) balance play a key role in the stability of cortical networks and in brain dynamics [203, 204].

The sparsity of representations has computational benefits in sensory processing. Olshausen and Field [111] emphasised that sparsity may be the optimal encoding strategy for neural networks because it is energy efficient. This is especially important for biological neural networks, which operate under metabolic constraints. Sparsity also increases the memory-storage capacity and eases readout at subsequent processing layers. Babadi and Sompolinsky [205] showed that sparse and expansive coding (*i.e.* from a lower dimensional sensory input space to a higher dimensional neural representation) reduced the intra-stimulus variability, maximised the inter-stimulus variability, and allowed optimal and efficient readout of downstream neurons. This is the reason why sparse and expansive transformations are widespread in biology, *e.g.*, in rodents [206] or flies [207].

Limitations

The limitations of the present work could be addressed by applying similar analyses to more complex datasets and a variety of tasks. To scale to a more challenging dataset may require the replacement of a fully connected network with more suitable architectures trainable with the Forward-Forward protocol, *e.g.* the CNNs recently introduced in Papachristodoulou et al. [91] and Sun et al. [92]. This approach could provide insights into how data characteristics influence sparsity levels and the resulting ensemble-like structures. Moreover, as highlighted in Yang [90], the choice of hyperparameters potentially affects representation sparsity, as well as the goodness function of choice.

Future work

Many questions are left open and will be addressed in future works. A closer inspection of the activation patterns within each category will be necessary to test for the co-existence of multiple patterns, with one dominant and possibly many subdominant patterns. We have not yet investigated this *microstructure* [208], leaving it to possible extensions of this work. Based on experimental results indicating the presence of small category-specific ensembles, a promising avenue for future research in this field encompasses exploring model compression through pruning [209], with the design of new strategies based on the relevance of the ensembles, as well as investigating the dynamic evolution of ensemble size and organisation throughout the training process. An especially intriguing perspective, inspired by recent work on optimal sparsity in hippocampal memory models [210], suggests that sparsity levels are *dynamic* variables, rather than fixed properties of the network, that can be tuned to the compressibility of sensory inputs to reach optimal performance. Such inquiries hold the potential to shed light on the formation, evolution, interactions, and persistence or replacement of ensembles in artificial neural networks. Lastly, from the analysis of representations learned by the Forward-Forward algorithm, our work suggests novel directions for comparing artificial and biological representations [211, 212] – particularly for biologically plausible learning algorithms – by leveraging a well-established concept in Neuroscience: that of neuronal ensembles.

Chapter 7

Conclusion

Modern-day artificial intelligence — built upon machine learning and neural networks research — has already managed to carve and establish its place in contemporary society. Countless institutions, businesses, and individuals alike already use it daily in activities once thought an essential prerogative of human beings. Mutually propelling and being propelled by such societal change, also the development of AI itself has changed pace — with new model architectures, applications, and AI-enabled products being developed daily — and the need for new data, compute, and energy towards this purpose is ever increasing.

Along such successful upward spiral, however, some gaps between artificial and human intelligence remained wide or even increased — in particular those aspects where cognitive structure, resilience, and adaptability are in interplay. This warrants the need for further research on the matter. In this thesis, still far from closing such gaps, we analysed some of those peculiar and intriguing phenomena — offering at the same time some effective and actionable mitigations.

We first investigated the problem of adversarial robustness — which makes deep learning models brittle and potentially untrustworthy — and developed Carso, a *state of the art* adversarial defence. Such method blends techniques from adversarial training, adversarial purification, and model ensembling by making the role of the internal representation of the attacked classifier more central.

Then, we showed that the same kind of methods used to train neural network models can lead to artificial physical systems that are more adaptable to unforeseen operative circumstances. In particular, by minimally modifying a photovoltaic system harvesting energy from incident radiation — with the addition of learnable components — its efficiency can be made almost optimal regardless of radiation frequency.

Furthermore, we developed a mathematical framework for multi-task supervised representation learning that allows the fast and generalisable adaptation to new unforeseen tasks at test-time. The method, based upon a nonlinear generalisation of Koopman operator learning and originally devised in the context of dynamical systems, is still under development — but holds potential to be used for *very-few-shot* imitation learning and may ameliorate the *reality gap* in the field of robotic simulation and control.

Finally, we analysed the internal representations that emerge in neural networks trained by the Forward-Forward algorithm, finding out that such biologically-plausible learning algorithm — similar to contrastive learning — makes them sparse and endowed with ensemble-like properties reminiscent of the sensory cortex. However, we also discovered that the specific choice of *goodness function* — the FF equivalent of the *loss function* in traditional deep learning models — plays an equally relevant role in such sense.

Coda

At this point, one may ask what is the purpose of all this. The *ultimate* goal, the reason why we took the *first* step of our journey. If we are allowed a shot of optimism — and think of transformative innovations AI has allowed so far — we can only imagine what a more flexible, adaptable, and intuitive AI would produce. Still firmly governed by humans, but equipped of some of the traits that make human intelligence so unique.

An unusual linguistic perspective may help shed even more light on the matter. Several expressions in vernacular English associate tools — as fixed-purpose implements — to a lack of cognitive prowess or thought independence. Among those, the term *tool* in itself, which — when referred directly to a person — carries a derogatory meaning. Under such light, we want to make AI a tool, which is also not *a tool*. That would be the ultimate purpose. ■

Bibliography

[1] E. Ballarin, A. Ansuini and L. Bortolussi. 'Blending adversarial training and representation-conditional purification via aggregation improves adversarial robustness'. In: *Transactions on Machine Learning Research*. 2025 (cit. on p. 1).

[2] E. Ballarin, D. A. Chisholm, A. Smirne, M. Paternostro, F. Anselmi and S. Donadi. 'Driving Enhanced Exciton Transfer by Automatic Differentiation'. In: *Machine Learning: Science and Technology* (2025) (cit. on p. 1).

[3] N. Tosato, L. Basile, E. Ballarin, G. de Alteriis, A. Cazzaniga and A. Ansuini. 'Emergent representations in networks trained with the Forward-Forward algorithm'. In: *Transactions on Machine Learning Research*. 2025 (cit. on p. 1).

[4] E. Ballarin, F. Giacomarra, A. Mecchina and N. A. Pearson. 'Demystifying Generative AI: A Pedagogical Approach'. In: *Ital-AI Workshop on Generative AI for Education*. 2025 (cit. on p. 1).

[5] R. Appel, P. McCrory, A. Tamkin, M. Stern, M. McCain and T. Neylon. *Anthropic Economic Index report: Uneven geographic and enterprise AI adoption*. 15th Sept. 2025 (cit. on p. 1).

[6] T. Eloundou, S. Manning, P. Mishkin and D. Rock. *GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models*. 2023. arXiv: 2303.10130 (cit. on p. 1).

[7] M. Belkin, S. Ma and S. Mandal. *To understand deep learning we need to understand kernel learning*. 2018. arXiv: 1802.01396 (cit. on p. 1).

[8] D. Hagemann, M. Ihmels, N. Bast, A. B. Neubauer, A. Schankin and A.-L. Schubert. 'Fluid Intelligence Is (Much) More than Working Memory Capacity: An Experimental Analysis'. In: *Journal of Intelligence* 11.4 (2023) (cit. on p. 1).

[9] V. Scherrer, M. Breit and F. Preckel. 'Crystallized Intelligence, Fluid Intelligence, and Need for Cognition: Their Longitudinal Relations in Adolescence'. In: *Journal of Intelligence* 12.11 (2024) (cit. on p. 1).

[10] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto and F. Roli. 'Evasion Attacks against Machine Learning at Test Time'. In: *Proceedings of the 2013th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III*. 2013 (cit. on pp. 2, 15).

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus. 'Intriguing properties of neural networks'. In: *International Conference on Learning Representations*. 2014 (cit. on pp. 2, 6, 15).

[12] V. Ballet, X. Renard, J. Aigrain, T. Laugel, P. Frossard and M. Detyniecki. 'Imperceptible Adversarial Attacks on Tabular Data'. In: *Thirty-third Conference on Neural Information Processing Systems, Workshop on Robust AI in Financial Services: Data, Fairness, Explainability, Trustworthiness, and Privacy (Robust AI in FS)*. 2019 (cit. on pp. 2, 15).

[13] F. Tramèr. 'Detecting Adversarial Examples Is (Nearly) As Hard As Classifying Them'. In: *Proceedings of the International Conference on Machine Learning*. 2022 (cit. on p. 2).

[14] J. Nokkala, J. Piilo and G. Bianconi. 'Complex quantum networks: a topical review'. In: *Journal of Physics A: Mathematical and Theoretical* (2024) (cit. on pp. 3, 29).

[15] S. van Frank, M. Bonneau, J. Schmiedmayer, S. Hild, C. Gross, M. Cheneau, I. Bloch, T. Pichler, A. Negretti, T. Calarco et al. 'Optimal control of complex atomic quantum systems'. In: *Scientific reports* 6.1 (2016), p. 34187 (cit. on pp. 3, 29).

[16] Y.-C. Cheng and G. R. Fleming. 'Dynamics of light harvesting in photosynthesis'. In: *Annual Review of Physical Chemistry* 60 (2009), pp. 241–262 (cit. on pp. 3, 33, 36).

[17] O. Mülken and A. Blumen. 'Continuous-time quantum walks: Models for coherent transport on complex networks'. In: *Physics Reports* 502.2 (2011), pp. 37–87 (cit. on pp. 3, 29, 30).

[18] A. G. Baydin, B. A. Pearlmutter, A. A. Radul and J. M. Siskind. 'Automatic differentiation in machine learning: a survey'. In: *The Journal of Machine Learning Research* 18.153 (2018), pp. 1–43 (cit. on pp. 3, 18, 33).

[19] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong and Q. He. *A Comprehensive Survey on Transfer Learning*. 2020. arXiv: 1911.02685 (cit. on pp. 3, 41).

[20] J. Wei, M. Bosma, V. Zhao, K. Guu, W. Yu, B. Lester, N. Du, A. Dai and Q. V. Le. 'Finetuned Language Models are Zero-Shot Learners'. In: *International Conference on Learning Representations*. 2022 (cit. on pp. 3, 41).

[21] W. Lu, R. K. Luu and M. J. Buehler. 'Fine-tuning large language models for domain adaptation: exploration of training strategies, scaling, model merging and synergistic capabilities'. In: *Nature Computational Materials* 11 (2025), p. 84 (cit. on pp. 3, 41).

[22] D. Anisuzzaman, J. G. Malins, P. A. Friedman and Z. I. Attia. 'Fine-Tuning Large Language Models for Specialized Use Cases'. In: *Mayo Clinic Proceedings: Digital Health* 3.1 (2025), p. 100184 (cit. on pp. 3, 41).

[23] Q. Yang, L. Wang, J. Wicker and G. Dobbie. 'Continual Learning: A Systematic Literature Review'. In: *Neural Networks* (2025), p. 108226 (cit. on pp. 3, 41, 42).

[24] S. Dohare, J. F. Hernandez-Garcia, Q. Lan, P. Rahman, A. R. Mahmood and R. S. Sutton. 'Loss of plasticity in deep continual learning'. In: *Nature* 632.8026 (Aug. 2024), pp. 768–774 (cit. on p. 3).

[25] S. Lee, S. Goldt and A. Saxe. 'Continual Learning in the Teacher-Student Setup: Impact of Task Similarity'. In: *Proceedings of the International Coference on International Conference on Machine Learning*. 2022 (cit. on p. 3).

[26] J. v. Neumann. 'Zur Operatorenmethode In Der Klassischen Mechanik'. In: *Annals of Mathematics* 33.3 (1932), pp. 587–642 (cit. on pp. 3, 42).

[27] P. Bevanda, S. Sosnowski and S. Hirche. 'Koopman operator dynamical models: Learning, analysis and control'. In: *Annual Reviews in Control* 52 (2021), pp. 197–212 (cit. on pp. 3, 42).

[28] S. Grünewälder, G. Lever, L. Baldassarre, S. Patterson, A. Gretton and M. Pontil. 'Conditional mean embeddings as regressors'. In: *Proceedings of the International Coference on International Conference on Machine Learning*. 2012 (cit. on pp. 3, 42, 44).

[29] V. Kostic, P. Novelli, A. Maurer, C. Ciliberto, L. Rosasco and M. Pontil. 'Learning Dynamical Systems via Koopman Operator Regression in Reproducing Kernel Hilbert Spaces'. In: *Advances in Neural Information Processing Systems* 17 (2022) (cit. on p. 3).

[30] J.-e. K. Miller, I. Ayzenshtat, L. Carrillo-Reid and R. Yuste. 'Visual stimuli recruit intrinsically generated cortical ensembles'. In: *Proceedings of the National Academy of Sciences* 111.38 (2014), E4053–e4061 (cit. on pp. 4, 12, 53, 60, 64).

[31] R. Yuste, R. Cossart and E. Yaksi. 'Neuronal ensembles: Building blocks of neural circuits'. In: *Neuron* (2024) (cit. on pp. 4, 12, 53, 64).

[32] D. E. Rumelhart, G. E. Hinton and R. J. Williams. 'Learning Representations by Back-Propagating Errors'. In: *Nature* 323.6088 (1986), pp. 533–536 (cit. on pp. 4, 11, 53).

[33] D. G. Stork. 'Is backpropagation biologically plausible?' In: *Proceedings of the International Joint Conference on Neural Networks*. 1989 (cit. on pp. 4, 53).

[34] G. Hinton. 'The forward-forward algorithm: Some preliminary investigations'. In: *arXiv preprint arXiv:2212.13345* (2022) (cit. on pp. 4, 11, 12, 53, 54, 56).

[35] T. Chen, S. Kornblith, M. Norouzi and G. Hinton. 'A Simple Framework for Contrastive Learning of Visual Representations'. In: *Proceedings of the International Conference on Machine Learning*. 2020 (cit. on pp. 4, 53).

[36] R. Yuste. 'From the Neuron Doctrine to Neural Networks'. In: *Nature Reviews Neuroscience* 16.8 (2015), pp. 487–497 (cit. on pp. 4, 54, 60, 106).

[37] B. A. Richards, T. P. Lillicrap, P. Beaudoin, Y. Bengio, R. Bogacz, A. Christensen, C. Clopath, R. P. Costa, A. de Berker, S. Ganguli et al. 'A Deep Learning Framework for Neuroscience'. In: *Nature Neuroscience* 22.11 (2019), pp. 1761–1770 (cit. on pp. 4, 11, 54).

[38] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006 (cit. on p. 5).

[39] K. P. Murphy. *Machine Learning: a Probabilistic Perspective*. MIT Press, 2012 (cit. on p. 5).

[40] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge: Cambridge University Press, 2011 (cit. on p. 5).

[41] K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022 (cit. on p. 5).

[42] K. P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023 (cit. on p. 5).

[43] A. Krizhevsky, I. Sutskever and G. E. Hinton. 'ImageNet Classification with Deep Convolutional Neural Networks'. In: *Advances in Neural Information Processing Systems*. 2012 (cit. on p. 5).

[44] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning*. MIT Press, 2016 (cit. on p. 5).

[45] S. J. Prince. *Understanding Deep Learning*. The MIT Press, 2023 (cit. on p. 5).

[46] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023 (cit. on p. 5).

[47] J. Hertz, R. G. Palmer and A. S. Krogh. *Introduction to the Theory of Neural Computation*. 1st. Perseus Publishing, 1991 (cit. on p. 5).

