University of Trieste
Dept. of Mathematics and Geosciences

Counter-Adversarial Recall of Synthetic Observations (CARSO)

A novel deep learning architecture and methodology for the improvement of adversarial robustness in image classification tasks, inspired by image recollection during visual learning

Research Thesis in *Deep Learning*

Author

Emanuele Ballarin

Supervisors

Prof. Luca Bortolussi

Dr. Alessio Ansuini

M.Sc. in Data Science and Scientific Computing

Academic Year 2021/2022

‘The supreme vice is shallowness’.

— O. WILDE – ‘DE PROFUNDIS’, 1897

*To all those who teach people to think deep and
to all those who teach deep networks to ‘think’.*

Contents

0 Abstract

Deep learning (i.e. *machine learning* of *deep artificial neural networks*, mainly in their *strongly-overparametrised regime*) constitutes the capstone of contemporary machine learning from the *accuracy viewpoint*, achieving *state of the art* results across a wide variety of tasks, and widespread and growing adoption in industry, consumer market, and services. Nonetheless – potentially hindering its application in critical scenarios – the paradigm may suffer from significant weaknesses, among which that of *adversarial (lack of) robustness*: the ability of purposefully-crafted *perturbations* of the inputs to unexpectedly alter the functioning of a *trained* model with respect to the *clean* inputs – and often to the informed expectations of the user – even *catastrophically*.

In the context of *supervised image classification* – on which we will focus in the present work – this may amount to a slight addition of adequately-distributed noise, imperceptible to human sight, to an otherwise *legitimate* and correctly-classified image being able to induce a misclassification with high confidence in a *neural classifier*; or, on the opposite end of the spectrum, an image resembling *white noise* being classified with high confidence as a given class, steerable by the *attacker*.

Given the utmost importance of such vulnerability in the context of *trustworthy artificial intelligence* – to ensure the development of *learning machines* whose output we can trust, and to harden them against tampering and deliberate misuse – the study of these phenomena, with the development of ever new *attacks* and *defences*, has been central to the deep learning research community in the last years. Yet, the field is evolving rapidly and – despite some remarkable results on a *case by case* basis – no universal or definitive solution exists.

In the following work, we propose a novel deep learning architecture and *training and inference methodology*, dubbed ***CARSO*** (*CounterAdversarial Recall of Synthetic Observations*), devised to defend against *gradient-based* adversarial attacks in the *white box* setting (i.e. with the attacker able to use freely the model, access weights and gradients), as a *pluggable add-on* to an *adversarially-(pre)trained* classifier. Despite requiring additional access to a dataset of *knowingly-unperturbed* images and to an *attack generation mechanism* (a subset of the requirements of adversarial training), the technique is otherwise fully *unsupervised* – allowing it to leverage any large amount of data – whose acquisition process has been deemed trustworthy – with no additional labelling effort.

For the *training phase* – *clean* images, and attacks targeting the pretrained classifier, are gathered. The *internal representation* produced inside the classifier by both sets of images is then used to condition a *conditional variational autoencoder*, learned to have as inputs the actual images producing the representation, and the corresponding *clean* images as outputs (i.e., a copy of the input if unperturbed; that before the application of the adversarial perturbation, otherwise).

This can be considered to be the unsupervised equivalent to the training of a *denoising class-conditional variational autoencoder* for *input purification.*

During inference – the representation produced by a new input in the pretrained classifier is

considered. It is subsequently used to condition the repeated *generative sampling* in the decoder, thus obtaining a collection of *candidate denoised images* associated with the same representation (and, by extension, with the original input image). On such collection, the pretrained classifier is finally used *conventionally*, and the resulting *mode class* returned as output.

Loosely inspired by the process of *active memory recollection* during *visual learning* tasks in animals, including primates and men, the technique just described is able to defend against *universal first-order white-box gradient-based* adversarial attacks effectively, and with only slight accuracy loss, in the settings investigated.

With respect to the already well established *iterative adversarial training* (with the same given *type* and *strength* of the attacks used both in training and inference), ***CARSO*** compares at slightly favourably at worst, as far as *accuracy under attack* is concerned.

A much more favourable comparison – though – is observed when *unforeseen attacks* come into play, i.e. when the attacks used during inference are potentially different (in strength and/or even type) from those seen during training – and whose usual consequence is the compromission of adversarial robustness to a varying extent, not rarely complete (i.e. *close-to-zero* adversarial accuracy). On specific occasions, ***CARSO*** against unforeseen attacks was able to recover *close-to-clean* accuracy.

Finally, the stochastic nature of the *generative sampling*, the non-differentiability of the *mode-selection* operation, and the *competing gradients* arising at the level of the pretrained classifier (used both to produce a representation serving as input to the autoencoder, and as a classifier for the very same autoencoder's output) concur at making the adversarial attack of the ***CARSO*** architecture itself a hard, constrained multiobjective optimisation problem – effectively shielding *natively* the additional subnetwork it introduces from the same pitfall it addresses in the original classifier.

0.1 Abstract (italiano)

Il *deep learning* (ovvero *machine learning* con *reti neurali artificiali profonde*, principalmente nel loro *regime fortemente sovraparametrizzato*) rappresenta l'epitome del machine learning contemporaneo dal punto di vista dell'*accuratezza*, capace di ottenere risultati *allo stato dell'arte* in un'ampia varietà di contesti, e di una diffusa e pur crescente adozione in ambito industriale, nel mercato di consumo e dei servizi. Ciononostante - e al punto da comprometterne l'applicabilità in scenari critici - tale paradigma può soffrire di significative debolezze, tra cui l'*(assenza di) adversarial robustness*: la possibilità per *perturbazioni* degl'input costruite ad arte di alterare il funzionamento di un modello *pre-allenato* rispetto ai corrispondenti input *puliti* - e spesso pure rispetto alle aspettative informate dell'utente - in modo anche *catastrofico.*

Nel contesto della *classificazione d'immagini supervisionata* - su cui ci concentreremo nel presente lavoro - questo può essere addirittura rappresentato da una lieve aggiunta di rumore adeguatamente distribuito, impercettibile alla vista umana, ad un'immagine altrimenti *legittima* e classificata correttamente, in grado di indurre un *classificatore neurale* ad una classificazione errata con elevata confidenza. Oppure, all'estremo opposto dello spettro, un'immagine costituita all'apparenza da *rumore bianco* in grado di essere classificata con elevata confidenza come una data classe-target, scelta a piacere dall'*attaccante.*

Data la cruciale importanza di suddette vulnerabilità nel contesto della *trustworthy artificial intelligence* - sviluppo di *macchine in grado di apprendere* dei cui output ci si possa fidare, e loro irrobustimento contro manipolazioni o uso deliberatamente improprio - lo studio di questi fenomeni, con lo sviluppo di sempre nuovi *attacchi* e *difese*, è da qualche anno centrale all'interno della comunità dei ricercatori nell'ambito del deep learning. Tuttavia, il campo è in rapida evoluzione e - nonostante alcuni interessanti risultati in *casi specifici* - non vi è ancora una soluzione universale e definitiva.

Nel lavoro in seguito sviluppato, viene proposta una nuova architettura per reti neurali artificiali, con associato *protocollo di training e inferenza*, chiamata `CARSO` (*CounterAdversarial Recall of Synthetic Observations*), pensata per difendere contro *adversarial attacks basati sul gradiente* nello scenario *white box* (ovvero con l'attaccante in grado di utilizzare liberamente il modello, e accedere a pesi e gradienti). Tale proposta si configura come un'aggiunta facilmente inseribile all'interno di un'altra architettura preesistente (e pre-allenata). Pur richiedendo l'accesso ulteriore ad un dataset di immagini che *si sappiano non perturbate* e ad un *meccanismo di generazione d'attacchi* (un sottoinsieme dei requisiti del classico *adversarial training*), tale tecnica è altrimenti completamente *non-supervisionata* - consentendo di sfruttare anche grosse moli di dati la cui acquisizione sia reputata legittima, senza alcuno sforzo di *labelling* ulteriore.

Nel corso della *fase di addestramento* - immagini *pulite*, e attacchi verso il classificatore pre-addestrato, sono prodotti e raccolti. A questo punto, le *rappresentazioni interne* del classificatore prodotte da entrambe queste collezioni d'immagini sono utilizzate per *condizionare* un *conditional variational autoencoder* al fine d'imparare a mappare gli input (*puliti* o perturbati che siano) verso

il corrispondente *pulito* di partenza (ovvero a copiare l'input, se *pulito*; a produrre quello precedente la perturbazione, in caso contrario).

Questo processo può essere considerato l'equivalente non supervisionato dell'addestramento di un *denoising class-conditional variational autoencoder* per la *purificazione degli input.*

In fase d'inferenza - la rappresentazione prodotta da un nuovo input all'interno del classificatore pre-allenato è estratta, e in seguito utilizzata per condizionare un *campionamento generativo* ripetuto nel decoder, ottenendo così una collezione di *immagini tentativamente prive di rumore* associate a detta rappresentazione (e, per estensione, all'immagine di partenza in input). Su questa collezione il classificatore è quindi utilizzato *convenzionalmente* e la *classe moda* risultante restituita come output.

Vagamente ispirata al processo di *richiamo attivo alla memoria* nel corso di attività di *apprendimento visivo* negli animali, inclusi i primati e l'uomo, la tecnica appena descritta è in grado di proteggere efficacemente contro *adversarial attacks white-box universali di primo ordine*, con una soltanto lieve perdita d'accuratezza, negli scenari considerati.

Rispetto ai risultati dell'ormai noto e rodato *adversarial training iterativo* (dove *tipo* e *intensità* degli attacchi sono i medesimi in fase di addestramento e inferenza), ***CARSO*** esce al peggio con discreto favore dal punto di vista dell'*accuratezza sotto attacco.*

In aggiunta, un confronto significativamente favorevole si osserva nel momento in cui si considerano *attacchi non osservati*, ovvero di intensità e/o tipo potenzialmente diversi da quelli considerati in *training* - e la cui tipica conseguenza è solitamente la compromissione della *adversarial robustness*, non raramente totale (cioè tale da produrre una accuratezza sotto attacco vicina allo zero). In occasioni specifiche, ***CARSO*** contro attacchi *non previsti* è stato in grado di esibire miglioramenti in termini d'accuratezza quasi paragonabili ad un suo recupero totale.

Da ultimo, la natura stocastica del *campionamento generativo*, la non-differenziabilità dell'operazione di *selezione della moda*, e la presenza di *gradienti in competizione* a livello del classificatore pre-allenato (dal momento che sono determinati sia nel produrre la rappresentazione input per l'autoencoder, sia l'output del classificatore stesso) concorrono a rendere un eventuale attacco all'architettura stessa di ***CARSO*** un problema di ottimizzazione vincolata multi-obiettivo, di assai difficile risoluzione. Questo consente di proteggere l'architettura aggiuntiva che ***CARSO*** costituisce dalle stesse vulnerabilità che cerca di mitigare nel classificatore, in modo *nativo.*

1 Introduction

Ten years have just passed[1] since what is informally considered the beginning of the *Deep Learning Revolution* – the landslide series of records in *visual recognition* competitions shattered by `AlexNet`[2], a *deep artificial neural network* whose training had been accelerated by the use of *graphical processing units*. It surely was not the first of its kind – and indeed the origin of the field may be traced back to the seminal and overenthusiastic work of McCulloch & Pitts[3], the critical but fundamental contributions from Minsky & Papert[4], or the more recent and optimistic introduction of the *error backpropagation* algorithm[5] which is now the almost-unanimous choice for the training of *deep neural architectures*... and all the crucial developments in between[6] – but the successes of `AlexNet` showed, probably for the first time outside the strictly academic community, the actual effectiveness of the convergence of *abundance of data*, *abundance of computing power* and properly trained *deep neural networks*. It was also the time *big data* were becoming popular in industry[7], and this contributed to further propel the field of *machine learning* (of which *deep learning* is part, and at the forefront) ahead.

Fast-forward to today, and *deep learning* has been endowed with not only an almost-ubiquitous role in everyday life of industrial societies (think, non-exhaustively, *e.g.*, of voice recognition in *smart devices*, recommender systems part of online multimedia-streaming or shopping platforms, realtime automated text-translation services), but also – and most importantly – with growing stakes in the decision process at many levels. In a varied landscape across countries – legally and socially –, it is currently not unusual to have *deep* (or, more generally, *machine*) *learning* systems assist or even replace the driver of a motor vehicle, perform candidate screening in human resource management, assess the solvability of potential debtors, validate insurance claims, aid medical diagnosis[8], plan and control supply-chain and manufacturing pipelines, *etc.*, among the many scenarios.

The clear explanation of such a widespread and ever growing use of *deep learning* lies in its sheer effectiveness – provided enough data and *compute* are available – in producing *data-adaptive* models without requiring reliance on handcrafted features, and in the ever more capable[9] systems devised thanks to increased understanding, research interest, and funding of such active and promising field.

Nonetheless, on the one hand, with the fast development of the area also comes an increased awareness and interest towards its limitations, their far-reaching implications, and potential ways to address such shortcomings; on the other hand, greater and growing adoption increases scrutiny, and interest in the transformations such new technology is producing upon society – for the good

[1] At the moment of writing, in October 2022.
[2] See [35].
[3] See [46].
[4] See [48].
[5] See [57].
[6] See, *e.g.* [24], [56], [26]
[7] See, *e.g.*, the rise in related Google searches.
[8] One example among the very many, see this Substack article by Eric Topol – that came out right during the writing of this work.
[9] See the SotA section of the PapersWithCode project for a non-exhaustive overview of their capabilities.

or bad – especially in those cases where *things do not go as planned*: anomalies, tampering, misuse, or simply an outcome perceived as unsatisfactory or damaging by some specific social group or subject.

The same interest – the latter in particular – is shared and closely followed by *law-* and *policy-makers*, in the attempt to find a delicate balance between the interests of all parties involved: the *research and development* community, the providers of services based on *deep learning*, its final users – usually at the corporate level, and all those (natural persons or otherwise) whose data are ingested by such systems and who may be finally affected by their output.

In such regard, the *Joint Research Committee of the European Commission* – while acknowledging that the goal of a cohesive regulatory framework is distant, but urgently needed – offers[10] two crucial directions for both the technical and legal community to follow, in order to ease normation and promote the development of *artificial intelligence* in the spirit of the already in-place GDPR[11]: *explainability* and *robustness* for such systems.

Along the lines of the latter of the two this work will develop. Specifically, the problem of *adversarial* robustness in *image classification* tasks by means of *deep artificial neural networks* will be addressed, and a novel technique to foil *gradient-based adversarial attacks* directed to the inputs of such architectures will be proposed and assessed.

Furthermore, from a purely scientific and more speculative viewpoint, the problem of (the lack of) *(adversarial) robustness* in otherwise extraordinarily capable, modern *deep learning* architectures may help to shed some light – though maybe just a glimpse – on the nature of biological intelligence and cognitive processes, and on how artificial *machine learning* systems may improve in key aspects by mimicking them. As such, this work places itself also at the intersection of the study of *artificial* and *biological* intelligence. Fields with common roots (*e.g.*, [46] is significant), whose goals, methods, and communities have since partially diverged, but whose deep similarities under the appropriate lens still persist, whose at times surprising differences force us to reason about their very nature and specificity, and whose potential for cross-fertilisation never ceases to attract the endless stream of human endeavour.

[10] See [23].

[11] General Data Protection Regulation of the European Union, *i.e.* `EU Regulation 2016/679`.

2 The *tools of the trade*

The aim of the section that follows will be to provide the essential elements of knowledge whose use has been abundant in, and required to better appreciate, the forthcoming – while also favouring the contextualisation of novel contributions within the broader field of *deep learning*, and that of *adversarial defences* specifically. As such, with no pretence of exhaustiveness, the following pages will focus chiefly on a high-level conceptual overview, along a *drill-down* from the goals of *artificial intelligence* down towards the very specific problems and use-cases considered.

2.1 ML $\subseteq$ AI

Artificial Intelligence is usually defined as the field that studies tools and methods capable of reproducing higher-level cognitive functions. *I.e.* rational and autonomous reasoning, decision-making and agency, and/or adaptation to complex or previously unseen scenarios. As such, it is an area that lies over a broad variety of disciplines and approaches: from computer science and mathematics, to psychology and philosophy.

The core gnosiological underpinning from which *Machine Learning* and *Deep Learning* move (called *computational cognitivism*) posits that the previously mentioned goals of *Artificial Intelligence* may be reached by reduction to – though arbitrarily rich and complex – algorithmic computation: thanks to the tools of mathematical formalisation and statistical-probabilistic reasoning as a way to quantify and operate under uncertainty.

Machine Learning can then be defined as the set of rigorous mathematical techniques (spanning *modelling*, *algorithmics*, *statistical/probabilistic learning theory*, optimisation) leading to the development of algorithms (called indeed *learning algorithms*) to extract information from *experience* (provided in the form of *data*) without being explicitly programmed to execute the specific task *learned*[12].

In a more measurable fashion[13]:

> *A computer program is said to learn from experience E w.r.t. to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

Specifically, in order to have a *program* show such kind of behaviour at a given task, it is necessary for the *learning algorithm* itself to learn from data a *mathematical model* of the essential phenomena involved in the fulfilment of the task – and whose use allows for its (eventually approximate) further execution, even on new, unseen input data. This clarification allows – in theory – to decouple the machine learning *model* from the *learning* (and potentially *inference*) *algorithm* used to determine its determining parameters.

[12] Such popular definition of *ML* comes from a rephrased quote from [59] – whose author also helped the development of TEX, the typesetting system which this thesis has been composed with.

[13] Such definition is attributed to Tom Mitchell.

2.1.1 Minimal systematics

At this point, one can preliminarily classify the (extremely varied, and often not clear-cut) landscape of machine learning tasks (and related algorithms) according to the amount of *supervision* required by the learning, or *w.r.t.* to the probability distribution the *model* is tasked to learn.

The former discrimination allows us to define:

- *Supervised* learning tasks, requiring an input/output mapping to be learned from a *training* dataset of knowingly-correct pairs of the same kind (*e.g.* image classification);
- *Unsupervised* learning, where data are provided as input and properties or transformations of them are required to be learned without further exemplification of the output (*e.g.* dimensionality reduction);
- *Reinforcement learning*, involving the determination of the optimal actions (among many) to be taken according to the state in which the *agent* and the *environment* are – by utilising only a *reward function*, and eventually under the further constraint of partial state observability and outcome nondeterminism.

Whereas, according to the latter description, we can have:

- *Generative* learning, dedicated to the modelling of the full data-generating distribution, *i.e.* $p(x)$ in the case of inputs only, or the joint $p(x, y)$ in the case of input/output pairs – or some property or transformation of them;
- *Discriminative* learning, dealing with the less informative conditional model of $p(y|x)$ – or some statistic of it – in the case of input/output pairs.

2.2 Deep Learning

Given the above, one can simply define *deep learning* as a subset of *machine learning* – whose models are *deep artificial neural networks* (whose precise nature will be right introduced).

2.2.1 From artificial neurons...

The essential building block of an *artificial neural network* – being it *shallow* or *deep* – is the *artificial neuron.* Though not a proper model of its biological counterpart – unless at a very high, conceptual level – its structure was loosely inspired[14] by the dendritic and axonal connectivity of neurons in a biological brain, abstracting away the differentiations that may occur.

From a mathematical modelling viewpoint, an *artificial neuron* is just an affine vector-to-scalar transformation followed by a (usually, except in the trivial case) nonlinear function, called *activation.*

i.e.

[14] See [46] and [56].

$$y = \mathcal{N}_1(\boldsymbol{x}) = \mathcal{A}(b + \boldsymbol{w} \cdot \boldsymbol{x})$$

with, in the most general setting $y \in \mathbb{R}$ (the output), $\boldsymbol{x} \in \mathbb{R}^n$, $n \in \mathbb{N}_{\setminus 0}$ (the vector input, also viewable as an ordered collection of scalar inputs, *e.g.* the outputs of other *neurons*). In such case, the activation $\mathcal{A} : \mathbb{R} \to \mathbb{R}$ and the model *parameters* $b \in \mathbb{R}$ (*bias*) and $\boldsymbol{w} \in \mathbb{R}^n$ (weights) – to be learned, eventually – fully define the model. In the usual scenario, the choice of $\mathcal{A}$ is not to be learned, but still its fixed functional form may depend on additional learnable parameters.

2.2.2 ...to deep artificial neural networks

If we consider, at this point, a group of N_ℓ ordered (and distinct) neurons $\mathcal{N}_i$ and provide them the same vector $\boldsymbol{x}$ as input, we obtain as output $y_i = \mathcal{N}_i(\boldsymbol{x}) = \mathcal{A}_i(b_i + \boldsymbol{w}_i \cdot \boldsymbol{x})$ for $i \in \{1, \ldots, N_\ell\}$, which we can rearrange in a vector $\boldsymbol{y}$. The transformation mapping $\boldsymbol{x}$ into $\boldsymbol{y}$ can be directly modelled by means of matrix-vector multiplication, thus defining a new mathematical device, called *linear (neural network) layer*:

$$\boldsymbol{y} = L(\boldsymbol{x}) = \mathcal{A}(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{x})$$

with $\boldsymbol{y}, \boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{x} \in \mathbb{R}^n$, $m, n \in \mathbb{N}_{\setminus 0}$ and $\boldsymbol{W}$ an adequately-defined $m \times n$ matrix. In this case, $\mathcal{A} : \mathbb{R}^m \to \mathbb{R}^m$ is usually (but not always[15]!) just an elementwise application of the same scalar function.

By considering again a group of N ordered (and distinct) *layers* – the j^{th} of which taking as input the output of the $(j-1)^{\text{th}}$ – we can finally define an *N-layers deep fully-connected feedforward*[16] *artificial neural network* $\mathcal{N}_N$ as:

$$\mathcal{N}_N(\boldsymbol{x}) = L_N(L_{N-1}(\ldots(L_2(L_1(\boldsymbol{x}))))) \ .$$

The output (or *post-activation*) of the j^{th} layer (*i.e.* $\boldsymbol{r_j} = L_j(\boldsymbol{r_{j-1}})$) – or, according to a different convention, its *pre-activation* (*i.e.* $\boldsymbol{b_j} + \boldsymbol{W_j}\boldsymbol{r_{j-1}}$) – is given the name of *j^{th}-layer representation*. The same naming convention can be scaled down at the neuron level, or up to an entire network (by considering the ordered representations of all composing layers/neurons).

Additionally, all the weights and biases of an artificial neural network (considered as part of an ordered collection) are called – as in the general statistical, *ML*, or modelling setting – *parameters* of the model; on the other hand, all the additional characterising elements described by a quantitative choice (or even non-quantitative, *lato sensu*) among many options – not meant to be learned – are

[15] See [67] as a refreshing example of such kind.

[16] Given the absence of *loops* in the (oriented) graph-based representation of the network. There, pre-activations are represented at the *neuron* level one scalar per node; outgoing edges imply the application of the *activation function* and multiplication with the scalar weight, whereas incoming edges, summation. Biases are encoded by the weight of additional edges with constant unit output. The set of all neurons with edges incoming into another (excluding biases) are called *receptive field* of the latter.

called *hyperparameters* (among which, *e.g.*, the number of layers – also called *depth* of the network – or the output size of each *layer*).

2.2.3 A whole *bestiary* of networks and *universal approximation*

At this point, one may ask whether *fully-connected feedforward artificial neural networks (FCNs)* are an adequate model to approximate the transformation of the inputs required by the task to be learned – or if other kinds of similar models are available (and more appropriate).

As far as the latter question is concerned – even if not strictly required for the prosecution of the present work – many variations have been proposed since the first *neural models*. Some of these act at the level of *single neurons* – *e.g.* by changing the way incoming inputs in the graph-representation (see: 16) of the network are handled, as in *spiking neural networks*; others change the layer-wide behaviour – *e.g.* by structuring neuron connectivity and enforcing weight-sharing, as in *convolutional neural networks*. Others more change the connectivity of the network in ways that go beyond single layers (as in *recurrent* or *graph neural networks*, where – among other differences – *e.g. feedbacks* are possible). Some of these cases – and definitely others – also require (or actually propose, as the only modification) a different *learning algorithm*[17] to be employed *w.r.t.* already established choices.

Surely – and beyond the simple variation of hyperparameters – ever new artificial neural models are routinely developed within deep learning research and practice, by variously composing already established ones (*i.e.* considering the outputs, or even more generally the representations or the parameters, of a network as the input of another), and by likewise adopting different training or inference strategies[18].

The adequacy of deep learning models to approximate a given map linking inputs and outputs of interest – and specifically whether such approximation can be learned (and how!) from examples – is a vast subject with rarely clear-cut answers, constantly evolving and attracting renovated research interest.

The main results so far – known as *universal approximation theorems* – establish for a given *artificial neural architecture*, seen as a family of algorithmically-generated functions parametrised by its weights and biases, their density within a function space of interest. Originally mostly focused on *feedforward* architectures of fixed depth at the increase of width, spaces of continuous functions between Euclidean spaces, and the notion of *density* induced by uniform convergence within compact sets – over the years many extensions to such theorems have been proposed and proven (most notably in the case of fixed width and increasing depth, and for a variety of commonly used neural architectures such as *convolutional neural networks* or those with a wide class of activation functions or subject to specific constraints).

In any case – though the striking expressive power of deep artificial neural networks is undoubtedly

[17] The main *learning algorithm* for deep neural models will be discussed in the following section.

[18] Of this latter kind is main element of novelty of this thesis.

confirmed by experimental results – such theorems are almost always *existence* ones, without providing a direct way of determining the (hyper)parameters actually approximating a given function within stated tolerance. Within the frame just described, the practical determination of such latter (hyper)parameters has largely been an empirical science – relying upon clever modelling choices (*e.g.* the choice of *inductive biases* – *i.e.* the properties of specific architectures *w.r.t.* the input/output mapping produced), the development of ever more effective learning algorithms, and always experimental evaluation.

It will go beyond the scope of this thesis to discuss in further *depth* or *width* the current state of research in the field. Some cornerstone results are contained in [28], [27], [41], [71], and [31].

2.2.4 *Actually learning* in Deep Learning

We introduced *deep learning* as a subset of *machine learning* – however, we have so far outlined just the *modelling* part of it, and briefly mentioned some formal guarantees whose transfer into practice is hardly possible. How can we *automatically* learn the the parameters of a deep learning model capable of approximately performing a given task (or at least give us the reasonable expectation it could) – only from inputs (*e.g.* in the case of unsupervised learning) or input/output pairs (*e.g.* in the case of supervised learning)?

The answer is, in principle, exactly equivalent to that typical of *traditional* numerical function approximation, or statistical model fitting.

Given a deep artificial neural network $\mathcal{N}$, we first define an adequate *loss function* – that should encode the degree of success with which the model is capable of solving the chosen task, and whose value generally depends on inputs and parameters (usually, but not exclusively, through $\mathcal{N}$) and, in the case of supervised learning, on the outputs. Then, we minimise the *loss*, optimising *w.r.t.* the parameters, while evaluating it on the given *training data*.

More precisely, calling $\boldsymbol{\theta}$ the collection of weights and biases for the entire model $\mathcal{N}$, and $\mathcal{L}_{\mathcal{N}}$ the chosen loss function, we seek the optimal parameters

$$\boldsymbol{\theta}^{\star} := \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{N}}(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}) \ .$$

Such simple formulation, however, hides one of the most relevant differences between deep learning and more traditional *(approximate) model fitting* approaches: the parameter space can be extremely high-dimensional[19], and the *loss landscape* – *i.e.* $\mathcal{L}_{\mathcal{N}}(\boldsymbol{\theta})$ – highly nonconvex, *rugged*, with abundant of local minima and/or saddle points[20]. This rules out any direct global optimisation approach, for problems beyond *toy examples*.

Though not necessarily the *only* choice – the almost-totality of learning algorithms for deep neural

[19] *E.g.*, among the largest *deep neural models* to date, Google's `GLaM` – see [14] – boasts > 1 trillion learnable parameters!

[20] For an impactful visualisation, see [40].

models is based upon *gradient descent iterations*, *i.e.* the approximation $\boldsymbol{\theta^\star} \approx \boldsymbol{\theta_{\hat{i}}}$ for a sufficiently large $\hat{i}$, and the following iteration step:

$$\boldsymbol{\theta_{i+1}} \leftarrow \boldsymbol{\theta_i} - \lambda \boldsymbol{g_i} \tag{1}$$

with $\lambda \in \mathbb{R}_+$ (*learning rate*) and $\boldsymbol{g_i} \coloneqq \frac{d\mathcal{L}_\mathcal{N}(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta_i}}$ (*local gradient*).