[48] P. Petersen and J. Zech. *Mathematical theory of deep learning*. 2025. arXiv: 2407.18384 (cit. on p. 6).

[49] I. Goodfellow, J. Shlens and C. Szegedy. 'Explaining and Harnessing Adversarial Examples'. In: *International Conference on Learning Representations*. 2015 (cit. on pp. 6, 7, 15).

[50] A. Madry, A. Makelov, L. Schmidt, D. Tsipras and A. Vladu. 'Towards Deep Learning Models Resistant to Adversarial Attacks'. In: *International Conference on Learning Representations*. 2018 (cit. on pp. 6, 7, 15).

[51] S. Gowal, C. Qin, J. Uesato, T. Mann and P. Kohli. *Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples*. 2020. arXiv: 2010.03593 (cit. on pp. 6, 15, 83, 93).

[52] F. Croce and M. Hein. 'Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks'. In: *Proceedings of the International Conference on Machine Learning*. 2020 (cit. on pp. 6, 16, 23, 24).

[53] S. Gowal, S.-A. Rebuffi, O. Wiles, F. Stimberg, D. A. Calian and T. A. Mann. 'Improving Robustness using Generated Data'. In: *Advances in Neural Information Processing Systems*. 2021 (cit. on pp. 6, 15).

[54] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles and T. Mann. 'Data Augmentation Can Improve Robustness'. In: *Advances in Neural Information Processing Systems*. 2021 (cit. on pp. 6, 15, 93).

[55] Z. Wang, T. Pang, C. Du, M. Lin, W. Liu and S. Yan. *Better Diffusion Models Further Improve Adversarial Training*. 2023. arXiv: 2303.10130 (cit. on pp. 6, 15, 23, 25, 84, 94).

[56] J. Cui, Z. Tian, Z. Zhong, X. Qi, B. Yu and H. Zhang. *Decoupled Kullback-Leibler Divergence Loss*. 2023. arXiv: 2305.13948 (cit. on pp. 6, 15, 16, 23–25, 84, 94).

[57] S. Peng, W. Xu, C. Cornelius, M. Hull, K. Li, R. Duggal, M. Phute, J. Martin and D. H. Chau. *Robust Principles: Architectural Design Principles for Adversarially Robust CNNs*. 2023 (cit. on pp. 6, 15, 16, 94).

[58] B. R. Bartoldson, J. Diffenderfer, K. Parasyris and B. Kailkhura. 'Adversarial Robustness Limits via Scaling-Law and Human-Alignment Studies'. In: *Proceedings of the International Conference on Machine Learning*. 2024 (cit. on pp. 6, 16, 19, 24, 94).

[59] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui and M. I. Jordan. 'Theoretically Principled Trade-off between Robustness and Accuracy'. In: *Proceedings of the International Conference on Machine Learning*. 2019 (cit. on p. 6).

[60] S. Gu and L. Rigazio. 'Towards Deep Neural Network Architectures Robust to Adversarial Examples'. In: *Workshop Track of the International Conference on Learning Representations*. 2015 (cit. on pp. 6, 16).

[61] P. Vincent, H. Larochelle, Y. Bengio and P.-A. Manzagol. 'Extracting and composing robust features with denoising autoencoders'. In: *International Conference on Machine Learning*. 2008 (cit. on p. 6).

[62] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu and J. Zhu. 'Defense Against Adversarial Attacks Using High-Level Representation Guided Denoiser'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018 (cit. on p. 6).

[63] P. Samangouei, M. Kabkab and R. Chellappa. 'Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models'. In: *International Conference on Learning Representations*. 2018 (cit. on p. 6).

[64] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio. 'Generative Adversarial Nets'. In: *Advances in Neural Information Processing Systems*. 2014 (cit. on p. 6).

[65] Z. Zhang, M. Li and J. Yu. 'On the Convergence and Mode Collapse of GAN'. In: *SIGGRAPH Asia 2018 Technical Briefs*. 2018 (cit. on p. 7).

[66] M. Hill, J. Mitchell and S.-C. Zhu. 'Stochastic Security: Adversarial Defense Using Long-Run Dynamics of Energy-Based Models'. In: *International Conference on Learning Representations*. 2021 (cit. on pp. 7, 94).

[67] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato and F. Huang. 'A tutorial on energy-based learning'. In: *Predicting structured data*. MIT Press, 2006. Chap. 1 (cit. on p. 7).

[68] W. Nie, B. Guo, Y. Huang, C. Xiao, A. Vahdat and A. Anandkumar. 'Diffusion Models for Adversarial Purification'. In: *Proceedings of the International Conference on Machine Learning*. 2022 (cit. on pp. 7, 16, 24, 94).

[69] U. Hwang, J. Park, H. Jang, S. Yoon and N. I. Cho. 'PuVAE: A Variational Autoencoder to Purify Adversarial Examples'. In: *IEEE Access*. 2019 (cit. on p. 7).

[70] C. Shi, C. Holtz and G. Mishne. 'Online Adversarial Purification based on Self-supervised Learning'. In: *International Conference on Learning Representations*. 2021 (cit. on pp. 7, 16).

[71] Z. Yang, Z. Xu, J. Zhang, R. Hartley and P. Tu. 'Adversarial Purification with the Manifold Hypothesis'. In: *AAAI Conference on Artificial Intelligence*. 2024 (cit. on p. 7).

[72] J. Yoon, S. J. Hwang and J. Lee. 'Adversarial Purification with Score-based Generative Models'. In: *Proceedings of the International Conference on Machine Learning*. 2021 (cit. on pp. 7, 16, 94).

[73] H. Chen, Y. Dong, Z. Wang, X. Yang, C. Duan, H. Su and J. Zhu. *Robust Classification via a Single Diffusion Model*. 2023. arXiv: 2305.15241 (cit. on pp. 7, 16, 18).

[74] M. Lee and D. Kim. 'Robust Evaluation of Diffusion-Based Adversarial Purification'. In: *International Conference on Computer Vision*. 2024 (cit. on pp. 7, 16, 18, 24, 25, 90, 93, 94).

[75] D. P. Kingma and M. Welling. 'Auto-Encoding Variational Bayes'. In: *International Conference on Learning Representations*. 2014 (cit. on pp. 8, 16, 86).

[76] D. J. Rezende, S. Mohamed and D. Wierstra. 'Stochastic Backpropagation and Approximate Inference in Deep Generative Models'. In: *Proceedings of the International Conference on Machine Learning*. 2014 (cit. on pp. 8, 16).

[77] K. Sohn, H. Lee and X. Yan. 'Learning Structured Output Representation using Deep Conditional Generative Models'. In: *Advances in Neural Information Processing Systems*. 2015 (cit. on pp. 9, 16, 19).

[78] X. Yan, J. Yang, K. Sohn and H. Lee. 'Attribute2Image: Conditional Image Generation from Visual Attributes'. In: *Proceedings of the European Conference on Computer Vision*. 2016 (cit. on pp. 9, 16).

[79] D. J. Griffiths and D. F. Schroeter. *Introduction to Quantum Mechanics*. Third edition. Cambridge; New York, NY: Cambridge University Press, 2018 (cit. on p. 9).

[80] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. Usa: Cambridge University Press, 2011 (cit. on p. 9).

[81] A. Quarteroni, R. Sacco and F. Saleri. *Numerical Mathematics (Texts in Applied Mathematics)*. Berlin, Heidelberg: Springer-Verlag, 2006 (cit. on p. 9).

[82] S. Sgroi, G. Zicari, A. Imparato and M. Paternostro. 'Efficient excitation-transfer across fully connected networks via local-energy optimization'. In: *EPJ Quantum Technology* 11.1 (2024), pp. 1–18 (cit. on pp. 10, 30, 37).

[83] D. M N and T. A. Brun. 'Continuous limit of discrete quantum walks'. In: *Phys. Rev. A* 91 (6 June 2015), p. 062304 (cit. on p. 10).

[84] V. Gorini, A. Kossakowski and E. C. G. Sudarshan. 'Completely positive dynamical semigroups of N-level systems'. In: *Journal of Mathematical Physics* 17.5 (1976), pp. 821–825 (cit. on pp. 10, 31).

[85] G. Lindblad. 'On the generators of quantum dynamical semigroups'. In: *Communications in mathematical physics* 48 (1976), pp. 119–130 (cit. on pp. 10, 31).

[86] S. L. Brunton. *Notes on Koopman Operator Theory*. Department of Mechanical Engineering, University of Washington. Seattle, WA, 2019 (cit. on p. 10).

[87] S. L. Brunton and J. N. Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. 2nd ed. Cambridge: Cambridge University Press, 2022 (cit. on p. 10).

[88] M. Carandini and D. J. Heeger. 'Normalization as a canonical neural computation'. In: *Nature Reviews Neuroscience* 13.1 (2011), pp. 51–62 (cit. on p. 11).

[89] Y. LeCun and C. Cortes. *The MNIST handwritten digit database*. 2010 (cit. on pp. 12, 83).

[90] Y. Yang. 'A theory for the sparsity emerged in the Forward Forward algorithm'. In: *arXiv preprint arXiv:2311.05667* (2023) (cit. on pp. 11, 66).

[91] A. Papachristodoulou, C. Kyrkou, S. Timotheou and T. Theocharides. 'Convolutional Channel-Wise Competitive Learning for the Forward-Forward Algorithm'. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 38.13 (2024), pp. 14536–14544 (cit. on pp. 12, 66).

[92] L. Sun, Y. Zhang, J. Wen, L. Shen, W. Xie and W. He. 'DeeperForward: Enhanced Forward-Forward Training for Deeper and Better Performance'. In: *The Thirteenth International Conference on Learning Representations*. 2025 (cit. on pp. 12, 66).

[93] E. B. Terres-Escudero, J. D. Ser and P. G. Bringas. 'On the Improvement of Generalization and Stability of Forward-Only Learning via Neural Polarization'. In: *European Conference on Artificial Intelligence*. 2024 (cit. on p. 12).

[94] D. O. Hebb. *The organization of behavior: A neuropsychological theory*. Psychology press, 2005 (cit. on pp. 12, 64).

[95] K. D. Harris. 'Neural signatures of cell assembly organization'. In: *Nature reviews neuroscience* 6.5 (2005), pp. 399–407 (cit. on pp. 12, 60, 64).

[96] B. György. 'Neural syntax: cell assemblies, synapsembles, and readers'. In: *Neuron* 68.3 (2010), pp. 362–385 (cit. on pp. 12, 64).

[97] J. J. Hopfield. 'Neural networks and physical systems with emergent collective computational abilities.' In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558 (cit. on pp. 12, 64).

[98] L. Carrillo-Reid, S. Han, W. Yang, A. Akrouh and R. Yuste. 'Controlling visually guided behavior by holographic recalling of cortical ensembles'. In: *Cell* 178.2 (2019), pp. 447–457 (cit. on pp. 12, 13, 64).

[99] L. Carrillo-Reid and R. Yuste. 'Playing the piano with the cortex: role of neuronal ensembles and pattern completion in perception and behavior'. In: *Current opinion in neurobiology* 64 (2020), pp. 89–95 (cit. on pp. 12, 13, 64).

[100] L. Carrillo-Reid and V. Calderon. 'Conceptual framework for neuronal ensemble identification and manipulation related to behavior using calcium imaging'. In: *Neurophotonics* 9 (2022) (cit. on p. 12).

[101] M. Rigotti, O. Barak, M. R. Warden, X.-J. Wang, N. D. Daw, E. K. Miller and S. Fusi. 'The importance of mixed selectivity in complex cognitive tasks'. In: *Nature* 497.7451 (2013), pp. 585–590 (cit. on pp. 12, 65).

[102] S. Fusi, E. K. Miller and M. Rigotti. 'Why neurons mix: high dimensionality for higher cognition'. In: *Current opinion in neurobiology* 37 (2016), pp. 66–74 (cit. on pp. 12, 65).

[103] T. Yoshida and K. Ohki. 'Natural images are reliably represented by sparse and variable populations of neurons in visual cortex'. In: *Nature communications* 11.1 (2020), p. 872 (cit. on pp. 12, 60, 106).

[104] C. Dupre and R. Yuste. 'Non-overlapping neural networks in Hydra vulgaris'. In: *Current Biology* 27.8 (2017), pp. 1085–1097 (cit. on p. 13).

[105] J. Liu and S. C. Baraban. 'Network properties revealed during multi-scale calcium imaging of seizure activity in zebrafish'. In: *Eneuro* 6.1 (2019) (cit. on p. 13).

[106] R. Boyce, R. F. Dard and R. Cossart. 'Cortical neuronal assemblies coordinate with EEG microstate dynamics during resting wakefulness'. In: *Cell Reports* 42.2 (2023) (cit. on p. 13).

[107] J. E. Pérez-Ortega, T. Alejandre-García and R. Yuste. 'Long-term stability of cortical ensembles'. In: *Elife* 10 (2021), e64449 (cit. on p. 13).

[108] A. M. Packer, L. E. Russell, H. W. Dalgleish and M. Häusser. 'Simultaneous all-optical manipulation and recording of neural circuit activity with cellular resolution in vivo'. In: *Nature methods* 12.2 (2015), pp. 140–146 (cit. on p. 13).

[109] E. Doi and M. Lewicki. 'Sparse coding of natural images using an overcomplete set of limited capacity units'. In: *Advances in Neural Information Processing Systems* 17 (2004) (cit. on p. 13).

[110] D. J. Field. 'What is the goal of sensory coding?' In: *Neural Computation* 6.4 (1994), pp. 559–601 (cit. on p. 13).

[111] B. A. Olshausen and D. J. Field. 'Sparse coding of sensory inputs'. In: *Current Opinion in Neurobiology* 14.4 (2004), pp. 481–487 (cit. on pp. 13, 66).

[112] L. Bortolussi and G. Sanguinetti. *Intrinsic Geometric Vulnerability of High-Dimensional Artificial Intelligence*. 2018. arXiv: 1811.03571 (cit. on p. 15).

[113] Y. Qin, N. Carlini, I. Goodfellow, G. Cottrell and C. Raffel. 'Imperceptible, Robust, and Targeted Adversarial Examples for Automatic Speech Recognition'. In: *Proceedings of the International Conference on Machine Learning*. 2019 (cit. on p. 15).

[114] A. Kurakin, I. J. Goodfellow and S. Bengio. 'Adversarial Examples in the Physical World'. In: *Artificial Intelligence Safety and Security* (2018) (cit. on p. 15).

[115] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran and A. Madry. 'Adversarial Examples Are Not Bugs, They Are Features'. In: *Advances in Neural Information Processing Systems*. 2019 (cit. on p. 15).

[116] B. Biggio and F. Roli. 'Wild patterns: Ten years after the rise of adversarial machine learning'. In: *Pattern Recognition* 84 (2018), pp. 317–331 (cit. on p. 15).

[117] S.-M. Moosavi-Dezfooli, A. Fawzi and P. Frossard. 'DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks'. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016 (cit. on p. 15).

[118] F. Tramèr, N. Carlini, W. Brendel and A. Madry. 'On Adaptive Attacks to Adversarial Example Defenses'. In: *Advances in Neural Information Processing Systems*. 2020 (cit. on pp. 15, 16).

[119] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin and N. Usunier. 'Parseval Networks: Improving Robustness to Adversarial Examples'. In: *Proceedings of the International Conference on Machine Learning*. 2017 (cit. on p. 15).