The exact dependence of $\boldsymbol{g_i}$ from $\boldsymbol{x}$ directly relates to the specific choice of an *aggregation scheme* of the elementwise $\boldsymbol{\theta_i}$-gradients computed for each datapoint available.

Let us suppose the usual *supervised learning* setting, with $\{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{i=\{1,2,\dots,N\}}$ as the *training dataset*. $\boldsymbol{g_i}$ iterations can then be defined as[21]

$$\boldsymbol{g_i} \coloneqq \sum_{k \in \mathcal{B}_j \subseteq \{1,2,\dots,N\}} \frac{d\mathcal{L}_\mathcal{N}(\boldsymbol{x_k}, \boldsymbol{y_k}|\boldsymbol{\theta})}{d\boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta_i}}$$

with simultaneous updates of i and j (*i.e.* for two consecutive iterations, summation is performed on different $\mathcal{B}_j$s), and $\bigcup_j \mathcal{B}_j$ a partition of the (generally shuffled) set of indexes $\{1, 2, \dots, N\}$.

The various $\{(\boldsymbol{x_i}, \boldsymbol{y_i})\}_{i \in \mathcal{B}_j}$s are called *(mini)batches* of the dataset, and their size (fixed, except eventually for last one) B (*batch size*) is such that #batches $= \lceil \frac{N}{B} \rceil$ – whereas the corresponding $\boldsymbol{g_i}$s are called *noisy local gradients* in case $B \neq N$ (as they are, indeed, a *noisy estimate* of the actual *local gradient*).

The choice of B and λ (which are additional *hyperparameters*, of the learning algorithm this time) can influence the convergence of the iterations. While it is true that *noisy gradient* iterations approximate those of true gradients as i grows, the choice of a larger B (up to the limit $B = N$, called *(full) batch gradient descent*) reduces the variance of the estimate at the cost of increased susceptibility to convergence toward local minima. On the other hand, a decrease in B (down to $B = 1$, *stochastic gradient descent*[22]) favours convergence toward the global minimum at the price of increased variance and number of iterations required to reach it. The number of times the training set (in the form of batches, eventually) is entirely used during training is called *number of epochs*. Ultimately, the choice of *batch size* in modern day boils down to the compromise[23] between *regularisation* of the optimisation problem and (reduction in) the number of iterations potentially required for proper convergence – provided in any case sufficient memory to store the whole minibatch[24].

[21] An equivalent formulation is also possible – with summation replaced by averaging of the gradients – implying a rescaling of the learning rate.

[22] Note, however, that such name is commonly used in practice to describe the whole family of these optimisation methods, regardless of the choice of B.

[23] See, *e.g.* [45] for an unusual take on the subject, endorsed by deep learning pioneer and expert Yann LeCun.

[24] While it is always possible to resort to *gradient accumulation* to counter memory starvation (see this forum comment by one of the PyTorch developers as an explanatory example), a memory-fittable alternative has to be preferred, due to the non-summability of *batch normalisation* statistics. The analysis of such regularisation technique will be discussed right next.

Finally, it must be stressed that *optimisation* plays a crucial role in the *learning algorithm* of a deep neural architecture: for that reason, many variations of the prototypical iteration described in (1) have been proposed. Chiefly – they augment such iteration with additional *iteration variables* (and subsequent additive terms to the gradient) in order to better exploit *greater-than-first* order information about the *loss landscape* and thus provide faster and/or more accurate convergence to the *true* global minimum. A thorough description of such proposals is again out of the scope of this work: however, the most common device used in this context must be mentioned. It is the case of *exponentially-weighted averages* of the gradient. As an example, the nowadays ubiquitous *stochastic gradient descent with momentum* variation proposes a simultaneous iteration of the type:

$$\boldsymbol{m_{i+1}} \leftarrow \boldsymbol{\theta_{i+1}} - \boldsymbol{\theta_i}$$

$$\boldsymbol{\theta_{i+1}} \leftarrow \boldsymbol{\theta_i} - \lambda \boldsymbol{g_i} + \beta \boldsymbol{m_i}$$

with $\beta \in (0, 1]$. Such addition – mimicking the *momentum*, indeed, of a body moving subject to a potential $\mathcal{L}_{\mathcal{N}}$ – *e.g.* both provides noise attenuation for the gradient estimates and ameliorates the convergence towards global minimum in the case of highly unbalanced (in absolute value) gradient components across dimensions. Additionally, `Adam`[25] – one more among the most relevant and used variations – further regularises learning via modulation of the *effective learning rate* in accordance to better estimates of local curvature.

2.2.5 *Overparametrisation* and regularisation strategies

As we have anticipated earlier, the number of parameters of a deep neural model can be even extremely large – and usually purposefully so: excessive parameter parsimony may in fact artificially cap model expressiveness beyond the (usually unknown beforehand) requirements of the task to be learned, and render optimisation by gradient descent harder[26].

Still, the deliberate modelling choice of employing a number of parameters (much) larger than reasonably estimable – so-called *overparametrisation* – does come at the cost of increased risk of overfitting[27] and unmanageable complexity. To counteract such downsides – as in *traditional* statistical practice – *regularisation strategies* have been developed, some of them specific to deep learning, which will be discussed next.

2.2.5.1 Weight decay

First of all, one could apply – to the optimisation problem of finding $\boldsymbol{\theta}^{\star}$ – the same *penalisation*

[25] See [32]. Also, `Adam` is the basis for the main optimiser used in the learning algorithm proposed by this work – `RAdam` (see [42]). The improvement upon `Adam` consists in an accurate analytical *de-biasing* of the *adaptive learning rate* variance, especially in the first few iterations of the algorithm, similarly to the previously-known *warmup* heuristic.

[26] Such latter statement has been heuristically explained by Terrence Sejnowski as follows: the probability that no direction of the *local gradient* points towards a pre-set point – in this case, the global optimum – decreases as the dimension of the parameters space increases, even for a randomly-picked *local gradient* vector.

[27] The loss-minimisation-driven adaptation of the model to the training dataset at the point of losing generalisation ability – *i.e.* the inability to learn the abstract task beyond the specific exemplifying data.

regularisation techniques typical of *traditional* statistics; in particular, L_2-penalised fitting (*à la* ridge regression). Such approach is called – within the deep learning community – *weight decay*[28]; it is indeed possible, for iterations of the type described in (1) to express the $\mathcal{L}_{\text{ridge}} \leftarrow \mathcal{L} + \frac{\gamma}{2}||\boldsymbol{\theta}||_2$ regularisation as a modified iteration, *i.e.*

$$\boldsymbol{\theta_{i+1}} \leftarrow \boldsymbol{\theta_i} - \lambda \boldsymbol{g_i} - \lambda\gamma\boldsymbol{\theta_i} \ .$$

The two routes, however, become different in the cases – outlined above – where exponentially-weighted averaging, too, comes into play at the optimisation step by means of a more sophisticated scheme. This previously unnoticed detail has since then produced, once again, different variations of previously known optimisers[29] – with an empirical general preference for proper *weight decay*, but no clear-cut conclusion (and a relatively modest effect, compared with other *tricks*).

2.2.5.2 *Dropout*

Another[30] – *deep-learning-specific* – technique directly attempts model fitting with a stochastically-selected subset of parameters: for each batch of training data, some randomly-sampled neuron representations (in a given pre-set proportion, layerwise) are forced to zero, resulting in the corresponding weights and biases to be unmodified by the learning iteration. At *inference time*, no constraint is imposed, but each *weight* is further weighted by the corresponding complementary probability of *training time* zeroing.

Such technique – and its more advanced variations[31] – have demonstrated remarkable success in improving generalisation, by taking into account in a less intertwined fashion the various different *pathways* activated through the model by a given input example – thus increasing the overall parameter efficiency of the network. This comes at the cost of a generally less *sparse* and more *distributed* representation – which may not always be an intended goal of the training.

2.2.5.3 *Batch Normalisation*

Though not properly considered a *regularisation technique*, *batch normalisation* has a twofold effect: speeding up and improving converge to the *true* global minimum for practically any optimiser, and reducing the *noisiness* of batched input data – by normalising each datapoint coordinate within its respective batch, and further scaling and shifting it according to further learnable parameters.

Initially considered to due its efficacy to a reduction in so-called *internal covariate shift*[32], it has been later established[33] that its main contribution is a net smoothing effect on the loss landscape.

2.2.5.4 Learning rate scheduling

Finally, in the same family of *improper regularisation techniques* as *batch normalisation*, there is

[28] See [36].
[29] See, *e.g.*, the most relevant optimiser of this kind, `AdamW` – described in [43].
[30] See [63].
[31] See *e.g.* [5].
[32] See the original paper, *i.e.* [30].
[33] [61].

learning rate scheduling: the adaptation of *epoch*- (or, more generally *batch*-) *specific* learning rate to some pre-set or adaptive schedule. Such technique, originally developed for *non-learning-rate-adaptive* optimisers, further speeds up convergence – while simultaneously avoiding that excessively large steps during the initial iterations of the algorithm steer convergence away from the global minimum, and during the later stages prevent convergence to narrower, but optimal, basins. Many scheduling schemes – with varying levels of theoretical justification and/or empirical vetting, but no definitive evidence across all possible application scenarios – have been proposed, and their actual choice often depends on habit, compromise between improvement and additional hyperparameters to tune, or *brute force trial and error.*

2.2.6 *Algorithmic differentiation* and the *backpropagation* algorithm

We are left – at this point – with one last, but crucial, question: how does all the *machinery* outlined until now work in practice, at the implementation level? In fact, until now, we have neither talked about how the terms $\boldsymbol{g_i}$ in (1) are computed from the mere datapoint-wise knowledge of $\mathcal{L}_{\mathcal{N}}(\boldsymbol{x}|\boldsymbol{\theta})$, nor we have put specific constraints on the differentiability (or even continuity!) of $\mathcal{L}_{\mathcal{N}}$: and in fact such constraints are mostly unnecessary in practice[34].

The core ingredient seamlessly allowing such kind of computations is (collectively) called *automatic* or *algorithmic differentiation* – and refers to a general algorithmic strategy to determine exact pointwise evaluations[35] of derivatives for (practically) any piece of *legal* code in a given programming language. The underlying principles and technicalities powering such approaches go far beyond the scope of this work. As a *proof-of-concept* justification, an intuitive mechanism allowing similar flexibility (though rarely used in modern *AutoDiff* frameworks) – that of *dual numbers* – is thoroughly described in [17]. An accurate and comprehensive survey of *automatic differentiation* methods for machine (and *deep*, indeed) learning is available in [3].

Finally, in order to ensure computational efficiency for the whole process, the *error backpropagation* algorithm is employed – guaranteeing that the term $\boldsymbol{g_i}$ is computed linearly in the number of parameters *w.r.t.* the number of atomic differentiation operations. This comes possible thanks to the graph-based representation of a neural architecture, the application of the *chain rule of differentiation* (by noting that in a *multilayer* architecture the derivatives of representations at layer i only depend from at most all the representations of layer $i-1$), and the *memoisation* of such derivatives during the computations required by those of the innermost layer. Such approach to the computation of the entire $\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}$ Jacobian from a reverse network-graph traversal (and the relevant related data structures built upon it) is called *reverse-mode automatic differentiation* and powers the large majority of modern deep learning libraries.

[34] Indeed, heuristically – for any reasonably modern *automatic differentiation* framework, such as PyTorch, see [54] – it is strictly sufficient that $\mathcal{L}(\boldsymbol{\theta})$ and $\mathcal{L}(\boldsymbol{x})$ are piecewise *dual-number*-differentiable within open sets, and such *pieces* definable by algorithmic branching and/or recursion. *I.e.* discontinuous activation functions are allowed, and so are selection function such as in-place sorting.

[35] *I.e.* without resorting to *numeric approximation* of derivatives, and neither computing them symbolically such as in *computer algebra systems* or *e.g.* `SymPy`.

2.2.7 *(more)* Advanced topics in *deep learning*

To conclude the overview of preliminary knowledge used in the portion of work that follows, this section will present some – selected – fundamental principles and results taken from deep learning theory or practice, with no goal of completeness. Such abridged exposition – at the cost of partiality – represents though the collection of core underpinnings on which our novel proposal is based, as opposed to the easy misconception of it being just a *random sample* of tricks and fortuitous consequences.

2.2.7.1 *Deep Learning* as *homoiconic* approximate probabilistic programming

Even though the previous sections may have given the impression of deep learning as a field composed of mostly disjoint techniques and *architectures*, remarkable synergy among all different *moving parts* is one of its most distinguishing elements. To the point that its development and use – a concept spearheaded since 2018 by one of its pioneers, Yann LeCun – can even be viewed through a *fresh* lens as a mostly compositional activity, in a similar spirit as organic synthesis or computer programming. Indeed, such latter analogy – that of a set of pre-constituted patterns (*e.g.* constructs such as *declarations of variables*, *typing*, *conditional statements*, structuring in *functions* or *objects*, use of design patterns, ...) each with self-standing dignity and theoretical justification, but variously and cleverly extended, assembled, and vetted as a new whole – is also a remarkably clear high-level description of the *true nature* of modern deep learning: the composition of transformations between real-valued, (usually high-dimensional) vector spaces – whose very nature is learnt through optimisation of an adequate metric – of which the *entry-* and *exit- points* may be the actual training or inference data, but whose internal representations and dominant ingredients are *probability distributions* or surrogates thereof.

This description, together with the realisation that, indeed, such *learnably-parametrised* transformations may be able to naturally *end-to-end* transform inputs into outputs in ultimately an approximately same way *traditional* computer programs do[36], allows to properly frame deep learning as an adaptive platform for example-driven programming that is both *homoiconic*[37] and able to express uncertainty probabilistically, yet within a finite, well-defined description.

Additionally, such viewpoint leads remarkably close to *neural computation* in biological systems, where a similar framing of the field is offered in terms of (biological) neurons capable of plastic analogue processing of frequency-coded electrical stimuli (with the astounding and tangible result that cognition, indeed, is!) – further strengthening the connections[38] of deep learning with its original motives.

2.2.7.2 *Auto-encoders*: compressing by *learning to reconstruct*

Back to a more specific description, the first of such aforementioned *patterns* we will describe is

[36] And even *Turing-completely* so; see *e.g.* [55].

[37] *I.e.* such that the *program* is at the same time represented in terms of (some of) its *primitive types* – either *parameters* or *representations* in the case of deep learning, when not even both.

[38] Pun involuntarily intended.

that of the *auto-encoder*[39]: a peculiar deep learning architecture used to first learn, and then apply, a *compression* of input data. In this case – and almost always, unless explicitly specified – such input is to be considered part of a somehow structured collection: being it that of the samples of a specific – but potentially unknown – probability distribution of interest (or a mixture of them), or more broadly that of inputs generically satisfying a property (*e.g.* being natural images – *i.e.* not being purposefully generated in a corner-case manner, for whatever reason! –, being recordings of a human voice or animal call, being time *ticks* of stock title pricing, *etc...*), often enough to specify the context dictating their use.

And indeed, the goal of finding any *compressed representation* for the data of interest directly translates in exploiting the properties of the overall collection – or the collection of properties of individual datapoints – to reduce their dimensionality in the most information-preserving[40] fashion.

The practical approach required to accomplish such (potentially very hard!) goal is on the other hand particularly simple.

The neural architecture will be composed of two subnetworks $\mathcal{E}$ and $\mathcal{D}$ – respectively the *encoder* and the *decoder* – linked by the following conventional relations:

$$\boldsymbol{c} = \mathcal{E}(\boldsymbol{x});\ \ \tilde{\boldsymbol{x}} = \mathcal{D}(\boldsymbol{c})$$

with, usually, $\dim(\boldsymbol{c}) \ll \dim(\boldsymbol{x})$ ad a loss encoding the similarity between the original input $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$, that constitutes in this case its tentative reconstruction – *e.g.* an L_2 similarity metric *s.t.* $\mathcal{L}_{\text{AE}}(\boldsymbol{x}) = ||\boldsymbol{x} - \mathcal{D}(\mathcal{E}(\boldsymbol{x}))||_2$.

The *dimensionality bottleneck* – also called indeed an *information bottleneck* – induced by the property $\dim(\boldsymbol{c}) \ll \dim(\boldsymbol{x})$ forces the network, trained under the loss-driven optimisation, to reconstruct the original input from a much smaller-dimensional representation of it; and as a consequence, forces the *encoder* to convey as much information as possibly learnable in the $\boldsymbol{c} = \mathcal{E}(\boldsymbol{x})$ step – with the *code* $\boldsymbol{c}$ being the compressed result sought after.

At this point – with $\mathcal{D}$ usually discarded except in very particular scenarios – the learnt *compressor* $\mathcal{E}$ may be used on any potential input, in order to obtain a *compressed representation* of it – which is then treated as its characteristic, lower-dimensional, set of *features* for further processing.

One potential downside of such architecture – beyond the specific task of learning a *compression* of the data – is the fundamental lack of use for the corresponding *generator*, which is nonetheless functional for the whole, but not on its own.

2.2.7.3 Learning distributions by *auto-encoding* sampled data

Right from such latter observation, one can consider the thought process leading to the development

[39] A nowadays well-established architecture in the field, for which it is difficult to backtrace to a clear *first* in literature. See, *e.g.*, [34].

[40] Here intended both in the *information-theoretical* sense, more formally – but also in the less *well posed* meaning of *'being able to preserve all features of interest, while potentially discarding the less relevant ones'*.

of the *variational*[41] and *variational conditional*[42] autoencoders: probabilistic counterparts of the *simple autoencoder* which allow generative modelling (and indeed whose *encoder* alone is almost never used on its own).

The main issue in dealing with the *decoder* of an *autoencoder* in the attempt to generate samples from the same collection – or more precisely, in this case a proper *probability distribution* of the inputs – is the *understructuredness* of the *latent space* (*i.e.* the space of the codes). In fact – being autoencoders only trained to match single given inputs to single learned codes (and viceversa) – no guarantee is offered on the learnability of a *meaning* for codes outside those coincidentally correspondent to inputs *fed* at inference time. Only in such case, a well-formed reconstruction is probabilistically guaranteed within arbitrarily tight bounds[43].

Additionally, even in the case a *probabilistic* device is employed in order to *inform* the encoder/decoder pair about a sampling process happening in latent space (as will indeed be the case for the *variational* class of autoencoders), the tractability of the sampled posterior (*i.e.* the output of the decoder, in this setting) is far from guaranteed *w.r.t.* differentiation and thus loss gradient computation – both in general terms due to the sampling itself, whose parameters may be inter-dependent, and for specific, but arbitrary, probability distributions to be sampled from in latent space.

Both these concerns are addressed – for parameter-only conditional *p.d.f.*s (e.g. in the form $\boldsymbol{x} \sim \boldsymbol{p}(\tilde{\boldsymbol{x}}|\phi_1, \ldots, \phi_{s\in\mathbb{N}})$) by maintaining the same architecture for the encoder/decoder pair, further endowing it with:

- An actual **sampling in *code-space***: the result of the encoding is not a deterministic code, but a collection of ordered parameter-vectors specifying a given-form *p.d.f.*, and a sample of whose is passed onto the decoder.
- A latent *p.d.f.* adequately reparametrised (***reparametrisation trick***) in such way that the model posterior may be obtained as $\tilde{\boldsymbol{x}} = \mathcal{D}_{\boldsymbol{\theta}_{\boldsymbol{\mathcal{D}}}}(\boldsymbol{c})$ – a deterministic function of $\boldsymbol{c}$, parametrised by the weights of the decoder – and $\boldsymbol{c} \sim \boldsymbol{p}_{\text{latent}}(\boldsymbol{c}|\mathcal{E}_{\boldsymbol{\theta}_{\boldsymbol{\mathcal{E}}}}(\boldsymbol{x}))$.

Since $\boldsymbol{p}_{\text{latent}}$ cannot be jointly learnt from data and being able to satisfy the requirements of the *reparametrisation* trick, the loss function to be adopted should both ensure that $\boldsymbol{x}$-$\tilde{\boldsymbol{x}}$ similarity is preserved, and that the samples $\boldsymbol{c} \sim \boldsymbol{p}_{\text{latent}}(\boldsymbol{c}|\mathcal{E}_{\boldsymbol{\theta}_{\boldsymbol{\mathcal{E}}}}(\boldsymbol{x}))$ are close to those of a known, given *p.d.f.* from which they will be sampled at inference time to observe new posterior samples as the output of the decoder.

The resulting loss becomes $\mathcal{L}_{\text{VAE}} \coloneqq \mathcal{L}_{\text{AE}} + \text{KL}(\boldsymbol{p}_{\text{latent}}, \boldsymbol{p}_{\text{lobs}})$, with $\boldsymbol{p}_{\text{latent}}$ the desired latent *p.d.f.*, $\boldsymbol{p}_{\text{lobs}}$ the observed latent *p.d.f.* obtained as $\boldsymbol{p}_{\text{latent}}(\boldsymbol{c}|\mathcal{E}_{\boldsymbol{\theta}_{\boldsymbol{\mathcal{E}}}}(\boldsymbol{x}))$, and KL their *Kullback-Leibler divergence*.

[41] See [33] for the paper in which it was introduced.

[42] See [62] for the paper in which it was introduced.

[43] In the sense of *Probably Approximately Correct* bounds, given enough – but finite – width and depth of the architecture, training data, training epochs, and a sufficiently large dimension for the codes. For a hint on how such actual estimations are performed, see *e.g.* [16] and [13].

Overall, the process described by the minimisation of such *loss*, closely resembles that of *variational free energy minimisation*, whence the name of such architectural *pattern*.

Finally, in order to sample from a conditional *p.d.f.* the *random variate* posterior $\tilde{\boldsymbol{x}}_{\text{r.v.}} \sim \boldsymbol{p}(\tilde{\boldsymbol{x}}_{\text{r.v.}}|\boldsymbol{x}_{\text{c.v.}}, \phi_1, \dots, \phi_{s\in\mathbb{N}})$, given a set of conditioning variables $\boldsymbol{x}_{\text{c.v.}}$, the training protocol is further modified. Both $\boldsymbol{x}_{\text{r.v.}}$ and $\boldsymbol{x}_{\text{c.v.}}$ are jointly given as input to the encoder, and the *code* is built by further concatenating the output of the encoder with $\boldsymbol{x}_{\text{c.v.}}$. The loss function and the generative process are the same as in the case of *VAE*s.

2.3 *Adversarial Robustness* (and lack thereof)

The previous sections – though tentatively *hype-agnostic*, but also never denying evidence of success! –, by describing only what we know *has worked* so far, may have given the impression of *deep learning* as a flawless paradigm able to solve any problem *thrown at it*, no matter how tractable or difficult to think it as such[44].

In spite of extraordinary results – at even *human* or *better-than-human level* in a well-specified set of tasks – *deep learning* is not free from pitfalls. Among which, that of *vulnerability to adversarial attacks*, which constitutes a major hurdle to the adoption of deep learning in safety-critical or heavily-regulated scenarios, and among the most relevant to address in order for the public to trust it with their most valuable assets or decisions. Additionally, the study and development of methods to improve *adversarial robustness* have also a direct effect in enhancing the behaviour of solutions in presence of *non-adversarial* but *unforeseen*, *corner-case*[45] inputs.

2.3.1 *Essential* definition(s)

Trying to preserve the greatest generality possible, *adversarial attacks* towards a machine learning system are specially-crafted (inference-time) inputs designed to disrupt its expected behaviour. As most often referred to in the context of classification – which we will discuss in this thesis – we will focus on inputs to a classifier, able to produce a *misclassification.*[46]

2.3.1.1 Commented systematics of *adversarial attacks*

Still within the scope of (supervised) classification, we can first classify *adversarial attacks w.r.t.* the types of anomaly they seek to produce in the attacked model.

Focusing on the **input**, we have:

[44] It is difficult not to think at the fact that a similar trend, is happening – this time *for real* (or maybe even not so *real*) – in the small/medium-sized industry landscape, trying to deal with *artificial intelligence.*

[45] As we will see, the phenomenon of *adversarial attacks* rises indeed from – quite literally – *corner-cases*, in classification settings.

[46] Let us preliminarily address a typical concern, here. Indeed one may ask what *exactly* a *misclassification* or *unexpected* behaviour is. Without delving into an epistemological *rabbit hole*, it will suffice to define them as any marked difformity *w.r.t.* the (rational, informed – if applicable) confident decision of the prototypical system the model is trying to mimic. *E.g.* in the case of image captioning for a *social network* platform, any caption grossly misdescribing the original picture for the majority (or *key interest groups*) of its users (or a statistical sample thereof).

- *(purely) Synthetic* attacks, if they produce inputs exclusively from the use of the attacked model – and, of course, of an *attack generation algorithm* – (*e.g.* excluding the possibility of exploiting external knowledge, datasets, *etc.*). The result is usually similar to data *noise*, generally carries no resemblance with elements of a hypothetical training dataset, and (in cases where inputs are perceivable) is easily recognised as anomalous by a trained human operator.

- *Perturbative* attacks, if they are expressed as a *perturbation* of legitimate – already known to the attacker – input. Perturbations are usually additive and constrained in their norm[47], resulting in realistic and often undetectable (provided the constraint is tight enough) perceptual alterations with high effectiveness in *fooling* the classifier. They usually carry the highest risk, due to their scarce detectability (even by machine learning models themselves trained for the task – being them susceptible to the same kind of attack!)[48].

If we move to the analysis of **outputs** (or better, the changes in output before and after the attack)[49], we can distinguish among:

- *Targeted* attacks, if the result of the attack should be classified as a given, pre-stated, class. Or, more rarely, if the input to be perturbed should belong to a given, pre-stated, class.

- *Untargeted* attacks, if a change in output is the only desired outcome – with no further constraint. Generally speaking, *untargeted* attacks are easier to perform successfully – and more difficult to defend against. Additionally, the set of *untargeted* attacks also contains the *best*[50] *targeted* attack among all classes.

Finally, we can further subdivide *attacks* in relation to the ***degree of knowledge*** of the *adversary* about the model attacked. We can perform:

- *Black-box* attacks – *i.e.* attacks relying only on knowledge extractable from the model along strictly expected use. This includes, and is restricted to, inputting data and reading the corresponding output. No further access is given to model architecture, structure or representations/gradients. This is the less susceptible case and corresponds to the model being deployed behind an *ideal* API (with no rate limiting, though), and maximally relying on *security by obfuscation.*

- *White-box* attacks – *i.e.* attacks that can rely on all extractable knowledge from the model, as if `root` access was granted to the very system the model is running on. This includes the

[47] In the scenario – assumable *w.l.o.g.* – where inputs are real-valued vectors or mappable as such.

[48] Additionally, *perturbative* attacks include – for a constraint lax enough, and starting from whichever single one input – all possible *purely synthetic* counterparts. This makes the *perturbative* setting a sufficient one to be studied.

[49] We implicitly assume here, and in all the following, the perspective of *top-1 accuracy under attack assessment*: the model is considered as an *end-to-end* system outputting the most probable class corresponding to its input – the only class we care about. This has numerous advantages *w.r.t.* the work that follows, as it allows for easier comparisons of attacks exploiting different vulnerabilities in the attacked classifier, is generally harder to recover *w.r.t.* the unattacked model, and avoids the otherwise *cyclical reference* fallacious rank-based approaches, implicitly relying on the fact that the positions of output classes in a ranking are conditional on the previous ones having *truly* that position – which is disproved after a successful *top-1 class attack.*

[50] In the sense of *lower loss* $\arg\min$; this will be clear after we discussed how (the greatest majority of) attacks are generated.

ability to input data into the model, read the corresponding output, but also its architecture, implementation details, representations, gradients, auxiliary variables of layers (*i.e. batch normalisation* statistics) or of the optimiser. The threat model closely reflects that of services operating on a compromised system, or that of commercial items that can be bought and probed in a protected environment. This is the hardest scenario for the defender – and reasonably the most realistic for the widest array of *actually* deployed products.

- *Grey-box* attacks – comprising all attacks lying in between the two: usually a *white-box* with additional restrictions imposed.

2.3.2 A geometric intuition: the *manifold hypothesis*[51]

In order to better reason about *adversarial attacks*, *defences*, and how it is possible to improve the *robustness* of deep neural models – let us focus first on a (relatively intuitive[52]) introduction to the *data (sub)manifold hypothesis*: it will also naturally lead to a precise description of *perturbative* adversarial attacks, and some quantitative measures for their assessment.