[120] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh and P. McDaniel. 'Ensemble Adversarial Training: Attacks and Defenses'. In: *International Conference on Learning Representations*. 2018 (cit. on p. 15).

[121] G. Carbone, M. Wicker, L. Laurenti, A. Patanè, L. Bortolussi and G. Sanguinetti. 'Robustness of Bayesian Neural Networks to Gradient-Based Attacks'. In: *Advances in Neural Information Processing Systems*. 2020 (cit. on p. 15).

[122] M. Zhang, S. Levine and C. Finn. 'MEMO: Test Time Robustness via Adaptation and Augmentation'. In: *Advances in Neural Information Processing Systems*. 2022 (cit. on p. 15).

[123] F. Tramèr and D. Boneh. 'Adversarial Training and Robustness for Multiple Perturbations'. In: *Advances in Neural Information Processing Systems*. 2019 (cit. on p. 15).

[124] X. Jia, Y. Zhang, B. Wu, K. Ma, J. Wang and X. Cao. 'LAS-AT: Adversarial Training With Learnable Attack Strategy'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022 (cit. on p. 15).

[125] N. D. Singh, F. Croce and M. Hein. *Revisiting Adversarial Training for ImageNet: Architectures, Training and Generalization across Threat Models*. 2023. arXiv: 2303.01870 (cit. on p. 15).

[126] C.-W. Huang, J. H. Lim and A. C. Courville. 'A Variational Perspective on Diffusion-Based Generative Models and Score Matching'. In: *Advances in Neural Information Processing Systems*. 2021 (cit. on p. 16).

[127] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala. 'PyTorch: An Imperative Style, High-Performance Deep Learning Library'. In: *Advances in Neural Information Processing Systems*. 2019 (cit. on pp. 16, 33, 101).

[128] G. W. Ding, L. Wang and X. Jin. *AdverTorch vo.1: An Adversarial Robustness Toolbox based on PyTorch*. 2019. arXiv: 1902.07623 (cit. on p. 16).

[129] E. Ballarin. *ebtorch: Collection of PyTorch additions, extensions, utilities, uses and abuses*. https://github.com/emaballarin/ebtorch. 2025 (cit. on p. 16).

[130] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. 2009 (cit. on pp. 16, 22).

[131] P. Chrabaszcz, I. Loshchilov and F. Hutter. *A Downsampled Variant of ImageNet as an Alternative to the CIFAR datasets*. 2017. arXiv: 1707.08819 (cit. on pp. 16, 22).

[132] A. Athalye, N. Carlini and D. Wagner. 'Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples'. In: *Proceedings of the International Conference on Machine Learning*. 2018 (cit. on pp. 18, 26).

[133] A. Athalye, L. Engstrom, A. Ilyas and K. Kwok. 'Synthesizing Robust Adversarial Examples'. In: *Proceedings of the International Conference on Machine Learning*. 2018 (cit. on pp. 19, 24).

[134] T. M. Mitchell. *The Need for Biases in Learning Generalizations*. Tech. rep. New Brunswick, NJ: Rutgers University, 1980 (cit. on p. 20).

[135] H. Robbins and S. Monro. 'A Stochastic Approximation Method'. In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407 (cit. on p. 20).

[136] L. Bottou. 'On-Line Algorithms and Stochastic Approximations'. In: *On-Line Learning in Neural Networks*. Cambridge University Press, 1999. Chap. 2 (cit. on p. 20).

[137] S. Hashem. 'Optimal linear combinations of neural networks'. In: *Neural Networks* 10.4 (June 1997), pp. 599–614 (cit. on p. 21).

[138] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao and J. Han. 'On the Variance of the Adaptive Learning Rate and Beyond'. In: *International Conference on Learning Representations*. 2020 (cit. on pp. 23, 49).

[139] M. Zhang, J. Lucas, J. Ba and G. E. Hinton. 'Lookahead Optimizer: k steps forward, 1 step back'. In: *Advances in Neural Information Processing Systems*. 2019 (cit. on p. 23).

[140] L. N. Smith. 'Cyclical Learning Rates for Training Neural Networks'. In: *IEEE Winter Conference on Applications of Computer Vision*. 2017 (cit. on p. 23).

[141] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed and A. Lerchner. 'beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework'. In: *International Conference on Learning Representations*. 2017 (cit. on p. 23).

[142] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal and M. Hein. 'RobustBench: a standardized adversarial robustness benchmark'. In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021 (cit. on pp. 23, 93).

[143] F. Croce and M. Hein. *Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack*. 2020. arXiv: 1907.02044 (cit. on p. 24).

[144] M. Andriushchenko, F. Croce, N. Flammarion and M. Hein. 'Square Attack: a query-efficient black-box adversarial attack via random search'. In: *16th European Conference on Computer Vision*. 2020 (cit. on p. 24).

[145] G. Lin, Z. Tao, J. Zhang, T. Tanaka and Q. Zhao. *Robust Diffusion Models for Adversarial Purification*. 2024. arXiv: 2403.16067 (cit. on pp. 24, 25, 94).

[146] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry and A. Kurakin. *On Evaluating Adversarial Robustness*. 2019. arXiv: 1902.06705 (cit. on p. 26).

[147] H. M. Dolatabadi, S. Erfani and C. Leckie. 'ℓ_∞-Robustness and Beyond: Unleashing Efficient Adversarial Training'. In: *18th European Conference on Computer Vision*. 2022 (cit. on p. 27).

[148] C. Laidlaw, S. Singla and S. Feizi. 'Perceptual Adversarial Robustness: Defense Against Unseen Threat Models'. In: *International Conference on Learning Representations*. 2021 (cit. on p. 27).

[149] M. Boguná, R. Pastor-Satorras, A. Díaz-Guilera and A. Arenas. 'Models of social networks based on social distance attachment'. In: *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics* 70.5 (2004), p. 056122 (cit. on p. 29).

[150] S. P. Borgatti, A. Mehra, D. J. Brass and G. Labianca. 'Network analysis in the social sciences'. In: *science* 323.5916 (2009), pp. 892–895 (cit. on p. 29).

[151] R. N. Mantegna and H. E. Stanley. 'Scaling behaviour in the dynamics of an economic index'. In: *Nature* 376.6535 (1995), pp. 46–49 (cit. on p. 29).

[152] G. Bonanno, G. Caldarelli, F. Lillo, S. Miccichè, N. Vandewalle and R. N. Mantegna. 'Networks of equities in financial markets'. In: *The European Physical Journal B* 38 (2004), pp. 363–371 (cit. on p. 29).

[153] E. Davidson and M. Levin. 'Gene regulatory networks'. In: *Proceedings of the National Academy of Sciences* 102.14 (2005), pp. 4935–4935 (cit. on p. 29).

[154] H. Jeong, S. P. Mason, A.-L. Barabási and Z. N. Oltvai. 'Lethality and centrality in protein networks'. In: *Nature* 411.6833 (2001), pp. 41–42 (cit. on p. 29).

[155] P. Lodahl. 'Quantum-dot based photonic quantum networks'. In: *Quantum Science and Technology* 3.1 (2017), p. 013001 (cit. on p. 29).

[156] Q. Zhuang and B. Zhang. 'Quantum communication capacity transition of complex quantum networks'. In: *Physical Review A* 104.2 (2021), p. 022608 (cit. on p. 29).

[157] H. Kristjánsson, Y. Zhong, A. Munson and G. Chiribella. 'Quantum networks with coherent routing of information through multiple nodes'. In: *npj Quantum Information* 10.1 (2024), p. 131 (cit. on p. 29).

[158] L. Innocenti, S. Lorenzo, I. Palmisano, A. Ferraro, M. Paternostro and G. M. Palma. 'Potential and limitations of quantum extreme learning machines'. In: *Communications Physics* 6.1 (2023), p. 118 (cit. on p. 29).

[159] S. Apers, S. Chakraborty, L. Novo and J. Roland. 'Quadratic Speedup for Spatial Search by Continuous-Time Quantum Walk'. In: *Physical Review Letters* 129 (16 Oct. 2022), p. 160502 (cit. on p. 29).

[160] F. Caruso, A. W. Chin, A. Datta, S. F. Huelga and M. B. Plenio. 'Highly efficient energy excitation transfer in light-harvesting complexes: The fundamental role of noise-assisted transport'. In: *The Journal of Chemical Physics* 131.10 (2009) (cit. on pp. 29, 30).

[161] A. W. Chin, A. Datta, F. Caruso, S. F. Huelga and M. B. Plenio. 'Noise-assisted energy transfer in quantum networks and light-harvesting complexes'. In: *New Journal of Physics* 12.6 (June 2010), p. 065002 (cit. on pp. 29, 30).

[162] A. Olaya-Castro, C. F. Lee, F. F. Olsen and N. F. Johnson. 'Efficiency of energy transfer in a light-harvesting system under quantum coherence'. In: *Phys. Rev. B* 78 (8 Aug. 2008), p. 085115 (cit. on p. 29).

[163] D. A. Chisholm, G. García-Pérez, M. A. C. Rossi, G. M. Palma and S. Maniscalco. 'Stochastic collision model approach to transport phenomena in quantum networks'. In: *New Journal of Physics* 23.3 (Mar. 2021), p. 033031 (cit. on pp. 29, 30).

[164] M. Sokolov, D. S. Hoffmann, P. M. Dohmen, M. Krämer, S. Höfener, U. Kleinekathöfer and M. Elstner. 'Non-adiabatic molecular dynamics simulations provide new insights into the exciton transfer in the Fenna–Matthews–Olson complex'. In: *Physical Chemistry Chemical Physics* 26.28 (2024), pp. 19469–19496 (cit. on p. 29).

[165] H. Ó. Gestsson, C. Nation, J. S. Higgins, G. S. Engel and A. Olaya-Castro. 'Non-perturbative exciton transfer rate analysis of the Fenna-Matthews-Olson photosynthetic complex under reduced and oxidised conditions'. In: *arXiv preprint arXiv:2412.14883* (2024) (cit. on p. 29).

[166] F. Gallina, M. Bruschi and B. Fresch. 'Strategies to simulate dephasing-assisted quantum transport on digital quantum computers'. In: *New Journal of Physics* 24.2 (2022), p. 023039 (cit. on p. 29).

[167] S. Cavazzoni, L. Razzoli, P. Bordone and M. G. Paris. 'Perturbed graphs achieve unit transport efficiency without environmental noise'. In: *Physical Review E* 106.2 (2022), p. 024118 (cit. on p. 29).

[168] E. Annoni, M. Frigerio and M. G. Paris. 'Enhanced quantum transport in chiral quantum walks'. In: *Quantum Information Processing* 23.4 (2024), p. 117 (cit. on p. 29).

[169] S. Davidson, F. A. Pollock and E. Gauger. 'Principles underlying efficient exciton transport unveiled by information-geometric analysis'. In: *Phys. Rev. Res.* 3 (3 July 2021), p. L032001 (cit. on p. 30).

[170] C. Uchiyama, W. J. Munro and K. Nemoto. 'Environmental engineering for quantum energy transport'. In: *npj Quant. Inf.* 4 (2018), p. 33 (cit. on p. 30).

[171] A. Zwick, G. A. Álvarez, G. Bensky and G. Kurizki. 'Optimized dynamical control of state transfer through noisy spin chains'. In: *New Journal of Physics* 16 (2014), p. 065021 (cit. on p. 30).

[172] Q. Ai, Y.-J. Fan, B.-Y. Jin and Y.-C. Cheng. 'An efficient quantum jump method for coherent energy transfer dynamics in photosynthetic systems under the influence of laser fields'. In: *New Journal of Physics* 16 (2014), p. 053033 (cit. on p. 30).

[173] P. Xiang, M. Litinskaya, E. A. Shapiro and R. V. Krems. 'Non-adiabatic control of quantum energy transfer in ordered and disordered arrays'. In: *New Journal of Physics* 15 (2013), p. 063015 (cit. on p. 30).

[174] H. Dong, D.-Z. Xu, J.-F. Huang and C.-P. Sun. 'Coherent excitation transfer via the dark-state channel in a bionic system'. In: *Light: Science & Applications* 1.3 (2012), e2–e2 (cit. on p. 30).

[175] J. E. Devries Paul L. Hasburn. *A first course in computational physics*. New York: John Wiley and Sons, 2011 (cit. on p. 32).

[176] D. Kingma and J. Ba. 'Adam: A Method for Stochastic Optimization'. In: *Proceedings of the International Conference on Learning Representations* 3 (2015) (cit. on pp. 33, 101).

[177] V. Godbole, G. E. Dahl, J. Gilmer, C. J. Shallue and Z. Nado. 2023. URL: github.com/google-research/tuning_playbook (cit. on p. 33).

[178] G. S. Engel, T. R. Calhoun, E. L. Read, T.-K. Ahn, T. Mančal, Y.-C. Cheng, R. E. Blankenship and G. R. Fleming. 'Evidence for wavelike energy transfer through quantum coherence in photosynthetic systems'. In: *Nature* 446.7137 (2007), pp. 782–786 (cit. on pp. 33, 36).

[179] M. B. Plenio and S. F. Huelga. 'Dephasing assisted transport: Quantum networks and biomolecules'. In: *New Journal of Physics* 10.11 (2008), p. 113019 (cit. on p. 35).

[180] A. Ishizaki and G. R. Fleming. 'Unified treatment of quantum coherent and incoherent hopping dynamics in electronic energy transfer: Reduced hierarchy equation approach'. In: *The Journal of Chemical Physics* 130.23 (June 2009), p. 234111 (cit. on p. 36).

[181] B.-X. Wang, M.-J. Tao, Q. Ai, T. Xin, N. Lambert, D. Ruan, Y.-C. Cheng, F. Nori, F.-G. Deng and G.-L. Long. 'Efficient quantum simulation of photosynthetic light harvesting'. In: *NPJ Quantum Information* 4.1 (2018), p. 52 (cit. on p. 36).

[182] K. Khetarpal, M. Riemer, I. Rish and D. Precup. 'Towards Continual Reinforcement Learning: A Review and Perspectives'. In: *Journal of Artificial Intelligence Research* 75 (Dec. 2022), pp. 1401–1476 (cit. on p. 42).

[183] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro and Y. Zhang. *Sparks of Artificial General Intelligence: Early experiments with GPT-4.* 2023. arXiv: 2303.12712 (cit. on p. 42).

[184] A. J. Smola and B. Schölkopf. 'A tutorial on support vector regression'. In: *Statistics and Computing* 14.3 (2004), pp. 199–222 (cit. on p. 45).

[185] A. E. Hoerl and R. W. Kennard. 'Ridge Regression: Biased Estimation for Nonorthogonal Problems'. In: *Technometrics* 12.1 (1970), pp. 55–67 (cit. on p. 46).

[186] D. Zachariah, M. Sundin, M. Jansson and S. Chatterjee. 'Alternating Least-Squares for Low-Rank Matrix Reconstruction'. In: *IEEE Signal Processing Letters* 19.4 (Apr. 2012), pp. 231–234 (cit. on p. 46).

[187] T. Hastie, R. Mazumder, J. D. Lee and R. Zadeh. 'Matrix completion and low-rank SVD via fast alternating least squares'. In: *J. Mach. Learn. Res.* 16.1 (Jan. 2015), pp. 3367–3402 (cit. on p. 46).