In the general setting of a classifier accepting *real vector-valued* inputs of given dimension, the model itself will produce a valid class as output, among those on which it has been trained[53], independently from the actual nature of such input (*e.g.* it being a *natural image* or not) and just provided it is *legal*. This allows for a hypothetical, idealised, exhaustive labelling of any possible *legal* input with the class in which it is mapped through the classifier – which is indeed a *partition* and equivalent to a full specification of the trained model. We will now ask how *input data of interest* are *placed* within such partition and in relation to their ambient space.

The *data manifold hypothesis*[54] posits that data belonging to the hypothetical collection on which the classifier is *properly* expected to be used – and of which, excluding *gross* mistakes or procedures explicitly meant to be otherwise, also the *training data* is part – belong to a manifold whose intrinsic dimension is (much) smaller *w.r.t.* its ambient space. On such manifold, we could[55] also draw the *ground truth* decision boundaries we would expect the *ideal* classifier to abide to.

From the intersections of the *data manifold*, *ideal decision boundaries* on it, and classifier *decision boundaries* in input space, we can identify the following regions (in input space):

- The intersection of *ideal* and *actual* decision regions, on the *data manifold*. In this region the learned classifier behaves exactly as expected *w.r.t.* the predicted class.

[51] For a much more *in-depth* analysis, see *e.g.* the excellent [15].

[52] By choice, and not being a necessary requirement for the theory itself – whose exhaustive and far-reaching analysis is out of the scope of this work.

[53] This explicitly excludes classifiers specifically-built for *open set detection*. However – even thought the inner dynamics leading to an input being classified as part of the *open set* are different from those class-specific – the *open set* may be considered just as an additional class like all the others, and the reasoning will still be valid!

[54] Even though called a *hypothesis*, of such next claim there are theoretical supporting arguments, and experimental proofs within specific data domains – even dating back thirty years, though not directly in this form; see [37] for an interesting example of such kind. What it is more *hypothetical*, indeed, is its relationship with *robustness*.

[55] The hypothetical is used because of the impossibility of doing it exhaustively, unless the dataset is algorithmically generated with such goal in mind.

- Still on the *manifold* – outside the previous region – and close to *misplaced* decision boundaries of the classifier. Even though such region is the *richest* of training input data, this is very often not enough to ensure exactly *ideal behaviour* is observed[56]. Inputs belonging to this region will definitely *look* real (or an interpolation of different realistic datapoints): some of them will surely be *perturbative adversarial attacks* to the classifier.

- Still on the *manifold* – outside previous regions – and far from the regions of high *boundary-point density.* Training datapoints in such region should be minimal (indeed, exactly zero in the case of an *ideally-regularised* model of the adequate size and trained until complete convergence) and misclassifications occur due to *e.g.* the represented class being not a *legitimate* output of the classifier[57]. Such region does not contain (practically *by convention*) *adversarial attacks* and poses little risks for the user or the provider, and are easily recognisable from experience.

- The *off-manifold* region – for which a *true class* is not even definable. Such region contains the most *adversarial attacks*, including all those resulting from larger perturbations. It is practically unavoidable for any classifier trained on *clean*[58] inputs of sufficiently high dimension[59] and poses manageability problems for such reason and due to its lack of structure *w.r.t.* the *data manifold.*

In the usual process of *attack generation* – outlined below – none of such region is *intentionally* targeted (with minor exceptions pertaining only to the *off-manifold* region), but one of them necessarily results as containing the best solution of a *loss-minimisation* problem similar (if not identical, in some cases) to that of *training* an artificial neural network.

2.3.3 *How it's made:* Attacks

From the perspective of an *attacker*, no attack is better than that which succeeds. While such truism may justify *literally* any technique resulting in even occasional misclassifications – which, even if the result of sheer luck, may carry extraordinary amounts of risk or actual damage, nonetheless –, a more systematic approach to the problem, especially from the standpoint of the *defender* who must counteract such attacks, strongly benefits from mathematical formalisation and amenability to optimisation strategies to automate, speed up, and pinpoint the search of relevant perturbations.

Given the generality of the *perturbative* setting, attacks usually seek additive perturbations to knowingly *legitimate* inputs, bound by a measure of strength.[60]. An ϵ-*perturbative* adversarial attack to the classifier $\mathcal{N}$ (considered as an *end-to-end class-outputting device*), given $\epsilon \in \mathbb{R}_{+,\setminus 0}$

[56] And indeed, the behaviour of the classifier is very likely still *extrapolatory*: see [2].

[57] As an extreme example, a classifier perfectly trained to assign the animal species to photos of animals will still classify *non-animals* as a particular species of them, by visual similarity or – most often – due to the effects of confounders.

[58] In the sense of *natural*, *non-adversarial* examples.

[59] See, *e.g.* [6].

[60] Which may be given – as anticipated – *e.g.* by the norm of the perturbation. If such constraint is not into place, one may think of perturbations given by the difference of two legitimate input points: the resulting *perturbed input* is a *successful attack* by constructive definition, yet definitely not the intended goal.

and the norm $||\cdot||$, is thus any point $\boldsymbol{x^\star} = \boldsymbol{x_0} + \boldsymbol{p}$ *s.t.* $\mathcal{N}(\boldsymbol{x^\star}) \neq \mathcal{N}(\boldsymbol{x_0})$[61] and $||\boldsymbol{p}|| < \epsilon$ for some given $\boldsymbol{p}$, and *legitimate* input $\boldsymbol{x}$.

The problem can now be decomposed as follows. For each among some ϵ-balls centred at *knowingly unperturbed* datapoints (within the input space, and defined by $||\cdot||$), the optimal $\boldsymbol{p}$ is determined according to some *loss* quantifying the confidence in the success of the resulting *perturbed input* if used as an attack.

2.3.3.1 *White-box* scenario

In the *white-box* setting, the knowledge of gradients within the model generated by any input conceivable allows for *gradient-based* optimisation schemes, *mutatis mutandis* similar to those described for the tentatively-optimal choice of weights during training. In particular, the weights are kept fixed while the optimal perturbation (or, directly, *adversarial input*) is optimised *w.r.t.* to the uphill direction of the gradient induced by a *similarity loss* not in principle different from that used to train the network.

Of this kind, *e.g.* and absolutely not exhaustively, but of particular interest:

- The *FGSM attack*[62] (*i.e. Fast Gradient Sign Method*), which iterates just once along the uphill gradient sign direction, with a *displacement* in input space of norm ϵ.
- The *PGD attack*[63] (*i.e. Projected Gradient Descent*, reasonably to be considered an iterative extension of the *FGSM*. In particular, iterations of variable norm (not necessarily ϵ) are performed in the direction of the local gradient and proportionally to its modulus – starting from a *legitimate input* and for a given number of times; after each step, if the resulting point falls outside the ϵ-ball of interest, it is orthogonally reprojected on its border.

Of a different kind, *e.g.*, the *DeepFool attack*[64], instead, explicitly looking for the closest *perturbed inputs* at the intersection of the *input space* with the hyperspace orthogonal to the binary[65] *one-vs-all* boundary of the input class, still within the given norm constraint.

2.3.3.2 *Black-box* scenario

In the case where *model gradient* information is missing or unavailable, the success rate of attacks is reduced *w.r.t.* their *white-box* counterparts. However, still effective techniques rely on:

- *Gradient-free* optimisation schemes (*e.g.* genetic programming, *reinforcement learning* approaches) with actual success rate (or any *proxy* metric) as the function to be maximised;
- Machine learning systems aimed at directly sampling from an *adversarial region* of the input space, whose identification is delegated to an additional model to be trained on similar (or

[61] This is indeed the *untargeted* version of an attack. The *targeted* equivalent is obtained by replacing the inequality with an equality to the *target class*, different from the original. The analysis will focus on the former, with the latter always easily obtained with minimal modifications therein.

[62] Introduced in [22].

[63] Introduced in [44].

[64] Introduced in [50].

[65] In the case of *untargeted* attacks; additional care should be exercised in case of *targeted* ones.

the exact, if available) inputs to those employed for the attacked model. Of this kind, *e.g.*, *GAN-based*[66] attacks – which can even be used in a *white-box* setting, by variously providing the additional information to one or either the two networks involved in *GAN* training (of this type – both *black* or *white-box* – *e.g.* `AdvGan`[67]).

- *White-box* attacks targeting surrogate models (or mixtures thereof) of the one to be attacked. Such surrogate models may be obtained by *re-training* them on the same (or similar) dataset used for the model actually under attack[68] – or directly (and expensively) derived just from the *input/output* pairings produced by it.

2.3.4 *How it's made:* Defences

While research of *attacks* is fundamental in advancing the field – with better understanding of the actual dynamics determining such vulnerabilities, and to promote *offensive security* of machine learning systems – the *holy grail* of *adversarial robustness* research is to produce models, training algorithms, and inference protocols able to counteract such weakness, for the development of more trustworthy *AI*, and *for the greater good.*

Equally extensive, but often relying on much deeper knowledge and elaborate procedures *w.r.t.* that of *attacks*, the field of *adversarial defences* is in continuous evolution and rooted on an always renewing body of intuitive, empirical, and formal knowledge.

The main avenues of successful, practical developments in the field are the following:

- *Adversarial training*[69]. By large margin the most used approach – consisting in augmenting the training set of a model (of given weights, potentially at any point along its training) with *adversarial inputs* generated according to specific attacks (*e.g.* from the set right discussed) but endowed with their original label, and continue training on the augmented dataset. Expanding the *on-manifold* regions where the classifier works as expected with ever new *perturbed* datapoints where they are most needed, and providing a *true label* for selected points *off-manifold*, it is one among the few techniques with full applicability to any already existing architecture. This comes at the cost of the necessity for *model/dataset-specific* re-training, and much increased reliance on the correct *threat modelling* choices to mirror the (even *unforeseeable*) threats to be faced – though transferability of defences is partially possible across models, datasets, and attack types/strength.
  In such regard, specific attacks are of particular interest – so-called *universal*: those potentially able to converge within any point of the ϵ-ball of choice (as opposed to those operating only

[66] Where two deep neural networks learn each to perform a different task: one must map random-noise samples into *realistic-looking* images, while the other must discriminate whether an input is sampled from an unperturbed training set or produced by the other network. In a turn-based, *minimax* game, the *generator* becomes incrementally better at its task, under the improved capabilities of the, improving too, *discriminator*. In the use of similar schemes for *adversarial attack generation*, information about the class put out by the *target classifier* should also be included somehow in the loss of the generator. For an *in-depth* overview of *GAN*s, see [21].

[67] See [68].

[68] Of this type, *e.g.* the solution by Florian Tramèr to Aleksander Madry's MNIST Adversarial Examples Challenge for the *black-box* setting, 2017 – still the second-best even after 5 years of open leaderboard.

[69] Originally proposed as a technique in [22].

at a fixed displacement from the original input, or showing preferential directions), and thus able to theoretically produce the maximally representative *perturbations* for the model to learn from within adversarial training. *E.g.* the *PGD attack* is *universal*; *FGSM* it is not, due to the fixed displacement of exactly ϵ.

Importantly, *adversarially-trained* models can still be attacked with the same techniques (indeed just their weights have changed, not the threat model): the defence is meant to reduce their effectiveness, and on average increases the time or number of iterations required to find a successful perturbation.

Finally, multiple *adversarial attacks* (in the sense of different types, strength, even threat model) may be used within the adversarial training of the same network. This process – though effective – comes to terms with what is indeed a training on a much larger dataset of examples, sometimes carrying conflicting perturbations *w.r.t.* to the optimal way to update model parameters.[70]

- *Adversarial detection*. Framed as the solution of a binary classification problem, this approach aims at just *detecting* whether (or with what probability) an input is *adversarial w.r.t.* a given model, inviting the user to caution (or rejecting the output right after) in case it probably is. Typical approaches tailored toward *anomaly detection* may be employed, with very little domain-specific adaptation. *E.g.* the *discriminator* of a *GAN* of the type described in previous-page footnote can be used for the purpose.

  Though a potentially interesting track to pursue, such techniques do not attempt to *solve* the problem: they seek to make it manageable. Indeed, they are inherently susceptible of *denial-of-service* attacks – by adversarially targetting the *detector* to always raise a flag, extending *attack surface* of the system, and carrying no guarantee of success.

- *Adversarial purification*. Similar in goals to *adversarial training* (*i.e.* directly attempting to ensure a correct classification of *adversarial attacks*) and in tools to *adversarial detection* (*i.e.* delegating the task to a different system than the attacked), such class of methods aims at *recovering* from the eventually *perturbed* input its original, *clean* version – by the development of a unified mapping of *clean* inputs into themselves, and of *adversarial* ones back across the additive perturbation. Such technique, similar in framing to more typical *denoising* or *controlled editing* tasks, may be performed in a fashion completely independent from the *true labels* of data, thus being usually free from the burden of sometimes required additional labelling efforts. Supervised, self-supervised (even *energy-based*) and weakly-supervised alternatives in the same spirit do further exist. *Encoder/decoder* architectures, and conditional generative models are dominant within this class of approaches.

  Due to the lack of specificity to the attack, or reliance on *true labels*, these defences can become very creative. As illustrative examples:

  ⟶ `PuVAE`[71] obtains an increased robustness to *attacks*, without the need for *adversarial*

[70] An experimental evaluation of the process – and much more! – is given *e.g.* in [65].

[71] See [29].

training by first learning a *true class-conditional VAE* to map an adversarially-perturbed input and its *true class* to its purified version. Then, at inference time, the pair of *adversarial input / each class* is passed through the model and the resulting reconstruction bearing the most similarity with the perturbed image, is selected and finally classified.
⟶ The generator of `Defense-GAN`[72], instead, learns to generate tentative input reconstructions – subject to difference-norm constraints *w.r.t.* the perturbed input – while the discriminator learns to score the *adversarial likelihood* of the result. Still being one of the best-performing defences in both *white-box* and *black-box* scenarios, such device carries the cost of extremely lengthy training times, and much *longer than average* inference times too.

- *Inference-time defences*. Class of techniques containing various approaches trying to enhance resilience to *adversarial attacks* by adapting the inference protocol of the model. Inspired by *test-time augmentation* and still in their relative infancy, the most remarkable contributions propose to classify copies of the same input subject to standardised transformations (*e.g.* in the case of images: rotation, cropping, *etc.*) – provided that the model has been trained to correctly classify analogously-transformed *clean* inputs. Such approach usually – but inconsistently – decrease the success rate of adversarial attacks, unless the *adversary* succeeds in the much harder goal of producing correspondingly invariant attacks.