[188] Garnier, Simon, Ross, Noam, Rudis, Robert, Camargo, A. Pedro, Sciaini, Marco, Scherer and Cédric. *viridis(Lite) - Colorblind-Friendly Color Maps for R.* 2024 (cit. on p. 50).

[189] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on pp. 53, 56).

[190] H. Xiao, K. Rasul and R. Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.* 2017. arXiv: 1708.07747 (cit. on pp. 56, 58, 83).

[191] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu and A. Y. Ng. 'Reading digits in natural images with unsupervised feature learning'. In: *Deep Learning and Unsupervised Feature Learning Workshop, NeurIPS* (2011) (cit. on p. 56).

[192] K. Alex. *Learning multiple layers of features from tiny images.* 2009 (cit. on p. 56).

[193] P. O. Hoyer. 'Non-negative matrix factorization with sparseness constraints.' In: *Journal of machine learning research* 5.9 (2004) (cit. on p. 57).

[194] D. C. Cireşan, U. Meier, L. M. Gambardella and J. Schmidhuber. 'Deep, Big, Simple Neural Nets for Handwritten Digit Recognition'. In: *Neural Computation* 22.12 (Dec. 2010), pp. 3207–3220 (cit. on p. 58).

[195] S. Pitsios. *SVHN Number Recognition using Deep Learning.* https://github.com/pitsios-s/SVHN. 2017 (cit. on p. 58).

[196] Z. Lin, R. Memisevic and K. Konda. 'How far can we go without convolution: Improving fully-connected networks'. In: *arXiv preprint arXiv:1511.02580* (2015) (cit. on p. 58).

[197] G. Georgiadis. 'Accelerating convolutional neural networks via activation map compression'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2019, pp. 7085–7095 (cit. on pp. 59, 107).

[198] V. Bonin, M. H. Histed, S. Yurgenson and R. C. Reid. 'Local diversity and fine-scale organization of receptive fields in mouse visual cortex'. In: *Journal of Neuroscience* 31.50 (2011), pp. 18506–18521 (cit. on p. 64).

[199] T. Hromádka, M. R. DeWeese and A. M. Zador. 'Sparse representation of sounds in the unanesthetized auditory cortex'. In: *PLoS biology* 6.1 (2008), e16 (cit. on p. 64).

[200] A. L. Barth and J. F. Poulet. 'Experimental evidence for sparse firing in the neocortex'. In: *Trends in neurosciences* 35.6 (2012), pp. 345–355 (cit. on p. 64).

[201] S. Leutgeb, J. K. Leutgeb, A. Treves, M.-B. Moser and E. I. Moser. 'Distinct ensemble codes in hippocampal areas CA3 and CA1'. In: *Science* 305.5688 (2004), pp. 1295–1298 (cit. on p. 64).

[202] B. M. Lake, R. Salakhutdinov and J. B. Tenenbaum. 'Human-level concept learning through probabilistic program induction'. In: *Science* 350.6266 (2015), pp. 1332–1338 (cit. on p. 65).

[203] W. Gerstner and W. M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002 (cit. on p. 66).

[204] G. Deco, A. Ponce-Alvarez, P. Hagmann, G. L. Romani, D. Mantini and M. Corbetta. 'How local excitation–inhibition ratio impacts the whole brain dynamics'. In: *Journal of Neuroscience* 34.23 (2014), pp. 7886–7898 (cit. on p. 66).

[205] B. Babadi and H. Sompolinsky. 'Sparseness and expansion in sensory representations'. In: *Neuron* 83.5 (2014), pp. 1213–1226 (cit. on p. 66).

[206] P. Mombaerts, F. Wang, C. Dulac, S. K. Chao, A. Nemes, M. Mendelsohn, J. Edmondson and R. Axel. 'Visualizing an olfactory sensory map'. In: *Cell* 87.4 (1996), pp. 675–686 (cit. on p. 66).

[207] G. C. Turner, M. Bazhenov and G. Laurent. 'Olfactory representations by Drosophila mushroom body neurons'. In: *Journal of neurophysiology* 99.2 (2008), pp. 734–746 (cit. on p. 66).

[208] R. F. Galán. 'On how network architecture determines the dominant patterns of spontaneous neural activity'. In: *PloS one* 3.5 (2008), e2148 (cit. on p. 66).

[209] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle and J. Guttag. 'What is the state of neural network pruning?' In: *Proceedings of machine learning and systems* 2 (2020), pp. 129–146 (cit. on p. 66).

[210] A. Shah, R. Hen, A. Losonczy and S. Fusi. 'Optimal sparsity in autoencoder memory models of the hippocampus'. In: *bioRxiv* (2025), pp. 2025–01 (cit. on p. 66).

[211] D. G. Barrett, A. S. Morcos and J. H. Macke. 'Analyzing biological and artificial neural networks: challenges with opportunities for synergy?' In: *Current opinion in neurobiology* 55 (2019), pp. 55–64 (cit. on p. 66).

[212] M. Schrimpf, J. Kubilius, H. Hong, N. J. Majaj, R. Rajalingham, E. B. Issa, K. Kar, P. Bashivan, J. Prescott-Roy, F. Geiger, K. Schmidt, D. L. K. Yamins and J. J. DiCarlo. 'Brain-Score: Which Artificial Neural Network for Object Recognition is most Brain-Like?' In: *bioRxiv* (2018) (cit. on p. 66).

[213] S. Amini, M. Teymoorianfard, S. Ma and A. Houmansadr. *MeanSparse: Post-Training Robustness Enhancement Through Mean-Centered Feature Sparsification*. 2024. arXiv: 2406.05927 (cit. on p. 94).

[214] Y. Bai, M. Zhou, V. M. Patel and S. Sojoudi. 'MixedNUTS: Training-Free Accuracy-Robustness Balance via Nonlinearly Mixed Classifiers'. In: *Transactions on Machine Learning Research*. 2024 (cit. on p. 94).

[215] Y. Bai, B. G. Anderson, A. Kim and S. Sojoudi. 'Improving the Accuracy-Robustness Trade-Off of Classifiers via Adaptive Smoothing'. In: *SIAM Journal on Mathematics of Data Science*. 2024 (cit. on p. 94).

[216] TorchVision maintainers and contributors. *TorchVision: PyTorch's Computer Vision library*. https://github.com/pytorch/vision. 2016 (cit. on p. 101).

[217] M. Raghu, J. Gilmer, J. Yosinski and J. Sohl-Dickstein. 'SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability'. In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 108).

[218] S. Kornblith, M. Norouzi, H. Lee and G. Hinton. 'Similarity of neural network representations revisited'. In: *International conference on machine learning*. Pmlr. 2019, pp. 3519–3529 (cit. on p. 108).

[219] G. J. Székely, M. L. Rizzo and N. K. Bakirov. 'Measuring and Testing Dependence by Correlation of Distances'. In: *The Annals of Statistics* (2007), pp. 2769–2794 (cit. on p. 108).

Appendix A

Supplementary Material for Chapter 3

A.1 Justification of Adversarially-balanced batches

During the incipient phases of experimentation, preliminary tests were performed with the MNIST [89] and Fashion-MNIST [190] datasets – using a conditional *VAE* as the *purifier*, and small FCNs or *convolutional ANN*s as the *classifiers*. Adversarial examples were generated against the adversarially pre-trained *classifier*, and tentatively denoised by the *purifier* with one sample only. The resulting recovered inputs were classified by the *classifier* and the overall accuracy was recorded.

Importantly, such tests were not meant to assess the *end-to-end* adversarial robustness of the whole architecture, but only to tune the training protocol of the *purifier*.

Generating adversarial training examples by means of PGD is considered the *gold standard* [51] and was first attempted as a natural choice to train the purifier. However, in this case, the following phenomena were observed:

- Unsatisfactory *clean* accuracy was reached upon convergence, speculatively a consequence of the *VAE* having never been trained on *clean*-to-*clean* example reconstruction;
- Persistent vulnerability to same norm-bound FGSM perturbations was noticed;
- Persistent vulnerability to smaller norm-bound FGSM and PGD perturbations was noticed.

In an attempt to mitigate such issues, the composition of adversarial examples was adjusted to specifically counteract each of the issues uncovered. The adoption of any smaller subset of attack types or strength, compared to that described in subsection 3.2.4, resulted in unsatisfactory mitigation.

At that point, another problem emerged: if such an adversarial training protocol was carried out in homogeneous batches, each containing the same type and strength of attack (or none at all), the resulting robust accuracy was still partially compromised due to the homogeneous ordering of attack types and strengths across batches.

Such observations lead to the final formulation of the training protocol, detailed in subsection 3.2.4, which mitigates to the best the issues described so far.

A.2 Architectural details and hyperparameters

In the following section, we provide more precise details about the architectures (subsection A.2.1) and hyperparameters (subsection A.2.2) used in the experimental phase of our work.

A.2.1 Architectures

In the following subsection, we describe the specific structure of the individual parts composing the *purifier* – in the three scenarios considered. As far as the *classifier* architectures are concerned, we redirect the reader to the original articles introducing those models (*i.e.*, those by Cui et al. [56] for *scenarios (a)* and *(b)*, Wang et al. [55] for *scenario (c)*).

During training, before being processed by the *purifier* encoder, input examples are standardised according to the statistics of the respective training dataset.

Afterwards, they are fed to the disjoint input encoder (see subsection 3.2.3), whose architecture is shown in Table A.1. The same architecture is used in all scenarios considered.

Table A.1: *Architecture for the disjoint input encoder of the purifier. The same architecture is used in all scenarios considered. The architecture is represented layer by layer, from input to output, in a PyTorch-like syntax. The following abbreviations are used:* `Conv2D`*: 2-dimensional convolutional layer;* `ch_in`*: number of input channels;* `ch_out`*: number of output channels;* `ks`*: kernel size;* `s`*: stride;* `p`*: padding;* `b`*: presence of a learnable bias term;* `BatchNorm2D`*: 2-dimensional batch normalisation layer;* `affine`*: presence of learnable affine transform coefficients;* `slope`*: slope for the activation function in the negative semi-domain.*

Disjoint Input Encoder (all scenarios)
`Conv2D(ch_in=3, ch_out=6, ks=3, s=2, p=1, b=False)`
`BatchNorm2D(affine=True)`
`LeakyReLU(slope=0.2)`
`Conv2D(ch_in=6, ch_out=12, ks=3, s=2, p=1, b=False)`
`BatchNorm2D(affine=True)`
`LeakyReLU(slope=0.2)`

The original input is also fed to the *classifier*. The corresponding internal representation is extracted, preserving its layered structure. In order to improve the scalability of the method, only a subset of *classifier* layers is used instead of the whole internal representation. Specifically, for each *block* of the WideResNet architecture, only the first layers have been considered; two *skip connections* have also been added for good measure. The exact list of those layers is reported in Table A.2.

Table A.2: *Classifier model (WideResNet-28-10) layer names used as (a subset of) the internal representation fed to the layerwise convolutional encoder of the purifier. The names reflect those used in the model implementation.*

All scenarios
`layer.0.block.0.conv_0`
`layer.0.block.0.conv_1`
`layer.0.block.1.conv_0`
`layer.0.block.1.conv_1`
`layer.0.block.2.conv_0`
`layer.0.block.2.conv_1`
`layer.0.block.3.conv_0`
`layer.0.block.3.conv_1`
`layer.1.block.0.conv_0`
`layer.1.block.0.conv_1`
`layer.1.block.0.shortcut`
`layer.1.block.1.conv_0`
`layer.1.block.1.conv_1`
`layer.1.block.2.conv_0`
`layer.1.block.2.conv_1`
`layer.1.block.3.conv_0`
`layer.1.block.3.conv_1`
`layer.2.block.0.conv_0`
`layer.2.block.0.conv_1`
`layer.2.block.0.shortcut`
`layer.2.block.1.conv_0`
`layer.2.block.1.conv_1`
`layer.2.block.2.conv_0`
`layer.2.block.2.conv_1`
`layer.2.block.3.conv_0`
`layer.2.block.3.conv_1`

Each extracted layerwise (pre)activation tensor has the shape of a multi-channel image, which is processed – independently for each layer – by a different CNN whose individual architecture is shown in Table A.3 (*scenarios (a)* and *(b)*) and Table A.4 (*scenario (c)*).

Table A.3: *Architecture for the layerwise internal representation encoder of the purifier. The architecture shown in this table is used in scenarios (a) and (b). The architecture is represented layer by layer, from input to output, in a PyTorch-like syntax. The following abbreviations are used:* `Conv2D`*: 2-dimensional convolutional layer;* `ch_in`*: number of input channels;* `ch_out`*: number of output channels;* `ks`*: kernel size;* `s`*: stride;* `p`*: padding;* `b`*: presence of a learnable bias term;* `BatchNorm2D`*: 2-dimensional batch normalisation layer;* `affine`*: presence of learnable affine transform coefficients;* `slope`*: slope for the activation function in the negative semi-domain. The abbreviation* `[ci]` *indicates the number of input channels for the (pre)activation tensor of each extracted layer. The abbreviation* `ceil` *indicates the ceiling integer rounding function.*

Layerwise Internal Representation Encoder (scenarios (a) and (b))

```
Conv2D(ch_in=[ci], ch_out=ceil([ci]/2), ks=3, s=1, p=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
Conv2D(ch_in=ceil([ci]/2), ch_out=ceil([ci]/4), ks=3, s=1, p=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
Conv2D(ch_in=ceil([ci]/4), ch_out=ceil([ci]/8), ks=3, s=1, p=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
```

The resulting tensors (still having the shape of multi-channel images) are then jointly processed by a fully-connected subnetwork whose architecture is shown in Table A.5. The value of `fcrepr` for the different scenarios considered is shown in Table A.10.

The *compressed input* and *compressed internal representation* so obtained are finally jointly encoded by an additional fully-connected subnetwork whose architecture is shown in Table A.6. The output is a tuple of means and standard deviations to be used to sample the stochastic latent code $\boldsymbol{z}$.

The sampler used for the generation of such latent variables $\boldsymbol{z}$, during the training of the *purifier*, is a reparameterised [75] Normal sampler $\boldsymbol{z} \sim \mathcal{N}(\mu, \sigma)$. During inference, $\boldsymbol{z}$ is sampled by reparameterisation from the *i.i.d* Standard Normal distribution $\boldsymbol{z} \sim \mathcal{N}(0, 1)$ (*i.e.* from its original prior).

The architectures for the decoder of the *purifier* are shown in Table A.7 (*scenarios (a)* and *(b)*) and Table A.8 (*scenario (c)*).

A.2.2 Hyperparameters

In the following section, we provide the hyperparameters used for *adversarial example generation* and *optimisation* during the training of the *purifier*, and those related to the

Table A.4: *Architecture for the layerwise internal representation encoder of the purifier. The architecture shown in this table is used in scenario (c). The architecture is represented layer by layer, from input to output, in a PyTorch-like syntax. The following abbreviations are used:* `Conv2D`*: 2-dimensional convolutional layer;* `ch_in`*: number of input channels;* `ch_out`*: number of output channels;* `ks`*: kernel size;* `s`*: stride;* `p`*: padding;* `b`*: presence of a learnable bias term;* `BatchNorm2D`*: 2-dimensional batch normalisation layer;* `affine`*: presence of learnable affine transform coefficients;* `slope`*: slope for the activation function in the negative semi-domain. The abbreviation* `[ci]` *indicates the number of input channels for the (pre)activation tensor of each extracted layer. The abbreviation* `ceil` *indicates the ceiling integer rounding function.*

Layerwise Internal Representation Encoder (scenario (c))

```
Conv2D(ch_in=[ci], ch_out=ceil([ci]/2), ks=3, s=1, p=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
Conv2D(ch_in=ceil([ci]/2), ch_out=ceil([ci]/4), ks=3, s=1, p=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
Conv2D(ch_in=ceil([ci]/4), ch_out=ceil([ci]/8), ks=3, s=1, p=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
Conv2D(ch_in=ceil([ci]/8), ch_out=ceil([ci]/16), ks=3, s=1, p=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
```

Table A.5: *Architecture for the fully-connected representation encoder of the purifier. The architecture shown in this table is used in all scenarios considered. The architecture is represented layer by layer, from input to output, in a PyTorch-like syntax. The following abbreviations are used:* `Concatenate`*: layer concatenating its input features;* `flatten_features`*: whether the input features are to be flattened before concatenation;* `feats_in`*,* `feats_out`*: number of input and output features of a linear layer;* `b`*: presence of a learnable bias term;* `BatchNorm1D`*: 1-dimensional batch normalisation layer;* `affine`*: presence of learnable affine transform coefficients;* `slope`*: slope for the activation function in the negative semi-domain. The abbreviation* `[computed]` *indicates that the number of features is computed according to the shape of the concatenated input tensors. The value of* `fcrepr` *for the different scenarios considered is shown in Table A.10.*

Fully-Connected Representation Encoder (all scenarios)

```
Concatenate(flatten_features=True)
Linear(feats_in=[computed], feats_out=fcrepr, b=False)
BatchNorm1D(affine=True)
LeakyReLU(slope=0.2)
```

Table A.6: *Architecture for the fully-connected joint encoder of the purifier. The architecture shown in this table is used in all scenarios considered. The architecture is represented layer by layer, from input to output, in a PyTorch-like syntax. The following abbreviations are used:* `Concatenate`*: layer concatenating its input features;* `flatten_features`*: whether the input features are to be flattened before concatenation;* `feats_in`*,* `feats_out`*: number of input and output features of a linear layer;* `b`*: presence of a learnable bias term;* `BatchNorm1D`*: 1-dimensional batch normalisation layer;* `affine`*: presence of learnable affine transform coefficients;* `slope`*: slope for the activation function in the negative semi-domain. The abbreviation* `[computed]` *indicates that the number of features is computed according to the shape of the concatenated input tensors. The value of* `fjoint` *for the different scenarios considered is shown in Table A.10. The last layer of the network returns a tuple of 2 tensors, each independently processed – from the output of the previous layer – by the two comma-separated sub-layers.*

Fully-Connected Joint Encoder (all scenarios)

```
Concatenate(flatten_features=True)
Linear(feats_in=[computed], feats_out=fjoint, b=False)
BatchNorm1D(affine=True)
LeakyReLU(slope=0.2)
( Linear(feats_in=fjoint, feats_out=fjoint, b=True),
       Linear(feats_in=fjoint, feats_out=fjoint, b=True) )
```

purifier model architectures. We also provide the hyperparameters for the PGD+EoT attack, which is used as a complementary tool for the evaluation of adversarial robustness.

Attacks

The hyperparameters used for the adversarial attacks described in subsection 3.2.4 are shown in Table A.9. The value of ϵ_∞ is fixed to $\epsilon_\infty = 8/255$. With the only exception of ϵ_∞, AutoAttack is to be considered a *hyperparameter-free* adversarial example generator.

Architectures

Table A.10 contains the hyperparameters that define the model architectures used as part of the *purifier*, in the different scenarios considered.

Training

Table A.11 collects the hyperparameters governing the training of the *purifier* in the different scenarios considered.

A.3 Ablation study on the need for adversarial training

In order to determine whether it is necessary to train on adversarial examples each of the constituent parts of Carso, an ablation study is performed. The architecture of Carso

Table A.7: *Architecture for the decoder of the purifier. The architecture shown in this table is used in scenarios (a) and (b). The architecture is represented layer by layer, from input to output, in a PyTorch-like syntax. The following abbreviations are used:* `Concatenate` *: layer concatenating its input features;* `flatten_features` *: whether the input features are to be flattened before concatenation;* `feats_in`*,* `feats_out` *: number of input and output features of a linear layer;* `b` *: presence of a learnable bias term;* `ConvTranspose2D` *: 2-dimensional transposed convolutional layer;* `ch_in` *: number of input channels;* `ch_out` *: number of output channels;* `ks` *: kernel size;* `s` *: stride;* `p` *: padding;* `op` *: PyTorch parameter '*`output padding`*', used to disambiguate the number of spatial dimensions of the resulting output;* `b` *: presence of a learnable bias term;* `BatchNorm2D` *: 2-dimensional batch normalisation layer;* `affine` *: presence of learnable affine transform coefficients;* `slope` *: slope for the activation function in the negative semi-domain. The values of* `fjoint` *and* `fcrepr` *for the different scenarios considered are shown in Table A.10.*

Decoder (scenarios (a) and (b))

```
Concatenate(flatten_features=True)
Linear(feats_in=[fjoint+fcrepr], feats_out=2304, b=True)
LeakyReLU(slope=0.2)
Unflatten(256, 3, 3)
ConvTranspose2D(ch_in=256, ch_out=256, ks=3, s=2, p=1, op=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
ConvTranspose2D(ch_in=256, ch_out=128, ks=3, s=2, p=1, op=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
ConvTranspose2D(ch_in=128, ch_out=64, ks=3, s=2, p=1, op=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
ConvTranspose2D(ch_in=64, ch_out=32, ks=3, s=2, p=1, op=0, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
ConvTranspose2D(ch_in=32, ch_out=3, ks=2, s=1, p=1, op=0, b=True)
Sigmoid()
```

Table A.8: *Architecture for the decoder of the purifier. The architecture shown in this table is used in scenario (c). The architecture is represented layer by layer, from input to output, in a PyTorch-like syntax. The following abbreviations are used:* `Concatenate`*: layer concatenating its input features;* `flatten_features`*: whether the input features are to be flattened before concatenation;* `feats_in`*,* `feats_out`*: number of input and output features of a linear layer;* `b`*: presence of a learnable bias term;* `ConvTranspose2D`*: 2-dimensional transposed convolutional layer;* `ch_in`*: number of input channels;* `ch_out`*: number of output channels;* `ks`*: kernel size;* `s`*: stride;* `p`*: padding;* `op`*: PyTorch parameter '*`output padding`*', used to disambiguate the number of spatial dimensions of the resulting output;* `b`*: presence of a learnable bias term;* `BatchNorm2D`*: 2-dimensional batch normalisation layer;* `affine`*: presence of learnable affine transform coefficients;* `slope`*: slope for the activation function in the negative semi-domain. The values of* `fjoint` *and* `fcrepr` *for the different scenarios considered are shown in Table A.10.*

Decoder (scenario (c))

```
Concatenate(flatten_features=True)
Linear(feats_in=[fjoint+fcrepr], feats_out=4096, b=True)
LeakyReLU(slope=0.2)
Unflatten(256, 4, 4)
ConvTranspose2D(ch_in=256, ch_out=256, ks=3, s=2, p=1, op=1, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
ConvTranspose2D(ch_in=256, ch_out=128, ks=3, s=2, p=1, op=1, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
ConvTranspose2D(ch_in=128, ch_out=64, ks=3, s=2, p=1, op=1, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
ConvTranspose2D(ch_in=64, ch_out=32, ks=3, s=2, p=1, op=1, b=False)
BatchNorm2D(affine=True)
LeakyReLU(slope=0.2)
ConvTranspose2D(ch_in=32, ch_out=3, ks=3, s=1, p=1, op=0, b=True)
Sigmoid()
```

Table A.9: *Hyperparameters for the attacks used for training and testing the purifier The* Fgsm *and* Pdg *attacks refer to the training phase (see subsection 3.2.4), whereas the* Pgd+EoT *attack [74] refers to the robustness assessment pipeline. The entry* `CCE` *denotes the Categorical CrossEntropy loss function. The ℓ_∞ threat model is assumed, and all inputs are linearly rescaled within $[0.0, 1.0]$ before the attack.*

	Fgsm	Pgd	Pgd+EoT
Input clipping	$[0.0, 1.0]$	$[0.0, 1.0]$	$[0.0, 1.0]$
# of steps	1	40	200
Step size	ϵ_∞	0.01	0.007
Loss function	*CCE*	*CCE*	*CCE*
# of EoT iterations	1	1	20
Optimiser		*SGD*	*SGD*

Table A.10: *Scenario-specific architectural hyperparameters for the purifier, as referred to in Table A.5, Table A.6, Table A.7, and Table A.8.*

	Scenario (a)	*Scenario (b)*	*Scenario (c)*
`fcrepr`	512	512	768
`fjoint`	128	128	192

Table A.11: *Hyperparameters used for training the purifier, grouped by scenario. The entry CCE denotes the Categorical CrossEntropy loss function. The LR scheduler is stepped after each epoch.*

	All *scenarios*	*Sc. (a)*	*Sc. (b)*	*Sc. (c)*
Optimiser	RAdam+Lookahead			
RAdam β_1	0.9			
RAdam β_2	0.999			
RAdam ϵ	10^{-8}			
RAdam *Weight Decay*	None			
Lookahead *averaging decay*	0.8			
Lookahead steps	6			
Initial *LR*	5×10^{-9}			
Loss function	*CCE*			
Sampled reconstructions per input	8			
Epochs		200	200	250
LR warm-up epochs		25	25	31
LR plateau epochs		25	25	31
LR annealing epochs		150	150	188
Plateau *LR*		0.064	0.064	0.0128
Final *LR*		4.346×10^{-4}	4.346×10^{-4}	1.378×10^{-4}
β *increase* initial epoch		25	25	32
β *increase* final epoch		34	34	43
Batch size		5120	2560	1024
Adversarial *batch fraction*		0.5	0.15	0.01

provided in section 3.3.1 is compared in terms of *clean* and *robust* accuracy with those ablated as follows. A WideResNet-28-10 model is always used as the *classifier*.

- Both the initial instance of the *classifier* (that used to extract the *internal representation*) and the final (that used to actually perform classification) are trained on clean examples only.
- The final *classifier* is trained on clean examples only, whereas the former is adversarially-trained.
- The initial *classifier* is trained on clean examples only, whereas the latter is adversarially-trained.

Results of such comparison are shown in Table A.12.

Table A.12: *Results of the ablation study on the architecture of Carso. The Clean Acc. column shows the test-set accuracy on uncorrupted inputs for the specific ablated model; the randAA Acc. column shows the accuracy of the same model on test-set inputs perturbed by means of the version of AutoAttack suitable for stochastic defences. In the Type of ablation column, any entry different from* `None` *(original architecture) indicates the type of training used for the first (before the solidus) and the second (after the solidus) classifier following input-to-output flow, within the ablated Carso architecture.*

Dataset	Type of ablation	Clean Acc.	randAA Acc.
Cifar-10	*clean/clean*	*0.7314*	0.7070
	AT/clean	*0.6743*	$<$ 0.7070
	clean/AT	***0.8892***	**0.7975**
	None	*0.8686*	0.7613
Cifar-100	*clean/clean*	*0.4395*	0.4032
	AT/clean	*0.4373*	$<$ 0.4032
	clean/AT	***0.6876***	**0.6716**
	None	*0.6806*	0.6665
TinyImageNet-200	*clean/AT*	***0.5677***	0.5281
	None	*0.5632*	**0.5356**

As it is possible to see, only the *clean/AT* ablation provides *clean* and *adversarial* accuracies comparable to that of the original Carso architecture – and indeed, it even determines an improvement on the Cifar-10 and Cifar-100 datasets. On the other hand, adversarial training of the former instance of the classifier is necessary to achieve the best robustness results on TinyImageNet-200.

Keeping in mind that the TinyImageNet-200 dataset is the closest representative considered for larger-scale datasets, and that empirical *robust accuracy* results only constitute an upper bound of maximally attainable robust accuracy, we support the original Carso architecture as the most effective for the achievement of adversarial robustness on generic image classification datasets – even if at the cost of a slight *clean* accuracy penalty. Since an adversarially-trained classifier would be used nonetheless as the latter, this does not incur in increased training-time computational intensity.

A.4 Further results and comparisons

The following section contains additional results and comparisons, in the form of tabular data, that may be of interest to the reader.

In particular, Table A.14 compares the *clean* and *robust* accuracy of Carso with those of the top-5 adversarial training defences according to the RobustBench [142] leaderboard[1] and the top-5 (if available) purification-based defences according to Lee and Kim [74], for Cifar-10/Cifar-100.

On shared columns, Table 3.2 can be considered a subset of Table A.14. As discussed in subsection 3.3.2, Carso tops the comparison in terms of adversarial accuracy, while maintaining a clean accuracy comparable with that of some purification-based models.

Table A.13, instead, investigates the same comparison across Carso and its *classifier* model, for adversarially-pretrained networks different from those described in section 3.3.1 – in particular, worse-performing. As it is possible to see, the increase in *end-to-end* adversarial accuracy determined by Carso does not depend in absolute terms on the quality of the original *classifier* model employed.

Table A.13: *Clean (results in italic) and adversarial (results in upright) accuracy for additional models to those described in subsection 3.3.2. The following abbreviations are used:* `AT/Cl`*: clean accuracy for the adversarially-pretrained model used as the classifier, when considered alone;* `C/Cl`*: clean accuracy for the Carso architecture;* `AT/AA`*: robust accuracy (by the means of AutoAttack) for the adversarially-pretrained model used as the classifier, when considered alone;* `C/randAA`*: robust accuracy for the Carso architecture, when attacked end-to-end by AutoAttack for randomised defences.*

Dataset	Classification model	AT/Cl	AT/AA	C/Cl	C/randAA
Cifar-10	Rebuffi et al. [54]	*0.8733*	0.6075	*0.8152*	0.7070
	Gowal et al. [51]	*0.8948*	0.6280	*0.8361*	0.7335
Cifar-100	Rebuffi et al. [54]	*0.6241*	0.3206	*0.5723*	0.5580

[1] At the time of chapter review, Git SHA hash: `78fcc9e48a07a861268f295a777b975f25155964`.