- *By leveraging a theoretically-robust structure*. Defences within this class – though even extremely different among themselves – all try to transpose into usable models some theoretical insights coming from an experimental or conjectural study of knowingly robust models *components.* A comprehensive theory of robustness is still lacking; nonetheless some results of such kind have been successfully obtained. Examples may include *Parseval Networks*[73], trying to link well established mathematical properties of the function described by a model (*e.g.* its Lipschitz constant) with its robustness – and inform consequently model architecture design; or *`kWTA` networks*[74], in which neuron-wise activation functions are replaced with layer-wise equivalents based on sorting functions and much more robust after a successful adversarial training (a result not to be taken for granted, in comparison with *traditional* identical counterparts).

- *By changing the rules of the game*. A whole topic of its own, proposing completely different architectures for models or their parts (*e.g.* based on *spiking neural network* models), or different training algorithms, based on different interpretations of the network parameters or representations. Of this latter kind, *Bayesian Neural Networks* – whose robustness against *gradient-based* attacks has been well established by *e.g.* [7], at the cost of much higher training complexity.

[72] Proposed in [58].
[73] See [8].
[74] See the already-mentioned [67].

2.3.5 On threat models, attacks/defences transferability, unforeseeability

From the outline just described, it is clear that *white-box*, *untargeted*, *perturbative* attacks represent the expression of a threat model much more conceding to the *adversary w.r.t.* their counterparts – and more dangerous for the user (or the deployer) of the model. It is additionally evident that a successful *black-box* attack will succeed also in the respective *white-box* setting (where, indeed, all the additional information would be simply not used in such case).

This may suggest the simplistic – and only partially consistent – conclusion that tackling *white-box, untargeted, perturbative* attacks is enough to foil them all.

While this may approximate truth from a statistical viewpoint – in the sense that a moderately large portion of *black-box*-attacked inputs are within reach in the corresponding *white-box* counterpart, and given the ability of a model specifically hardened against *white-box* attacks to foil many *black-box* analogues – there still exist *black-box* and/or *targeted* techniques based on wildly different – even unique – mechanisms, both effective and hard to defend against, even specifically.

Additionally, even with a restriction to the exact (and only) threat model to be addressed, one is left with the Herculean task of determining which specific attacks to defend against (in both type and strength of the maximum allowed perturbation): the struggles are similar to those described right above for the threat model of choice – with no clear-cut answers[75]. This may give an example on how difficult a *general solution* to the problem of adversarial robustness actually is, and on why even smaller progresses are cheered by the community. The only possibility we are left with is to choose a combination of *threat model/type of attack*, try to develop new strategies to increase robustness in such scenario (*i.e.* resulting in so-called *foreseen attacks/settings*), and finally assess if any improvement also extends to other settings (*i.e. unforeseen attacks/settings*).

In such light, we can justify the adoption and study of the *white-box, untargeted, perturbative* threat model (preferably if resulting in *universal attacks*) as a reference – the hardest to face – but definitely not the exclusive in order to (try to) solve the problem of *adversarial vulnerability.*

[75] But, as briefly discussed earlier, *threat models* foreseeing *universal attacks* should be favoured, by theory.

3 Aiming at robustness, guided by neuropsychology

The following section – definitely shorter than the previous in spite of a reverse allocation of efforts necessary for it to be ultimately realised – will be dedicated to the description of `CARSO`[76] (*i.e. CounterAdversarial Recall of Synthetic Observations*) – a novel technique of *defence to adversarial attacks*, synergistically spanning the *adversarial training*, *purification*, and *inference-time* subclasses. Additionally, the technique allows for fully-unsupervised training (*i.e. class-label*-less), but provided a dataset of known (unlabelled) perturbations or an *attack generator* is available when starting from a pretrained adversarially-trained model – achieving even greater robustness against *foreseen* attacks, and strong robustness against *unforeseen* attacks the sole pretrained model is completely unable to foil.

The method has been developed as a defence against *white-box*, *untargeted* attacks towards image classifiers; the applicability to other settings within image classification is reasonably assumed, but still pending, planned verification – along that within other fields of application.

3.1 Neuroscience as a guiding inspiration

It is safe to say that the core ideas behind this work – and even its goal and field of application – definitely did not came about as such, but were gradually *discovered* along the way[77].
The starting point was a loosely defined intention to exploratorily investigate the liminal space across *artificial intelligence* – and specifically *machine learning* with *artificial neural architectures* – and *cognitive-to-computational neuroscience.*

Indeed, the origins of the field of *deep learning* – then represented by the study of *local* and *emergent* properties of essentialised mathematical models of interacting neurons, their interrelation and capabilities *w.r.t. learning* – shared with neuroscience the deep and ambitious goal of modelling in some way activities pertaining to *brains*, no matter how complex or *intelligent*, with the aim of *doing* the former, and of *understanding* the latter.

Decades later, the gap has widened – with clear (and often healthy, if within limits) differences in methods and mission – not rarely still spurring the debate about which degree of interconnection is the one to be wished for. And in the light of the extraordinary successes in *deep learning* –, even the most advanced of its systems is still unable to even remotely approach some peculiar cognitive features of animals or humans, let alone with the ease of those living organisms. And still, they may be the only practical realisation of the systemic behaviour some within the deep learning community are trying to replicate *in silico* from a different viewpoint.

Among the wide array of cognitive phenomena qualifying for the properties just outlined, that of *higher intelligence* in humans or primates, up to *cognition* and *consciousness*, is definitely the

[76] The subtext originates from the Italian name of the *Karst Plateau* region stretching across the Italian-Slovenian border, and whose colours, lovely hikes, and sober austerity is loved by all those who have visited – let alone lived, or still living within! – the Trieste area.

[77] In an unplanned corroboration of Kenneth Stanley's core tenet; see [64].

most striking and – not surprisingly – out-of-reach. No settled, *experimentable*, definition of it has been even produced! Regardless, armed with just a tentative, blurry, definition of *consciousness as 'thought about thought'* we fast approached a *sanity check*: is this even expressible in terms of the current *deep learning* framework?

3.1.1 Learning by *recall* and *self-introspection*

Losing much of the *deepness* in our thoughts, we finally focused on a very specific *flavour* of the definition we had in mind earlier: the *recall* of acquired memories – specifically, during the process of learning, so to slightly close the already very large gap with deep learning practice.

Even simply by *self-introspection* – but definitely also within published psychological and neuropsychological literature – it appears clear that *recall* of previously memorised information is crucial in (not only) learning processes across a wide variety of animal models, and in humans. In such regard, and with not even the slightest intention of completeness, see *e.g.* [47] and [72], about the peculiar phenomena of *forward testing* and *repeated testing*, potentially rooted in the explicit awareness that currently acquired, new, information may be necessary for future recall. Or, more generally, about the role of hippocampal dynamics in learning and memorisation of spatial information, in [4]. Finally, for much broader-encompassing treatise on *learning*, see published book [10].

From which, the idea to directly utilise the representation of (part of) an artificial neural network – as a rough approximation of *liminal* memory within a deep learning model – to inform the training of itself (or another portion of it) – with focus on using it as an *adversarial defence.*

The final result of the perambulation that ensued is reported in the following.

3.2 `CARSO`

3.2.1 Problem & solution statement

`CARSO` is a novel deep learning *model architecture* and associated *training* and *inference* methodology – designed to increase the robustness of image classifiers against *gradient-based perturbative adversarial attacks*: both *foreseen* and *unforeseen*, and to do so better than *iterative adversarial training*[78] according to *top-1 accuracy under attack*[79] evaluation, in both scenarios. The model architecture has been designed as generally independent from specific hyperparameter choice[80], which can be freely optimised according to the specific problem of interest.

Finally, since relying on the adversarial training of the classifier of interest, against *foreseen* attacks, such classifier can be directly provided as an *adversarially-trained*, *pre-trained* model – without preventing the training of further parts of `CARSO` or full applicability of the training/inference

[78] *I.e.* where each batch of the training set in enriched with adversarial attacks against the current parameters of the model, after which such parameters are updated as usual, and previous attacks discarded to be re-generated right afterwards.

[79] The ratio of correctly classified adversarial attacks targetting the model itself, labelling its input with the predicted most probable class.

[80] To the optimisation of which as been – indeed – dedicated little time and effort.

protocols. In such occurrence, no labelling efforts whatsoever are required to the *trainer* of CARSO.

3.2.2 General intuitions

The main intuitive considerations guiding the development of CARSO are synthesised along the following *thought flow*:

- At inference time, being all neurons governed by deterministic operations, the representation formed within each of them in a *feedforward* model must deterministically depend from those of the neurons in the previous layer, and the weights associated with the connections within the two layers. Such reasoning may be iterated straight from the scalar values part of the input (hypothetically part of an *input layer*) up to the final layer of the network (whose neuron with the largest representation determines the output class).
- Weights are fixed at inference time.
- If a given input is correctly classified by a *feedforward* classifier, but a perturbative adversarial attack starting from such input is performed and succeeds, then a *small* perturbation in the *input layer* must have been somehow preserved along the network until the *output layer.* In this respect, [12] has provided key insights.
- As a consequence, the ordered representation of all (potentially; less may be sufficient) neurons in the network must carry sufficient information to identify signature *pathways* activated by any *clean* or *perturbed* input datapoints.
- Such ordered representation can be used as input to a further model, with *e.g.* the goal of providing *adversarial detection.*
- Any *gradient-based* attack targetting such further *detector* must be able to produce a perturbation in the classifier input that – within the strength constrains of such input – jointly not only produces a *misclassification*, but also a (potentially unbound in strength) perturbation of the representation (considered as input to the *adversarial detector*) fooling the latter.
- The gradients computed along the computational graph associated with the original classifier up to its output *w.r.t.* the chosen *classification loss* and those tracked along the *detector* through the classifier representation are fundamentally different at the neurons of the classifier. Thus, any attacker attempting to do so, must at least optimise for two competing objectives – with no guarantee of success in both simultaneously. Published literature is completely lacking results pertaining the existence (or not) of such solutions.
- A similar result would be even more favourable in the case of *adversarial purification.* This can be obtained if the ordered representation of the classifier, produced by the (potentially perturbed) input, is used to *condition* the purification process without interest in the actual output class. Then, the purified input is classified by the same classifier having produced the representation – in order to ensure *competing gradients* in the classifier.

- If the information contained in the representation of the classifier is sufficient to even recover the input having produced it (and indeed it is possible, by *just* approximately inverting the function described by the first layer of the network, from its representation!), the same device can be used to generate an entire distribution of candidate *purified* inputs – by replacing the code associated with the input with a structured random sample at inference time. This is exactly what a *conditional variational autoencoder* does.

- Such latter sampling is also beneficial to the overall robustness of the system – in case of specially-crafted attacks against it. By classifying a randomly sampled set of *purified* inputs (instead of only one) and aggregating the resulting classes, (non-adversarial) robustness to reconstruction noise or blur[81] is increased dramatically. A hypothetical attacker should not only be able to solve the multi-objective optimisation problem described above: it should do so robustly *w.r.t.* generative sampling.

- Adversarial training of the original classifier does not interfere with the procedure just outlined, but it even strengthens it. Not only the number of successful adversarial attacks decreases for the classifier (thus resulting in an easier learning of the denoising process), but it also provides a further safeguard should the *purification* be *leaky*.

Such ideas – with additional minor *tricks* required to work around some training difficulties or *corner cases* are directly translated into the description of the methodology that follows.

3.2.3 Training protocol and architecture

The training protocol of `CARSO` is described schematically below, together with architectural elements required at each step. Illustrative diagrams are provided in the appendix.

The classifier of interest $\mathcal{N}$ – whose robustness has to be increased – is *adversarially trained* according to a pre-specified threat model (resulting in *foreseen attacks against the classifier*), until full convergence. This phase requires both an (or more) *attack generation mechanism* of choice and a labelled *clean* dataset. Alternatively, an *adversarially trained*, *pre-trained* classifier can be used: in such case, however, the threat model is not fully controllable.

Inputs (to $\mathcal{N}$) from a *clean* dataset are then perturbed according to a pre-specified threat model against the adversarially trained $\mathcal{N}$ from previous step (resulting in *foreseen attacks for purification*). Differently from previous step – though a *clean* dataset is still required – no labelling is necessary, nor it is the actual success of the attacks. In case a *pre-trained* $\mathcal{N}$ has been used, adversarial attacks directed at it may also be readily available: in such case not even an *attack generator* is necessary, at the further cost of fully losing control over the threat model; the corresponding *clean* datapoints of each attacked input must be available, though.

The ordered[82] representation of the classifier is extracted and used as conditioning set of a *conditional*

[81] Which is a typical phenomenon in such architectures.

[82] The actual order is not relevant, provided it is persistent *w.r.t.* the locations of the same neurons within the network.

variational autoencoder (of encoder $\mathcal{E}$ and decoder $\mathcal{D}$) acting as a *purifier*. By encoding inputs picked from the *clean* dataset, augmented with the *foreseen attacks for purification* – and conditioning on the ordered representation – a map to the corresponding *clean*[83] input is learned. The model could be specifically trained as any other cVAE.
In case the dimensionality of the input needs to be rebalanced against that of the classifier representation (or viceversa; a procedure usually done to improve the convergence properties of the cVAE and speed up its training), an *end-to-end* approach has to be preferred. Dimensionality-reducing *pre-encoders* $\mathcal{E}_{\text{DRI}}$ for the input and $\mathcal{E}_{\text{DRR}}$ for the representation can be directly placed on top of the cVAE and their training jointly performed.

At the end of such training, only trained $\mathcal{E}_{\text{DRR}}$ (if any) and $\mathcal{D}$, and the adversarially trained $\mathcal{N}$ are required for inference.

3.2.4 Inference protocol

The inference protocol of CARSO is now described schematically below. Illustrative diagrams are provided in the appendix.

On arrival of a new input, this is evaluated by the *adversarially-trained* $\mathcal{N}$ with the aim of obtaining only its ordered representation (*i.e.* there is no interest in the actual prediction, which can be safely discarded).

The representation just extracted is concatenated with a random sample extracted from the latent distribution implied during the training of the cVAE, and passed through $\mathcal{D}$ producing as output a candidate purification of the input. Such process can be repeated an arbitrary number of times, thus generatively sampling from the posterior distribution of purified inputs associated with the original one.