Table A.14: *Clean (results in italic) and adversarial (results in upright) accuracy for state-of-the-art adversarial defences, compared to Carso. The Robust Acc. column shows the best (i.e. lowest practically achieved) accuracy on test-set adversarial inputs, as obtained by either the original publication introducing the method, evaluation by AutoAttack as shown on the RobustBench leaderboard, evaluation of bespoke adaptive attacks as shown on the RobustBench leaderboard, or (for purification methods) evaluation by PGD+EoT from Lee and Kim [74]. The Def. Type column indicates whether the defence is based on adversarial training (AT), purification (P), or both (AT+P). Results for Carso are the same as for Table 3.2.*

Dataset	Model	Architecture	Clean Acc.	Robust Acc.	Def. type
Cifar-10	Bartoldson et al. [58]	WideResNet-94-16	*0.9368*	0.7371	AT
	Amini et al. [213]	MeanSparse WideResNet-94-16	***0.9568***	0.7310	AT
	Peng et al. [57]	RaWideResNet-70-16	*0.9311*	0.7107	AT
	Wang et al. [55]	WideResNet-70-16	*0.9325*	0.7069	AT
	Bai et al. [214]	ResNet-152 + WideResNet-70-16	*0.9519*	0.6971	AT
	Lin et al. [145]	Diffusion-based	*0.9082*	0.6641	P
	Lee and Kim [74]	Diffusion-based	*0.9053*	0.5688	P
	Nie et al. [68]	Diffusion-based	*0.9043*	0.5113	P
	Hill, Mitchell and Zhu [66]	Energy-based	*0.8412*	0.5490	P
	Yoon, Hwang and Lee [72]	Diffusion-based	*0.8612*	0.3711	P
	Ours	Carso (WideResNet-28-10)	*0.8686*	**0.7613**	AT+P
Cifar-100	Wang et al. [55]	WideResNet-70-16	*0.7522*	0.4266	AT
	Amini et al. [213]	MeanSparse WideResNet-70-16	*0.7513*	0.4225	AT
	Bai et al. [214]	ResNet-152 + WideResNet-70-16	*0.8308*	0.4180	AT
	Cui et al. [56]	WideResNet-28-10	*0.7385*	0.3918	AT
	Bai et al. [215]	ResNet-152 + WideResNet-70-16 + Mixing Net	***0.8521***	0.3872	AT
	Lin et al. [145]	Diffusion-based	*0.6973*	0.4609	P
	Ours	Carso (WideResNet-28-10)	*0.6806*	**0.6665**	AT+P

Appendix B

Supplementary Material for Chapter 4

B.1 Appendix 1: Additional case study

In this Appendix, we present findings on additional cases that were not explored in depth in the main text; they all refer to the off-resonant case when $\omega_r = 15$.

We begin by showing that, when considering the NN network, there is essentially no advantage in using many driving parameters ($R = 2, 7$) compared to the simplest case ($R = 1$), as illustrated in Figure B.1(A).

Next, we examine the stability of our results across networks of different sizes. Specifically, we analyse networks with $N = 4, 6, 8$ sites, respectively, and compare the improvement in the probability of reaching the sink when driving is applied. Figure B.1(B) shows that, although the effectiveness decreases slightly for larger networks, the improvements remain significant in all three cases.

Finally, in Figure B.1(C) we assess the robustness of our approach when decoherence effects increase by one order of magnitude compared to the case studied in the main text (see Figure 4.2(A)). The comparison reveals that this increase in decoherence does not affect the conclusions of our analysis; specifically, for the NN network of size $N = 4$, learning the couplings or adding driving are the most effective strategies to increase the probability that the excitation reaches the sink.

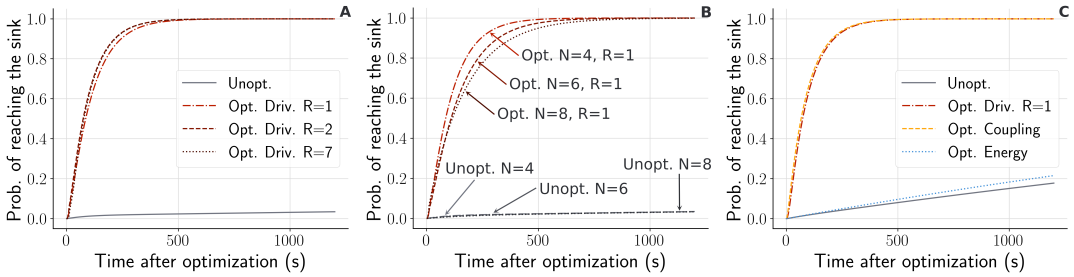

Figure B.1: *For all plots we are in the off-resonant case $\omega_r = 15$. (A) Probability of reaching the sink for a NN network with $N = 4$ sites unoptimised (grey continuous line) and when drivings increasingly complex are learnt with $R = 1, 2, 7$ in see Eqs. (4.2) and (4.2). (B) Study of the effectiveness of external driving with $R = 1$ for different size of the network $N = 4, 6, 8$. (C) Probability of reaching the sink for a NN network with $N = 4$ sites unoptimised, optimising drivings, couplings or site energies with $\lambda_N = 1$ one order of magnitude larger than the one considered in Figure 4.2 (A).*

Appendix C

Supplementary Material for Chapter 5

C.1 On the limitations of invertible linear latent dynamics in the presence of absorbing states

In the following section, we consider a much simplified version of the *state-space* and dynamical evolution rules described in subsection 5.3.1. In particular, we limit ourselves to only one *row* of the chessboard with *sticky edges*, no additional scalar f_j appended to chessboard states, no partitioned observability, and only two opposite tasks along the chessboard row: a rightward and a leftward moves. In such setting it is possible to prove that there do not simultaneously exist an invertible encoder α and an affine operator T per task acting in latent space (akin to those usually employed in Koopman operator learning, or to those described in subsection 5.2.2 in the absence of φ) able to correctly model both tasks. The setting is formalised in subsection C.1.1. The result is first proved for a 1-dimensional latent space in subsection C.1.2, and then generalised in subsection C.1.3 to the case of a low-dimensional latent space *w.r.t.* the number of states.

C.1.1 Setup and Notation

Consider a discrete *state-space* of n distinct positions. Each position $i \in \{0, \dots, n-1\}$ is represented by a *one-hot* vector $\boldsymbol{x}_i = \boldsymbol{e}_i \in \{0,1\}^n$, where $\boldsymbol{e}_i$ denotes the i-th standard basis vector for $\mathbb{R}^n$.

An encoder $\alpha : \mathbb{R}^n \to \mathbb{R}^{d_k}$ maps each state to a latent code:

$$\boldsymbol{z}_i = \alpha(\boldsymbol{x}_i).$$

For the encoding to be invertible (*i.e.*, for a decoder $\beta : \mathbb{R}^{d_k} \to \mathbb{R}^n$ to recover the original state), all latent codes must be distinct: $\boldsymbol{z}_i \neq \boldsymbol{z}_{i'}$ for $i \neq i'$.

Two *dynamical tasks* are defined on the state-space:

- **Rightward shift**: position i transitions to $\min(i+1, n-1)$,
- **Leftward shift**: position i transitions to $\max(i-1, 0)$.

Both tasks feature an *absorbing boundary*: once the rightmost (or leftmost, respectively) position is reached, the state remains the same.

We investigate whether there exist affine operators:

$$T_r(\boldsymbol{z}) = A_r\boldsymbol{z} + \boldsymbol{b}_r, \qquad T_l(\boldsymbol{z}) = A_l\boldsymbol{z} + \boldsymbol{b}_l,$$

with $A_r, A_l \in \mathbb{R}^{d_k \times d_k}$ and $\boldsymbol{b}_r, \boldsymbol{b}_l \in \mathbb{R}^{d_k}$, such that the latent dynamics exactly implement both tasks:

$$T_r(\boldsymbol{z}_i) = \boldsymbol{z}_{i+1} \quad \text{for } i < n-1, \qquad T_r(\boldsymbol{z}_{n-1}) = \boldsymbol{z}_{n-1}, \tag{C.1}$$

$$T_l(\boldsymbol{z}_i) = \boldsymbol{z}_{i-1} \quad \text{for } i > 0, \qquad T_l(\boldsymbol{z}_0) = \boldsymbol{z}_0. \tag{C.2}$$

Remark. ***Fixed points of affine maps***

An affine map $T(\boldsymbol{z}) = A\boldsymbol{z} + \boldsymbol{b}$ has fixed points satisfying:

$$\boldsymbol{z} = A\boldsymbol{z} + \boldsymbol{b} \quad \Longleftrightarrow \quad (I - A)\boldsymbol{z} = \boldsymbol{b}.$$

If $I - A$ is invertible, the unique fixed point is:

$$\boldsymbol{z}^* = (I - A)^{-1}\boldsymbol{b},$$

which can be any point in $\mathbb{R}^{d_k}$ depending on the choice of A and $\boldsymbol{b}$. This is in contrast to linear maps ($\boldsymbol{b} = \boldsymbol{0}$), whose only fixed point is the origin $\boldsymbol{0}$ (unless A has eigenvalue 1).

C.1.2 Case I: Scalar Latent Space ($d_k = 1$)

Proposition 1. *There exist no encoder $\alpha : \mathbb{R}^n \to \mathbb{R}$ and affine operators $T_r(z) = a_r z + b_r$, $T_l(z) = a_l z + b_l$ satisfying the dynamical requirements* (C.1) *and* (C.2) *with distinct latent codes.*

Proof. The proof proceeds in three steps.

Step 1: Interior invertibility. For interior states $i \in \{1, \dots, n-2\}$, applying the rightward operator followed by the leftward operator must return to the original state:

$$T_l(T_r(z_i)) = z_i.$$

Expanding:

$$a_l(a_r z_i + b_r) + b_l = a_l a_r z_i + a_l b_r + b_l = z_i.$$

Since this identity must hold for at least $n - 2 \geq 2$ distinct values of z_i (assuming $n \geq 4$), we require:

$$a_l a_r = 1 \quad \text{and} \quad a_l b_r + b_l = 0. \tag{C.3}$$

Hence $a_l = a_r^{-1}$ and $b_l = -a_r^{-1} b_r$.

Step 2: Non-triviality. Suppose $a_r = 1$. Then $T_r(z) = z + b_r$, and the fixed point condition $T_r(z_{n-1}) = z_{n-1}$ implies:

$$z_{n-1} + b_r = z_{n-1} \quad \Longrightarrow \quad b_r = 0.$$

But then T_r is the identity map, so $z_{i+1} = T_r(z_i) = z_i \forall i$, contradicting the distinctness of the latent codes. An analogous argument excludes $a_l = 1$.

Therefore $a_r \neq 1$ and $a_l \neq 1$, ensuring that both operators have well-defined unique fixed points.

Step 3: Fixed point coincidence. The absorbing boundary conditions require that z_{n-1} is the fixed point of T_r and z_0 is the fixed point of T_l:

$$z_{n-1} = \frac{b_r}{1 - a_r}, \qquad z_0 = \frac{b_l}{1 - a_l}.$$

Substituting the constraints from (C.3):

$$z_0 = \frac{-a_r^{-1} b_r}{1 - a_r^{-1}} = \frac{-a_r^{-1} b_r}{(a_r - 1)/a_r} = \frac{-b_r}{a_r - 1} = \frac{b_r}{1 - a_r} = z_{n-1}.$$

This contradicts the requirement that $z_0 \neq z_{n-1}$. □

C.1.3 Case II: Low-Dimensional Latent Space ($d_k \ll n$)

Proposition 2. *There exists no encoder $\alpha : \mathbb{R}^n \to \mathbb{R}^{d_k}$ with $d_k \ll n$ and affine operators T_r, T_l satisfying the dynamical requirements* (C.1) *and* (C.2) *with distinct latent codes.*

Proof. The proof generalises the scalar case through three steps.

Step 1: Affine recurrence and Krylov subspace constraint. The interior rightward dynamics $\boldsymbol{z}_{i+1} = A_r \boldsymbol{z}_i + \boldsymbol{b}_r$ defines an affine recurrence. Unrolling:

$$\boldsymbol{z}_i = A_r^i \boldsymbol{z}_0 + \sum_{j=0}^{i-1} A_r^j \boldsymbol{b}_r.$$

Let $\boldsymbol{c} = (I - A_r)^{-1} \boldsymbol{b}_r$ denote the fixed point of T_r (assuming $I - A_r$ is invertible; the singular case is addressed in *Step 2*). Define centred coordinates $\tilde{\boldsymbol{z}}_i = \boldsymbol{z}_i - \boldsymbol{c}$. Then:

$$\tilde{\boldsymbol{z}}_{i+1} = \boldsymbol{z}_{i+1} - \boldsymbol{c} = A_r \boldsymbol{z}_i + \boldsymbol{b}_r - \boldsymbol{c} = A_r(\boldsymbol{z}_i - \boldsymbol{c}) + (\boldsymbol{b}_r - \boldsymbol{c} + A_r \boldsymbol{c}) = A_r \tilde{\boldsymbol{z}}_i,$$

where we used $\boldsymbol{c} = A_r \boldsymbol{c} + \boldsymbol{b}_r$. Hence $\tilde{\boldsymbol{z}}_i = A_r^i \tilde{\boldsymbol{z}}_0$, and the centred dynamics are purely linear.

By the Cayley–Hamilton theorem, every matrix satisfies its own characteristic polynomial. For $A_r \in \mathbb{R}^{d_k \times d_k}$:

$$A_r^{d_k} = c_{d_k - 1} A_r^{d_k - 1} + c_{d_k - 2} A_r^{d_k - 2} + \cdots + c_0 I$$

for some coefficients $c_0, \ldots, c_{d_k - 1} \in \mathbb{R}$. Consequently, for $i \geq d_k$:

$$\tilde{\boldsymbol{z}}_i = A_r^i \tilde{\boldsymbol{z}}_0 = \sum_{j=0}^{d_k - 1} \gamma_j A_r^j \tilde{\boldsymbol{z}}_0 = \sum_{j=0}^{d_k - 1} \gamma_j \tilde{\boldsymbol{z}}_j$$

for some coefficients γ_j. The sequence $\{\tilde{\boldsymbol{z}}_i\}_{i=0}^{n-1}$ therefore lies in the Krylov subspace

$$\mathcal{K}_d(\tilde{\boldsymbol{z}}_0, A_r) = \operatorname{span}\{\tilde{\boldsymbol{z}}_0, A_r\tilde{\boldsymbol{z}}_0, \ldots, A_r^{d_k-1}\tilde{\boldsymbol{z}}_0\},$$

which has dimension at most d_k. Since $\boldsymbol{z_i} = \tilde{\boldsymbol{z}}_i + \boldsymbol{c}$, the original latent codes lie in a d_k-dimensional affine subspace of $\mathbb{R}^d$.

Step 2: Interior invertibility. For interior states $i \in \{1, \ldots, n-2\}$, the composition $T_l \circ T_r$ must act as the identity:

$$T_l(T_r(\boldsymbol{z_i})) = A_l(A_r\boldsymbol{z_i} + \boldsymbol{b}_r) + \boldsymbol{b}_l = A_lA_r\boldsymbol{z_i} + A_l\boldsymbol{b}_r + \boldsymbol{b}_l = \boldsymbol{z_i}.$$

This requires:

$$A_lA_r\boldsymbol{z_i} = \boldsymbol{z_i} \quad \text{and} \quad A_l\boldsymbol{b}_r + \boldsymbol{b}_l = 0 \qquad \forall i \in \{1, \ldots, n-2\}. \tag{C.4}$$

When $d_k \ll n$, there are $n - 2 \gg d_k$ interior states. By the Krylov subspace constraint, the latent codes $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{n-2}\}$ span a subspace of dimension at most d. Generically, they span exactly $\mathbb{R}^d$. Therefore, the condition $A_lA_r\boldsymbol{z} = \boldsymbol{z}$ for all $\boldsymbol{z}$ in this spanning set implies:

$$A_lA_r = I,$$

i.e., $A_l = A_r^{-1}$. The translation constraint then gives $\boldsymbol{b}_l = -A_r^{-1}\boldsymbol{b}_r$.

Step 3: Fixed point coincidence. The absorbing boundary conditions require that $\boldsymbol{z}_{n-1}$ is a fixed point of T_r and $\boldsymbol{z}_0$ is a fixed point of T_l:

$$\boldsymbol{z}_{n-1} = (I - A_r)^{-1}\boldsymbol{b}_r, \qquad \boldsymbol{z}_0 = (I - A_l)^{-1}\boldsymbol{b}_l.$$

Substituting $A_l = A_r^{-1}$ and $\boldsymbol{b}_l = -A_r^{-1}\boldsymbol{b}_r$:

$$\boldsymbol{z}_0 = (I - A_r^{-1})^{-1}(-A_r^{-1}\boldsymbol{b}_r).$$

To simplify $(I - A_r^{-1})^{-1}$, note that:

$$I - A_r^{-1} = A_r^{-1}(A_r - I),$$

hence:

$$(I - A_r^{-1})^{-1} = (A_r - I)^{-1}A_r.$$

Substituting:

$$\boldsymbol{z}_0 = (A_r - I)^{-1}A_r(-A_r^{-1}\boldsymbol{b}_r) = -(A_r - I)^{-1}\boldsymbol{b}_r = (I - A_r)^{-1}\boldsymbol{b}_r = \boldsymbol{z}_{n-1}.$$

This contradicts the requirement that $\boldsymbol{z}_0 \neq \boldsymbol{z}_{n-1}$. □

Appendix D

Supplementary Material for Chapter 6

D.1 Data

The MNIST dataset consists of pictures of handwritten Arabic numerals, from 0 to 9, each represented as a grayscale image of size 28×28. FASHIONMNIST has been designed as a drop-in replacement to MNIST, offering a more challenging classification task. It consists of ten classes of clothing items, still represented as grayscale images with a resolution of 28×28. Both datasets provide 60000 training and 10000 test images, balanced in terms of per-class numerosity.
SVHN contains coloured images of digits from house numbers, captured by Google StreetView. The images are composed of 32×32 RGB-encoded pixels. This dataset is slightly larger than the previous two, as it contains 73257 data-points in the training set and 26032 in the test set.
The SVHN images have been cropped in order to center the digit of interest within the frame. However, the presence of adjacent digits and other distracting elements, that have been kept within the images, introduces an additional layer of complexity when compared to MNIST and FASHIONMNIST, where the subjects are prominently displayed against a uniform black background. The CIFAR-10 consists of 60000 coloured natural images categorised in 10 balanced classes. The dataset is split in 50000 training images and 10000 test images. Each image, like SVHN has a resolution of 32×32 for each channel. Compared to previous datasets, this is the most challenging one for a fully connected network. The dataset split employed is provided by the TORCHVISION framework [216].

D.2 Training details

All our models (**FF**, **BP/FF** and **BP**), on all datasets (MNIST, FASHIONMNIST, SVHN and CIFAR-10), have been optimised using Adam [176] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, implemented in *PyTorch* [127]. A hyperparameter search has been performed to achieve sufficient accuracy for each model across all datasets. Every model was trained using batches of size 1024.

Table D.1: *Hyperparameters selected to train our models.*

Model		Mnist	FashionMnist	Svhn	Cifar-10
FF	Epochs	1200	100	1000	1000
	Learning rate	0.01	0.01	0.0001	0.0001
BP/FF	Epochs	300	300	200	200
	Learning rate	0.0001	0.0001	0.0001	0.0001
BP	Epochs	80	80	80	80
	Learning rate	0.0001	0.0001	0.0001	0.0001

D.3 Statistical test of ensemble assignment

To assign a unit i to a class-ensemble c, we compare its average activation on correct responses $\overline{x_{i,c}}$ against its leave-one-out average $\mathrm{LOO}_{i,c}$ (as described in subsection 6.2.5). We define

$$\delta_{i,c} \;=\; \overline{x_{i,c}} \;-\; 2 \cdot \mathrm{LOO}_{i,c}.$$

If $\delta_{i,c} > 0$, the unit is assigned to the ensemble for class c. To assess the statistical significance of each assignment, we compute a p-value by building an empirical null distribution. Specifically, we shuffle each relevant representation matrix row-wise 200 times, recalculate $\delta_{i,c}^{\mathrm{rand}}$ for each shuffle, and then define the p-value as the fraction of these shuffled values exceeding the observed $\delta_{i,c}$. We applied this test to every unit assigned to an ensemble across all model/dataset/layer/run combinations summarised in Table 6.3, totaling $\approx 457{,}000$ units. Of these, only 500 (about 1 in 1000) had $p > 0.05$. Most of those higher-p units occurred in Layer 2 of **BP/FF** on Cifar-10, representing $\approx 2.6\%$ of the assigned units in that specific configuration.

D.4 Activation patterns in deeper layers

In subsection 6.3.2 we claimed that in **FF** and **BP/FF** the images of a given category activate consistently a small set of units that we named ensembles, that share similarities to what is observed in sensory cortices. We reported in Figure 6.1 the activation map for Layer 1 (the first hidden layer) of **FF** trained on the Mnist dataset, and observed that very sparse ensembles emerge. In this section we show, in a similar fashion, the representations for Layers 2 and 3 (Figure D.1 and Figure D.2, respectively). We find high sparsity also for deeper layers of this specific network; a qualitatively similar conclusion is reached for **FF** models trained on FashionMnist, Svhn and Cifar-10. In **BP/FF** models a similar sparsity levels are observed, with the exception of the last layer that turns out to be non-sparse in all the datasets considered (see Table 6.3).

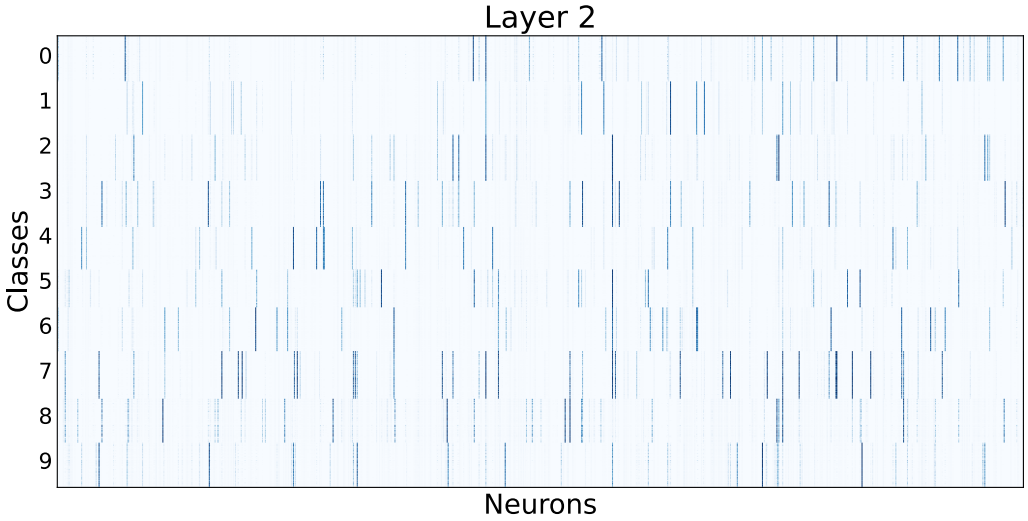

Figure D.1: *Activation patterns in a Multi-Layer Perceptron trained with the Forward-Forward algorithm, on the Mnist dataset. The image represents the activation map for neurons in Layer 2 for all images, grouped by class. A blue dot in position (x, y) indicates that neuron x is activated by input y; colour scale represents the intensity of such activation (incorrectly classified samples have been removed). Horizontal bands mark different categories; dark blue vertical lines mark active neurons. Each input category activates consistently a specific sets of neurons (ensemble). The sparsity measured according with the definition provided in subsection 6.2.5 is* 0.84.

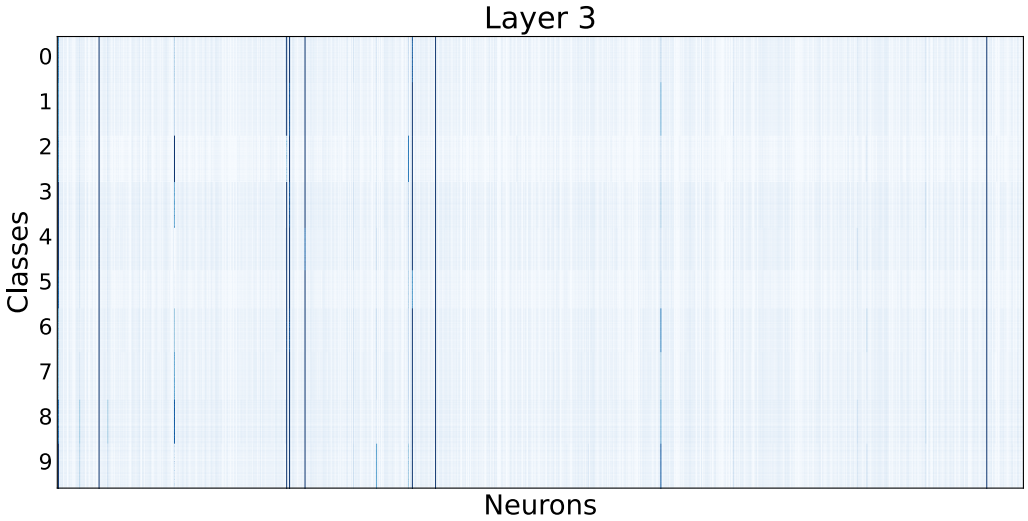

Figure D.2: *Activation reported as in Figure D.1, for Layer 3. Notice that there are only few units that activates significantly and does not play a role in discriminating categories. The role of this layer, in this experiment seems, not related to the classification task. Despite the low number of active units, the sparsity level of the representation is lower than that of Layer 2 ($S = 0.67$), due to the noise of the inactive units.*

D.5 Activation patterns in different models

In Figure 6.1 (Panel **C**) we show the activation patterns in Layer 1 of **FF** trained on Mnist. For the purpose of a qualitative comparison, we show here analogous patterns for **BP/FF** and **BP** (see Figure D.3 and Figure D.4).

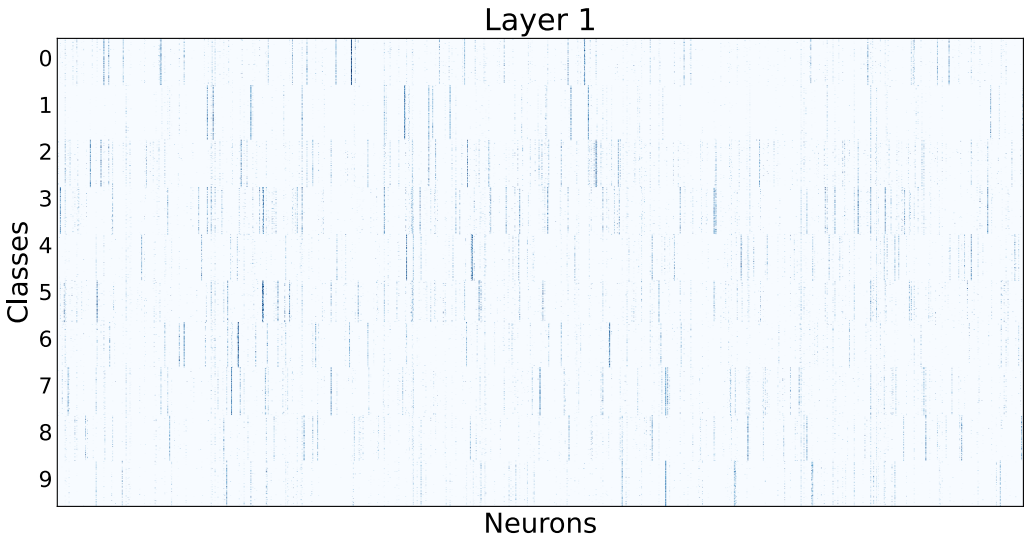

Figure D.3: *Activation pattern in Layer 1 of the **BP/FF** model trained on the Mnist dataset. The sparsity measure is* 0.89*, comparable with the correspondent first layer of the **FF** model, reported in Figure 6.1, Panel **C**.*

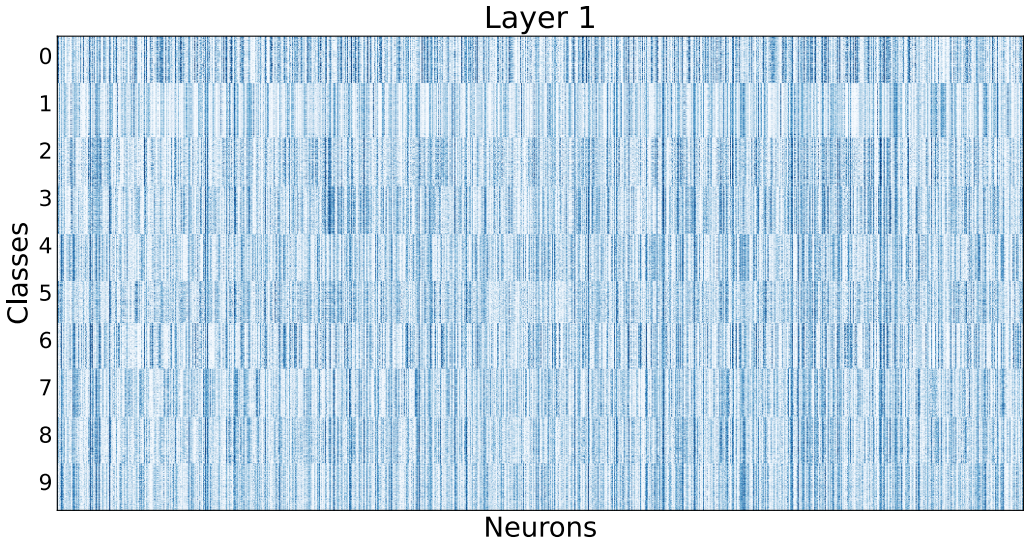

Figure D.4: *Activation pattern in Layer 1 of the **BP** model trained on the Mnist dataset. The sparsity measure is* 0.32 *(non-sparse representation), about* $\frac{1}{3}$ *of the sparsity level measured in the analogous experiment with **FF** and **BP/FF**.*

D.6 Further results on representations of unseen categories and their ensembles

We showed in subsection 6.3.4 that a **FF** model trained on the FashionMnist dataset – deprived of one category – can respond at test time to this unseen category with an ensemble (Figure 6.4).
We report here the results of similar experiments, removing one category at a time. It turns out that, in each of the ten possible cases (we performed a single run for each category), the representations of the unseen category form an ensemble; we show three examples in Figure D.5, different from the example shown in (Figure 6.4). It is with this situation in mind that we refer to "the ensembles related to unseen categories".

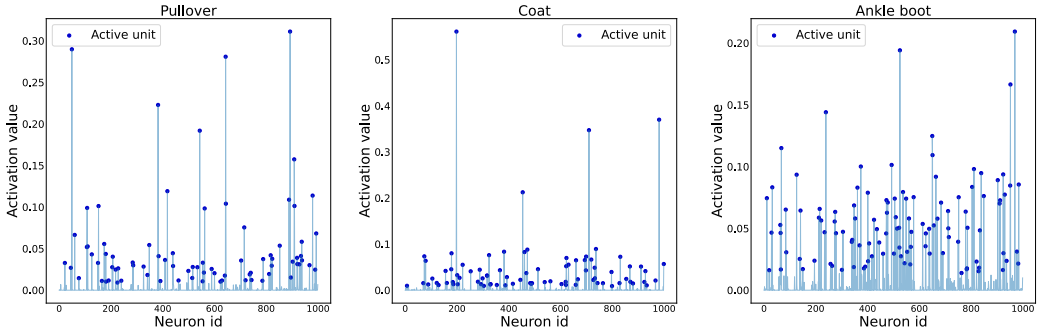

Figure D.5: *Ensembles elicited by the **FF** model trained on FashionMnist deprived of one category (we show three examples: `Pullover`, `Coat` and `Ankle boot`). We report for the three categories, the activation value of each neuron in the first hidden layer (Layer 1), averaged on all images of the unseen category. Neuron index on the x axis; average activation on the y axis. Blue dots indicate units that are considered active according to the method described in subsection 6.2.5.*

When an unseen category forms an ensemble, it generally exhibits a high level of integration with the ensembles associated with the categories encountered during training. This integration implies that it can share common units with ensembles belonging to related categories. We show in Figure D.6 how the ensembles of missing categories (same examples as in Figure D.5) integrate – by sharing units – with the other ensembles.