The resulting collection of purified inputs is – each separately – classified by the same *adversarially-trained* classifier $\mathcal{N}$, thus resulting in a distribution of classes.

The resulting *mode class* may be used as the actual output of the system.

3.3 Experimental evaluation

The architecture and protocols proposed have been assessed on a prototypical, self-developed benchmark test aiming at the evaluation of *top-1 accuracy under attack* in a simulated scenario with both *foreseen* and *unforeseen* perturbations.

Tests have been performed on the MNIST dataset[84], normalised to the $[0, 1]$ range and further standardised. The FGSM (constrained at $\epsilon = 0.15$ and $\epsilon = 0.3$ *w.r.t.* the $|| \cdot ||_2$ norm) and PGD (constrained at $\epsilon = 0.15$ and $\epsilon = 0.3$ *w.r.t.* the $|| \cdot ||_\infty$ norm) attacks have been considered as representatives of *foreseen* attacks – whereas DeepFool (constrained at $\epsilon = 0.15$, $\epsilon = 0.3$ and $\epsilon = 0.5$

[83] The *clean* input corresponding to a non-perturbed one is... itself!
[84] A dataset of small, square, greyscale digitisations of handwritten digits; see [39]

w.r.t. the $||\cdot||_\infty$ norm) and the stronger `FGSM` (constrained at $\epsilon = 0.5$ *w.r.t.* the $||\cdot||_2$ norm) and `PGD` (constrained at $\epsilon = 0.5$ *w.r.t.* the $||\cdot||_\infty$ norm) as those of *unforeseen* attacks.[85]

All iterative attacks (*i.e.* `PGD` and `DeepFool`) additionally bore constraints on the maximum number of iterations to be performed (fixed at 40 and 50 respectively) and on the size of per-iteration perturbation (0.01 and 0.02 respectively).

As far as any of the *deep learning* models are concerned, only *fully-connected feedforward* architectures have been considered due to their simplicity and the still underdeveloped study of the proposed technique. In order to reduce the dependency from hyperparameters, the number of neurons in *hidden* layers have been fixed at that interpolating the two adjacent – with the only exception being the classifier. This reduces the description of layer sizes along networks to that of the input, output, and the overall number of *hidden layers*.

The classifier was built as a *FCN* variation upon *convolutional* neural network `LeNet5`[86], with an input size of 28×28 (the size in *greyscale* pixels of the image), an output of 10 (corresponding the ten digit classes) and hidden layers of sizes 200 and 80. Each layer, with the exception of the last one, is preceded by 0.15 probability *dropout* and *batch normalisation*. The innovative `Mish`[87] activation function has been the *non-linearity* of choice – due to its effectiveness in expressing flexible mappings within even smaller models; the model has been trained via output-*softmax*ed *categorical cross-entropy* minimisation.

The resulting representation consisted of 290 scalars. The latter, and original input, have been pre-compressed by *2-hidden-layer* networks with *batch normalisation* and no *dropout* before hidden layers, with those using 0.1-*steep* `Leaky ReLUs`[88] as *non-linearities* – to respectively $1/5^{th}$ and $1/4^{th}$ their original size, before being finally being shrunk through a sigmoid.

The encoder of the `cVAE` consisted in a further *1-hidden-layer* network of the same kind – with hyperbolic tangent shrinking, followed by two independent linear layers sharing the same input to finally produce 36 means and standard deviations of independent Gaussian distributions.

The decoder mapping the conditioning set and the sample back to input space is a *2-hidden-layer* network – of the same kind as the *pre-compressor*, without *Batch Normalisation* before the last layer. A further sigmoid at the end ensures a correct *input coding* in the pixelwise $[0, 1]$ domain. The reconstruction loss for the `cVAE` has been *binary cross entropy*, as a more amenable alternative to $||\cdot||_2$ for inputs bound within $[0, 1]$.

The *unperturbed* classifier as a baseline, and both the adversarially trained and the `cVAE`, have been

[85] As a comparative note related to the strength of the perturbations, strengths above $\epsilon = 0.3$ are usually considered unrealistic, in the context of the `MNIST` dataset, due to easy detectability by the naked human eye. Bearing that in mind, the peculiar `DeepFool` has been selected as *unforeseen* due to the unique approach at eliciting vulnerabilities in the attacked model; $\epsilon = 0.5$-bound perturbations as deliberately *extreme* attacks to test the upper limits of the defence.

[86] See [38].

[87] A recently-proposed *smooth*, non-monotonic *non-linearity*, expressible as $x\ \tanh(\text{softplus}(x))$, whose popularity has steadily increase especially within the *computer vision* community. See [49] for a much more extensive analysis and experimental evaluation.

[88] See [70].

trained by the `RAdam` optimiser (with fixed $\beta_1 = 0.9$, $\beta_2 = 0.999$ and numerical stability constant $\varepsilon = 10^{-8}$), a starting learning rate of 0.05, 0.05 and 0.001 respectively – with further scaling by a factor 0.6, 0.6, and 0.7 upon first after 10-epochs window, since previous *learning rate* reduction, resulting in no overall loss decrease. A hard limit was liberally put at 300 epochs in total to ensure full convergence.

The number of purified samples at inference time was fixed at 1500, and aggregation performed by *mode*.

3.3.1 Results

Following the protocol described, and the specific `CARSO` architecture detailed by the *hyperparameters* just reported, our proposal has been compared with `IAT` resulting in the following table 1.

Attack (*type*) / Defence (*adv. acc.%*)	None	`IAT`	`CARSO`
None	**98.40**	97.17	96.72
FGSM $\|\|\cdot\|\|_2$, $\epsilon = 0.15$	12.09	91.89	**93.62**
FGSM $\|\|\cdot\|\|_2$, $\epsilon = 0.30$	01.21	76.94	**86.43**
(U) FGSM $\|\|\cdot\|\|_2$, $\epsilon = 0.50$	01.00	12.29	**13.59**
PGD $\|\|\cdot\|\|_\infty$, $\epsilon = 0.15$	01.60	90.54	**93.44**
PGD $\|\|\cdot\|\|_\infty$, $\epsilon = 0.30$	06.85	71.26	**86.27**
(U) PGD $\|\|\cdot\|\|_\infty$, $\epsilon = 0.50$	*20.66*	*11.67*	**38.38**
(U) DF $\|\|\cdot\|\|_\infty$, $\epsilon = 0.15$	00.66	90.25	**95.06**
(U) DF $\|\|\cdot\|\|_\infty$, $\epsilon = 0.30$	00.00	60.54	**93.31**
(U) DF $\|\|\cdot\|\|_\infty$, $\epsilon = 0.50$	00.00	00.78	**71.34**

Table 1: Top-1 accuracy under attack against different types of adversarial attack (rows) directed at the FCN classifier, defended by two techniques (columns): Iterative Adversarial Training and `CARSO`. Unforeseen attacks – against both the classifier and the purifier – are marked by '(U)'; the best performing defence per given attack is emboldened.

3.3.2 Ablation studies

Ablation studies were performed in order to assess whether the most relevant modelling choices resulting in the final architecture and protocols of `CARSO` were justified *w.r.t.* a simpler, more *traditional* alternative.

Results are summarised in the following.

- On the necessity of *adversarial training* altogether. Tests were performed with the same architecture as that described, training all models on *clean* inputs only. The results – in the same setting investigated right above – showed a significant increase in adversarial robustness compared to the *clean* model, with accuracies under attack in the range of 0.15% $\sim$ 0.25% for the classifier and 15% $\sim$ 25% for `CARSO`. Despite indicative of some degree of success, such approach was discarded as markedly unsatisfactory *w.r.t.* other simpler and more popular adversarial defences. Noteworthy the fact that – in some rare cases – such approach still managed to obtain a robustness comparable to or greater than *iterative adversarial training*. Similar attempts *w.r.t. partial* adversarial training (*i.e.* of the classifier or the `cVAE` alone) were performed, resulting in a slight increase in *accuracy under attack* in comparison to the entirely non-adversarial training; still, similar considerations applied, due to the lack of competitiveness with simpler alternative defences.

- On the number of purified samples. Sweeps across three orders of magnitude have been performed, with a sample size under 700 showing unsatisfactory results due to intolerable reconstruction noise. Detectable increases were present until $\sim$ 2000, becoming less significant after 1500.

- On the number of layers in the `cVAE` networks. Avoiding by design networks with no *hidden layers*, the number of them in those carrying > 1 have been – indeed – increased starting from 1 and with 1-increments as part of the development process, until the shallowest satisfactory network was reached.

Structured attempts at further *hyperparameter* optimisation, different kinds of adversarial attack for the *foreseen* and/or *unforeseen* cases, different image datasets (including changes in subject and/or colour space), or different types of data have not been performed – but are planned within the ongoing exploration of such defence technique.

3.3.3 Discussion

From the comparison of the accuracies reported above, it is possible to firstly see that – further along the lines of what occurs in the case of *iterative adversarial training*, and *adversarial training* in general – the newly proposed defence technique imposes an *accuracy toll* in the case of *clean* inputs. Being exactly the same *iteratively adversarially trained* classifier shared among `CARSO` and `IAT` approaches – and even considering the effects of purification noise/blur alone, it is expected that such *accuracy toll* is in the case of `CARSO`, even if marginally, greater *w.r.t.* `IAT`.

On the other hand, if we consider even just the results on against *foreseen* attacks, it is evident a pervasive and much more remarkable *accuracy under attack* gain by `CARSO`, which increases with the strength of the attack. Such peculiar phenomenon will be further discussed, in the light of what follows. In the case of strong *universal attacks*, `CARSO` attains more than an $1/5^{\text{th}}$ incremental accuracy gain. In no case among those tested, `IAT` fared better than `CARSO` against *foreseen* attacks. At this point – and further clearing the floor form the considerations about the the high resilience of `CARSO` against directed attacks – whether the additional time required for the training of the `cVAE` machinery, and the latency penalisations at inference determined by the repeated sampling process and classification, are justified by the more robust results is still up to debate.

The area in which `CARSO` shines the most is yet to be analysed: robustness towards *unforeseen* attacks. Though very marginal in the case of strong, *unforeseen* `FGSM` attacks (still producing an increase of around $1/12^{\text{th}}$), performance against strong `PGD` attacks and any `DeepFool` perturbation is very solid. While in the case of the weakest `DeepFool` attack the *accuracy under attack* gain is significant but ultimately modest *w.r.t.* `IAT`, the remaining strengths show its highest.

While the overall increased robustness shown by `CARSO` is definitely the effect of a synergistic interaction among the two proper *models* involved (*i.e.* the adversarially-trained classifier and the `cVAE`, which are – indeed – self-standing *adversarial defences* on their own) – in retrospect it is further possible to hypothesise a deeper insight into the *credit assignment* between the two cooperating strategies, and across the cases considered.

By noticing how – by construction – the `DeepFool` attack favours perturbations resulting *off-data-manifold*, and by correlating the much increased success in defending against such attack to the addition of the `cVAE` to an already *adversarially-defended* classifier, one may suggest that:

- The adversarial training of the classifier – with the highest density of perturbations close to the *data manifold*, due to both accidental choice of *foreseen* attacks and peculiar training dynamics – mostly helped in reducing the *local intrinsic dimension*[89] of decision boundaries between *on-manifold* classes.
- On the other hand, the addition of an encoder/decoder model tasked with the purification of adversarial inputs directed towards the same *adversarially-trained* classifier, may have mostly compensated the expansion along the *co-dimensional* submanifold, within input space. The increase in *accuracy under attack* offered by the addition of the `cVAE` with laxer strength constraints may indeed corroborate this hypothesis: within a smaller allowed displacement radius in input space, the *data manifold* may contain a close-enough moderately effective result of adversarial perturbation. However, it is *out-of-manifold* that the training set (of purely adversarial examples, used during training) is more disperse – and the effect of untargeted, competing perturbations lying on the surface on the *clean*-input-centred cone may easily cancel out in the long run. If allowed, a stronger attacker will probably find there a much more optimal attack.

The potential development along such final consideration is definitely interesting both from the viewpoint of pure *comprehension* of the phenomenon of adversarial attacks and robustness – but also for the practical development of actionable defences able to respond differently in case of different threats.

With respect to the last point, the results of the strongest (and unforeseen) `PGD` attack my be a motivating example: the robustness of the `IAT`-trained model is decreased *w.r.t.* the *clean* classifier. An explanation may depend on the learning capacity of the model being undersized for a the learning of all given examples within arbitrary tolerances: this induces competition among different examples within the model. And while the optimisation problem tackled by a gradient-based *attacker* is – as in the training process of a neural model was – poisoned by the same susceptibility to local convergence, such competition may have *involuntarily* smoothed the loss landscape for the *enemy*.

[89] See [6] for a much more in-depth analysis of the phenomenon – and definitely more!

4 Conclusions

In the work which is now concluding, we moved from the genuine curiosity about how the study of elusive cognitive phenomena may inform the development of ever more capable *deep artificial neural networks*. Along the way – we described the development of `CARSO`, a potentially promising, still *embryonic*, novel technique to provide reliable *adversarial defence* against *gradient-based foreseen* and *unforeseen* attacks targeting a *fully-connected feedforward image classifier* – and we positively assessed its effectiveness against a most relied-upon, alternative methodology.

Though coming at an increased cost in terms of *clean input* accuracy and training time (making it probably not the first choice for realtime, continuous-acquisition applications), `CARSO` is additionally able provide – as an *add on* to previously *adversarially-trained* classifiers, and without relying on additional labelling – *innate*, close to *clean* accuracy to perturbed inputs with a strong *off-data-manifold* component, and so self-defend against *gradient-based* attacks directed specifically at it.

Most importantly, it was able to provide further – deeper and enthralling – questions waiting to be played with, both down along the way towards the development of safer, less exploitable *deep learning* and within the *fertile crescent* at the confluence of the *biological* and the *mathematical* approach to the study of intelligence, in all its possible *hues*.

4.1 Future work

The contribution provided by this thesis is indeed minimal with such an ambitious destination in mind; yet it could be the starting point down potentially many avenues just discovered.

4.1.1 Incremental experimentation

Surely, the experimental evaluation of `CARSO` – across a wider variety of threat models, datasets and tasks, *adversarial attacks* and variations of the same protocol, is the most needed and immediate pursuit required to further understand and eventually *productionise* the technique.