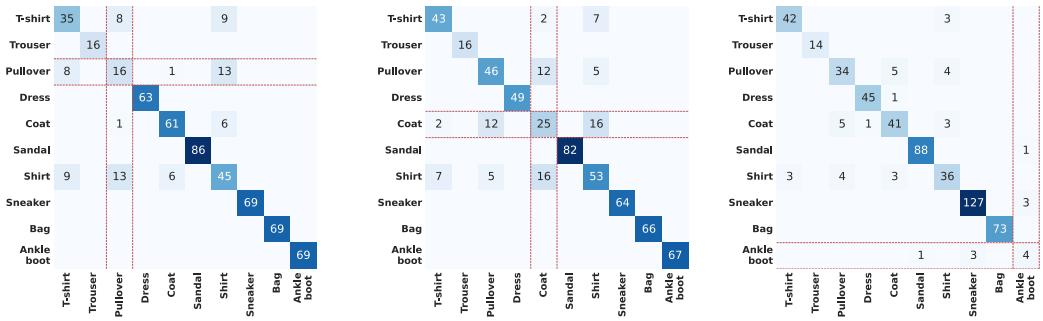

Figure D.6: *Shared units between the ensembles of unseen categories and the ensembles of categories seen during training (stripes delimited by the red lines). The results for `Pullover`, `Coat` and `Ankle boot` are shown.*

Overall, these result relates to biological neural networks [36, 103], where ensembles appear to be the functional building block of brain representations even in the absence of known stimuli.

D.7 Performance on unseen categories with linear probes

In this section, we explore the ability of a linear probe to discriminate between categories, including those unseen during training, leveraging the models' internal representations. Focusing on the FashionMnist dataset, we trained 10 models – each excluding a different category – for all of **FF**, **BP/FF**, and **BP**. A linear classifier was then trained on the training set representations. We note here that for this experiment **FF** and **BP/FF** were provided with the same data as **BP**, *i.e.* images without any pixels encoding positive and negative labels. Our findings, reported in Table D.2 reveal that the linear probes successfully recover good accuracy levels in almost all cases. This result holds consistently across all categories and models. We also report in Table D.3 the performance of linear probes averaged across all categories, seen and unseen, including a comparison with baseline models. This shows that the decoding performance of linear probes trained on models in which a category is held out during training is close to the original one (without any held out category).

Table D.2: *Linear probe accuracy on the missing category for models trained without that category.*

	Missing category									
Model	0	1	2	3	4	5	6	7	8	9
FF	0.803	0.901	0.719	0.817	0.714	0.869	0.508	0.876	0.920	0.889
BP/FF	0.787	0.939	0.699	0.856	0.742	0.894	0.454	0.836	0.947	0.937
BP	0.863	0.962	0.721	0.933	0.843	0.967	0.632	0.945	0.970	0.957

Table D.3: *Linear probe accuracy - averaged across all the categories - for models trained without one category. The Avg column reports the average across all the ten cases. The Baseline corresponds to the accuracy achieved by the model when all categories are included during training.*

	Missing category											
Model	0	1	2	3	4	5	6	7	8	9	Avg	Baseline
FF	0.854	0.857	0.847	0.854	0.849	0.851	0.843	0.851	0.858	0.862	0.852	0.849
BP/FF	0.868	0.869	0.859	0.864	0.863	0.866	0.859	0.856	0.866	0.858	0.864	0.877
BP	0.880	0.890	0.883	0.885	0.883	0.888	0.887	0.887	0.886	0.889	0.886	0.892

D.8 Enforcing sparsity in the BP model

In the **FF** and **BP/FF** models sparsity emerges without any explicit regularisation or constraint. Instead, if one wanted to promote sparse representations in the **BP** model, one

established way is by means of ℓ_1 regularisation on the activations [197]. In this section, we present the results obtained by training a **BP** model using the same hyperparameters as those employed in the main analysis and ℓ_1 regularisation on the activations, with weight set to 2.5×10^{-6}. We measure the layer-wise sparsity in the Mnist and FashionMnist datasets, and observe (Table D.4) sparsity values comparable – and often higher – than the ones that spontaneously emerge with **FF** and **BP/FF**, reported in Table 6.2.

Table D.4: *Sparsity of the **BP** model with ℓ_1 norm regularisation applied to layer activations.*

Model	Layer	Mnist	FashionMnist
BP regularised	1	0.971	0.955
	2	0.802	0.787
	3	0.813	0.781

However, despite having very high sparsity levels, the representations learned by **BP** with ℓ_1 regularisation and by unregularised **FF** display significant differences. Figure D.7 reports the Jaccard similarity between Mnist class ensembles (computed using the procedure of subsection 6.2.5) for the first layer of the regularised **BP** model (left) and the **FF** model (right). **FF** ensembles are highly specific, as they are completely disjoint, while in the case of regularised **BP** there is a non-zero overlap for any pair of classes, regardless of their visual dissimilarity.

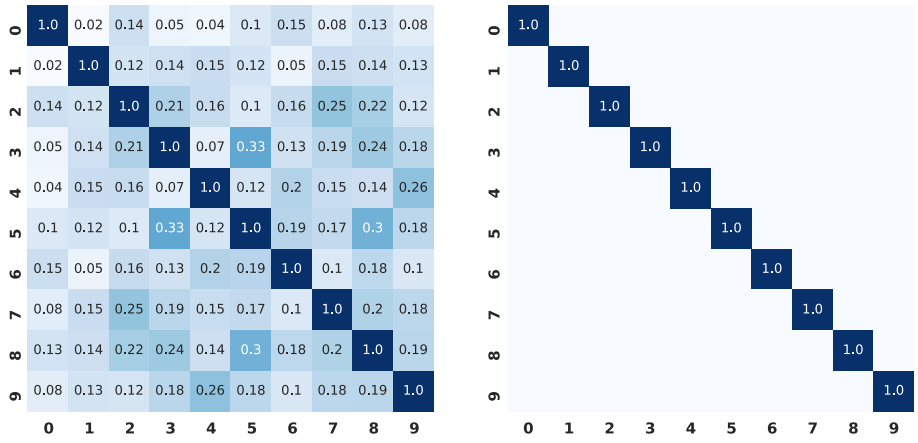

Figure D.7: *Jaccard similarity between first-layer ensembles on the Mnist dataset. Left: **BP** model with ℓ_1 regularisation. Right: **FF** model.*

D.9 Representation similarity

D.9.1 Model comparison

In this section, we analyze the similarity between representations produced by **FF**, **BP/FF** and **BP**. We employ three established representation similarity metrics, namely SVCCA [217], CKA [218] and Distance Correlation (dCor) [219]. We consider 5 training runs with independent weight initialisations for each model and compare representations layer-by-layer. The results, reported in Table D.5, show that:

- CKA and dCor exhibit greater variability than SVCCA across models and layers. We hypothesize that this may be because SVCCA is a linear metric, making it less informative in the presence of nonlinear correlation patterns;
- The first layer is almost always the most similar, possibly due to the fact that it is the one closest to the input, which is shared between models;
- Focusing on the second layer, **FF** and **BP/FF** are consistently the most similar pair of models according to CKA and dCor. The same does not apply for the third layer, which is in fact non-sparse in **BP/FF** (Table 6.3).

Table D.5: *Representation similarity between models. Results are averaged over 5 runs with independent random weight initialisation for each configuration.*

		FF v BP/FF			**FF v BP**			**BP/FF v BP**		
Dataset	Metric	1	2	3	1	2	3	1	2	3
MNIST	SVCCA	0.48	0.38	0.45	0.55	0.47	0.52	0.48	0.39	0.45
	CKA	0.79	0.53	0.04	0.53	0.49	0.23	0.62	0.43	0.02
	dCor	0.85	0.90	0.13	0.61	0.68	0.45	0.70	0.64	0.17
FASHIONMNIST	SVCCA	0.55	0.35	0.33	0.53	0.36	0.37	0.47	0.37	0.41
	CKA	0.60	0.63	0.24	0.51	0.40	0.34	0.60	0.40	0.12
	dCor	0.80	0.76	0.10	0.59	0.51	0.50	0.81	0.59	0.11
SVHN	SVCCA	0.57	0.55	0.55	0.56	0.53	0.49	0.58	0.50	0.51
	CKA	0.47	0.40	0.27	0.32	0.17	0.04	0.72	0.21	0.01
	dCor	0.60	0.46	0.09	0.45	0.27	0.13	0.90	0.38	0.03
CIFAR-10	SVCCA	0.54	0.53	0.54	0.53	0.50	0.50	0.56	0.46	0.47
	CKA	0.46	0.46	0.17	0.37	0.17	0.04	0.65	0.25	0.02
	dCor	0.64	0.53	0.06	0.49	0.22	0.07	0.86	0.39	0.06

D.9.2 Layer comparison

In Figure D.8 we report the CKA similarity between different layers of **FF** models, averaged over 10 independent training runs. Across all datasets, similarity is noticeably higher for layers 1 and 2 than for layers 2 and 3. The only partial exception is FASHIONMNIST, that displays a much narrower gap. These results align with those in Table 6.2: FASHIONMNIST is the only dataset in which sparsity in the second layer is lower than in the third one.

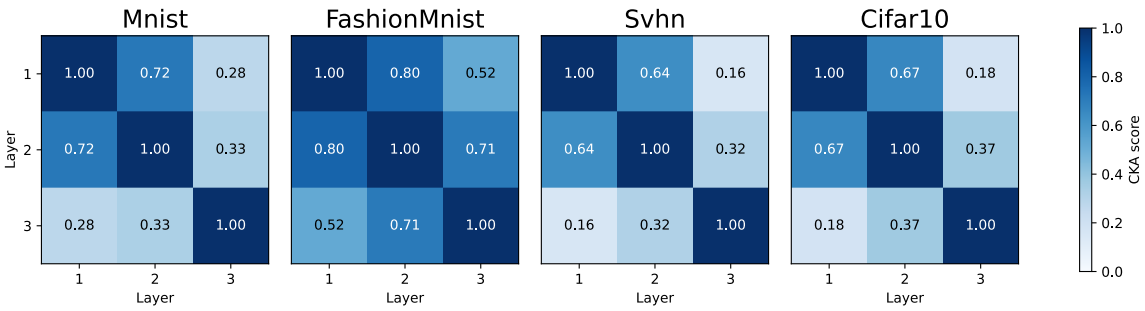

Figure D.8: *CKA similarity between layers in the **FF** model, averaged over 10 independent training runs.*

D.10 Forward-Forward with ℓ_1 goodness function

We investigated the effect of a different choice of goodness function by switching to the ℓ_1 norm. Consequently, we adjusted the normalisation for subsequent layers, performed according to the ℓ_1 norm as well. We trained 10 instances of **FF** and of **BP/FF** on MNIST and on FASHIONMNIST datasets. The hyperparameters were set as follows: learning rate: 0.001, epochs: 300, batch size: 1024. With this setup, we train the **FF** and **BP/FF** models on the MNIST and FASHIONMNIST datasets.

Our observations indicate that the accuracy achieved by both the **FF** and **BP/FF** models is comparable with the ℓ_2 results reported in Table 6.1. The level of sparsity is consistent with the findings presented in Table 6.2. However, in the case of the **BP/FF** model, the third layer exhibits significantly higher sparsity compared to when using the ℓ_2 norm. Despite this increase, sparsity remains insufficient for recognizing robust and consistent ensembles. The average fraction of active units per layer is reported in Table D.8. Results regarding ensemble overlap between visually similar classes in the **FF** model are reported in Figure D.9. The findings of subsection 6.3.3 remain valid when employing the ℓ_1 norm as a goodness function.

Table D.6: *Test-set classification accuracy for the models **FF** and **BP/FF**, using a goodness function based on the ℓ_1 norm. Results expressed as mean $\pm$ std. dev. over 10 runs with independent randomised weight initialisation.*

Dataset	**FF**	**BP/FF**
MNIST	0.949 ± 0.002	0.965 ± 0.001
FASHIONMNIST	0.859 ± 0.002	0.865 ± 0.002

Table D.7: *Average sparsity for **FF** and **BP/FF** with ℓ_1 goodness function, computed according to the definition given in subsection 6.2.5. Results are expressed as mean ± std. dev. computed over 10 runs with independent random weights initialisation.*

Model	Layer	MNIST	FASHIONMNIST
FF	1	0.887 ± 0.002	0.83 ± 0.001
	2	0.61 ± 0.005	0.647 ± 0.007
	3	0.61 ± 0.012	0.532 ± 0.012
BP/FF	1	0.944 ± 0.002	0.921 ± 0.003
	2	0.915 ± 0.005	0.919 ± 0.003
	3	0.441 ± 0.009	0.408 ± 0.011

Table D.8: *Average fraction of units taking part in ensembles for **FF** and **BP/FF** with ℓ_1 goodness function. Ensemble sizes are averaged across all categories, divided by the number of neurons in a layer, and then expressed in %. Ensembles are defined according to the LOO method presented in subsection 6.2.5. Results are expressed as mean ± std. dev.. In the third layer of **BP/FF** the representation is non-sparse.*

Model	Layer	MNIST	FASHIONMNIST
FF	1	4.22 ± 0.09	7.48 ± 0.11
	2	19.93 ± 0.53	19.44 ± 0.53
	3	17.24 ± 0.85	23.09 ± 1.15
BP/FF	1	3.88 ± 0.16	5.93 ± 0.21
	2	3.36 ± 0.29	5.26 ± 0.32
	3	-	-

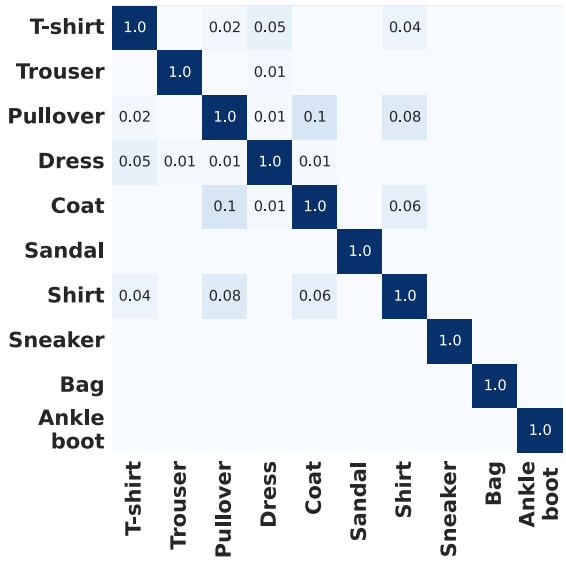

Figure D.9: *Jaccard similarity index between first-layer ensembles. Results obtained using the ℓ_1 norm as a goodness function in the **FF** model on the FASHIONMNIST dataset.*