4.1.2 `FiWAGR`[90]: *Filtering via Weight Agnostic Gradient Randomisation*

Furthermore, the leverage of *competing gradients* within model representation – at the basis on an increased defensibility against *directly-targeted* attacks – could inspire additional, independent evolutions towards *theoretically-guaranteed full gradient-based defences*. Of this kind, a for now only hypothetical, still unexplored, neural architecture containing *gradient-stopping weight-agnostic* layers[91] able in theory to provide a randomised gradient independent of model functionality – and converging to zero in expectation without resulting in *gradient masking* for individual samples.

[90] To be pronounced as *'figure'* – in a stretch, will it ever be possible: both to be carried out as a work, and/or to be pronounced as such.

[91] First theorised and experimented with in [18], but whose perceived interest waned over time within the *deep learning community*.

4.1.3 *Moonshot* goal(s)

Finally, still with the *original* goal in mind and potentially deviating from the riverbed of the work until now illustrated, more *direct* approaches towards *biological/artificial intelligence convergence* may be taken. *E.g.* by analysing the actual structure on representations along *neural models* and *neurobiological pathways* (*e.g.* the *ventral stream*) of live animals during the actual process of learning[92]. At this stage, we still do not know if such path will be walkable, or where it would eventually lead: maybe to nowhere, maybe to ever new questions, whose growing abundance may incidentally substantiate the same act of *thinking* we started interrogating about.

[92] Such fascinating direction of research has been suggested by Fabio Anselmi during the last *Neuroscience & Statistical Physics Workshop*, held at SISSA in late Spring 2022.

Bibliography

[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, July 2018.

[2] Randall Balestriero, Jerome Pesenti, and Yann LeCun. Learning in high dimension always amounts to extrapolation, 2021.

[3] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018.

[4] Chris M. Bird and Neil Burgess. The hippocampus and memory: insights from spatial processing. *Nature Reviews Neuroscience*, 9:182–194, 2008.

[5] Shahin Boluki, Randy Ardywibowo, Siamak Zamani Dadaneh, Mingyuan Zhou, and Xiaoning Qian. Learnable bernoulli dropout for bayesian deep learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3905–3916. PMLR, 26–28 Aug 2020.

[6] Luca Bortolussi and Guido Sanguinetti. Intrinsic geometric vulnerability of high-dimensional artificial intelligence. *ArXiv*, abs/1811.03571, 2018.

[7] Ginevra Carbone, Matthew Wicker, Luca Laurenti, Andrea Patane', Luca Bortolussi, and Guido Sanguinetti. Robustness of bayesian neural networks to gradient-based attacks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15602–15613. Curran Associates, Inc., 2020.

[8] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 854–863. PMLR, 06–11 Aug 2017.

[9] Joel Dapello, Jenelle Feather, Hang Le, Tiago Marques, David Daniel Cox, Josh Mcdermott, James J. DiCarlo, and SueYeon Chung. Neural population geometry reveals the role of stochasticity in robust perception. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[10] Stanislas Dehaene. *How We Learn: Why Brains Learn Better Than Any Machine... for Now.* Penguin Random House, 2018.

[11] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[12] Diego Doimo, Aldo Glielmo, Alessio Ansuini, and Alessandro Laio. Hierarchical nucleation in deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7526–7536. Curran Associates, Inc., 2020.

[13] Zehao Don, Weinan E., and Chao Ma. A priori estimates of the generalization error for autoencoders. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3327–3331, 2020.

[14] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR, 17–23 Jul 2022.

[15] Simant Dube. High dimensional spaces, deep learning and adversarial examples. *ArXiv*, abs/1801.00634, 2018.

[16] Baruch Epstein and Ron Meir. Generalization bounds for unsupervised and semi-supervised learning with autoencoders. *ArXiv*, abs/1902.01449, 2019.

[17] Ian Fischer. *Dual-Number Methods in Kinematics, Statics and Dynamics*. CRC Press, 1st edition, 1998.

[18] Adam Gaier and David Ha. Weight agnostic neural networks, 2019. https://weightagnostic.github.io.

[19] Valeria Gazzola and Christian Keysers. The observation and execution of actions share motor and somatosensory voxels in all tested subjects: Single-subject analyses of unsmoothed fmri data. *Cerebral Cortex (New York, NY)*, 19:1239 – 1255, 2009.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[22] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[23] Ronan Hamon, Henrik Junklewitz, and Jose Ignacio Sanchez Martin. JRC technical report: Robustness and explainability of artificial intelligence. *JRC Publications Repository*, 2020.

[24] Donald Olding Hebb. The organization of behavior: A neuropsychological theory, 1949.

[25] John A. Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the theory of neural computation.* Santa Fe Institute Studies in the Sciences of Complexity. CRC Press, Taylor & Francis Group, 1991.

[26] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[27] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[28] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[29] Uiwon Hwang, Jaewoo Park, Hyemi Jang, Sungroh Yoon, and Nam Ik Cho. Puvae: A variational autoencoder to purify adversarial examples. *IEEE Access*, 7:126582–126593, 2019.

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[31] Patrick Kidger and Terry Lyons. Universal Approximation with Deep Narrow Networks. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2306–2327. PMLR, 09–12 Jul 2020.

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

[33] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[34] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *Aiche Journal*, 37:233–243, 1991.

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

[36] Anders Krogh and John Hertz. A simple weight decay can improve generalization. In J. Moody, S. Hanson, and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.

[37] Simon B. Laughlin. Matching coding, circuits, cells, and molecules to signals: General principles of retinal design in the fly's eye. *Progress in Retinal and Eye Research*, 13(1):165–196, 1994.

[38] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[39] Yann LeCun and Corinna Cortes. The MNIST database of handwritten digits, 2005.

[40] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[41] Hongzhou Lin and Stefanie Jegelka. Resnet with one-neuron hidden layers is a universal approximator. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[42] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020.

[43] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[44] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[45] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *ArXiv*, abs/1804.07612, 2018.

[46] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 52:99–115, 1990.

[47] Kathleen B. McDermott. Paradoxical effects of testing: Repeated retrieval attempts enhance the likelihood of later accurate and false recall. *Memory & Cognition*, 34:261–267, 2006.

[48] Marvin Minsky and Seymour Papert. *Perceptrons: An introduction to computational geometry, 2nd ed.* MIT Press, 1972.

[49] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 4:2, 2019.

[50] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

[51] Paolo Muratore, Sina Tafazoli, Eugenio Piasini, Alessandro Laio, and Davide Zoccolan. Prune and distill: similar reformatting of image information along rat visual cortex and deep neural networks. *ArXiv*, abs/2205.13816, 2022.

[52] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[53] neptune.ai. Neptune: experiment tracking and model registry. `https://neptune.ai/`, 2022.

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

[55] Jorge Pérez, Javier Marinković, and Pablo Barceló. On the turing completeness of modern neural network architectures. In *International Conference on Learning Representations*, 2019.

[56] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[57] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[58] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018.

[59] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 44(1.2):206–226, 1959.

[60] Guillaume Sanchez. Torchélie. `https://github.com/Vermeille/Torchelie`, 2022.

[61] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[62] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[63] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[64] Kenneth O. Stanley and Joel Lehman. *Why Greatness Cannot Be Planned: The Myth of the Objective.* Springer Publishing Company, Incorporated, 2015.

[65] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[66] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

[67] Chang Xiao, Peilin Zhong, and Changxi Zheng. Enhancing adversarial defense by k-winners-take-all. In *International Conference on Learning Representations*, 2020.

[68] Chaowei Xiao, Bo Li, Jun yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3905–3911. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[69] Cihang Xie, Mingxing Tan, Boqing Gong, Alan Loddon Yuille, and Quoc V. Le. Smooth adversarial training. *ArXiv*, abs/2006.14536, 2020.

[70] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *ArXiv*, abs/1505.00853, 2015.

[71] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[72] Chunliang Yang, Rosalind Potts, and David R. Shanks. Enhancing learning and retrieval of new information: a review of the forward testing effect. *NPJ Science of Learning*, 3, 2018.

[73] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

Appendix: figures

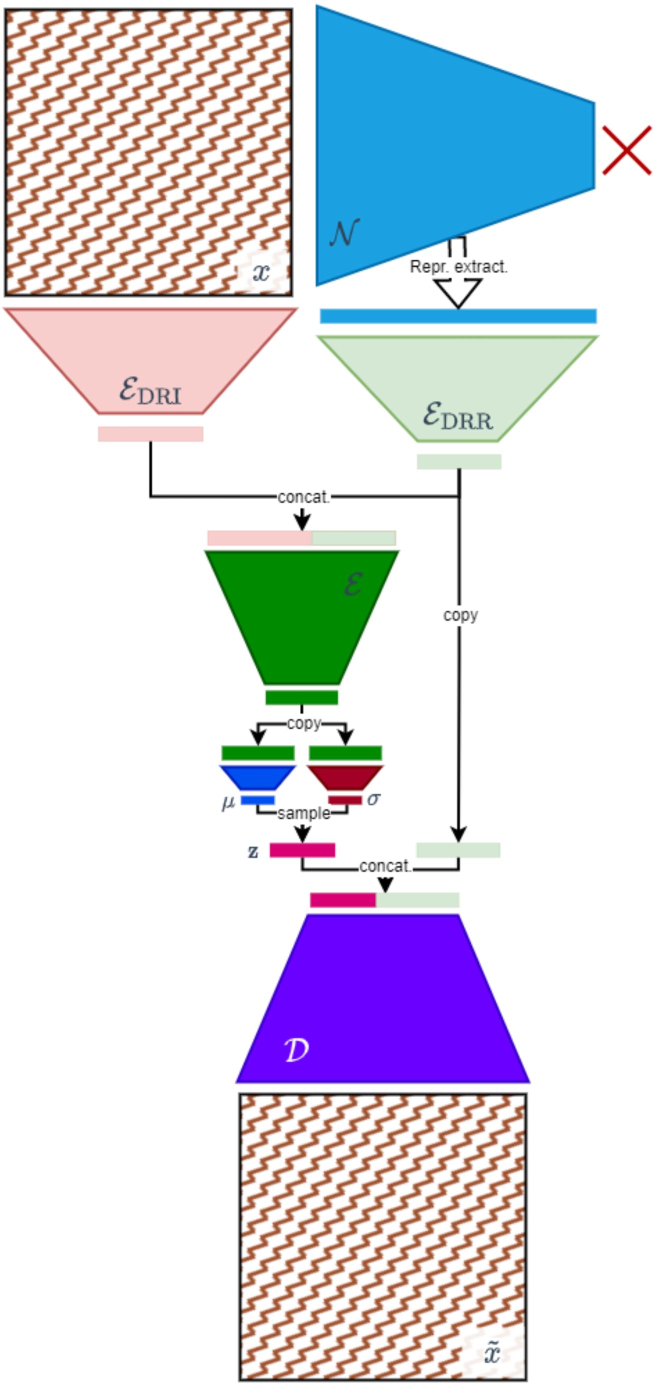

Figure 1: Essentialised `CARSO` *architecture at training time. Square elements denote inputs or their reconstruction (the same colours and graphical pattern are used regardless of implied similarity). The size of the elements is not necessarily in scale (an estimation based on typical scenarios has been made).*

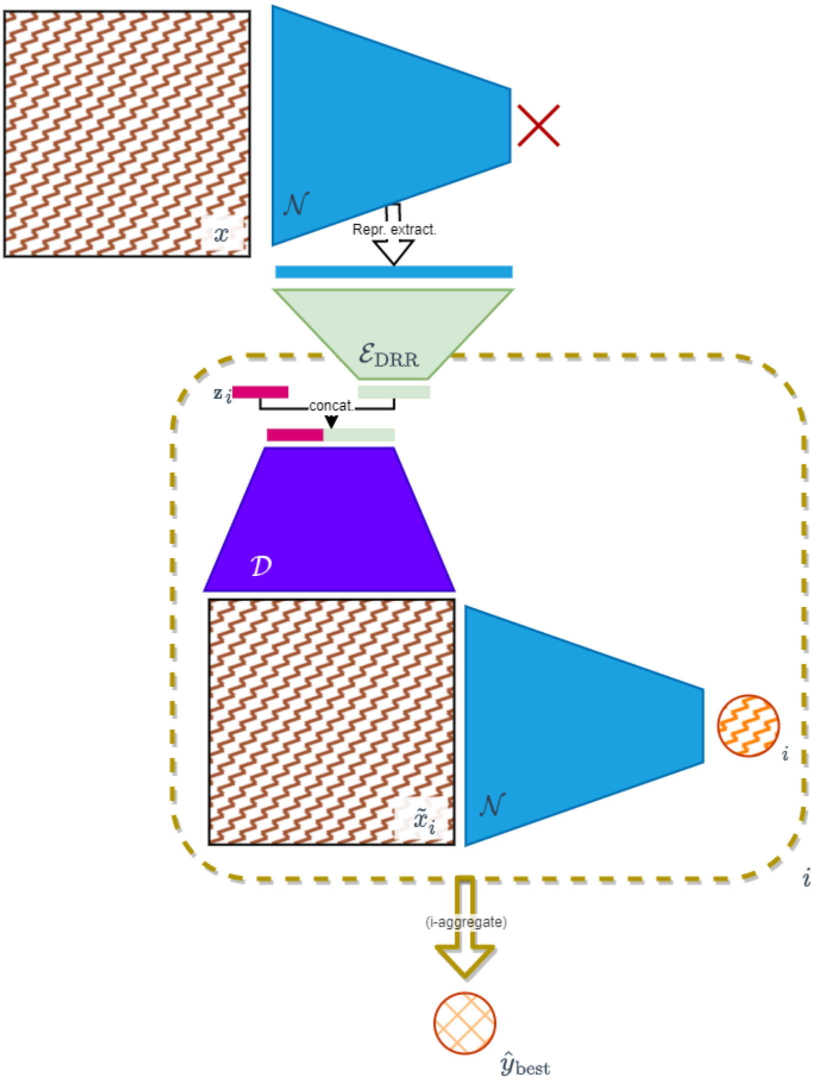

Figure 2: Essentialised `CARSO` architecture at inference time. Square elements denote inputs or their reconstruction (the same colours and graphical pattern are used regardless of implied similarity). Round elements denote the output class of the classifier taken into consideration. The region bordered by the dashed line constitutes the input-reconstruction sampler (followed by the classifier $\mathcal{N}$), whose use can be arbitrarily repeated, for fixed input $\boldsymbol{x}$. The size of the elements is not necessarily in scale (an estimation based on typical scenarios has been made).