



King's Research Portal

Document Version
Peer reviewed version

[Link to publication record in King's Research Portal](#)

Citation for published version (APA):

Tyukin, I., Higham, D., Bastounis, A., Woldegeorgis, E., & Gorban, A. (2023). The feasibility and inevitability of stealth attacks. *IMA JOURNAL OF APPLIED MATHEMATICS*.

Citing this paper

Please note that where the full-text provided on King's Research Portal is the Author Accepted Manuscript or Post-Print version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version for pagination, volume/issue, and date of publication details. And where the final published version is provided on the Research Portal, if citing you are again advised to check the publisher's website for any subsequent corrections.

General rights


Copyright and moral rights for the publications made accessible in the Research Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Research Portal

Take down policy

If you believe that this document breaches copyright please contact librarypure@kcl.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

The feasibility and inevitability of stealth attacks

IVAN Y. TYUKIN*

Department of Mathematics, King's College London, Strand, WC2R 2LS, London, UK

DESMOND J. HIGHAM

School of Mathematics, University of Edinburgh, Peter Guthrie Tait Road, EH9 3FD, Edinburgh, UK

AND

ALEXANDER BASTOUNIS, ELIYAS WOLDEGEORGIS, AND ALEXANDER N. GORBAN

School of Computing and Mathematical Sciences, University of Leicester, University Road, LE1 7RH,
Leicester, UK

*Corresponding author: ivan.tyukin@kcl.ac.uk

[Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year]

We develop and study new adversarial perturbations that enable an attacker to gain control over decisions in generic Artificial Intelligence (AI) systems including deep learning neural networks. In contrast to adversarial data modification, the attack mechanism we consider here involves alterations to the AI system itself. Such a *stealth attack* could be conducted by a mischievous, corrupt or disgruntled member of a software development team. It could also be made by those wishing to exploit a “democratization of AI” agenda, where network architectures and trained parameter sets are shared publicly. We develop a range of new implementable attack strategies with accompanying analysis, showing that with high probability a stealth attack can be made transparent, in the sense that system performance is unchanged on a fixed validation set which is unknown to the attacker, while evoking any desired output on a trigger input of interest. The attacker only needs to have estimates of the size of the validation set and the spread of the AI’s relevant latent space. In the case of deep learning neural networks, we show that a *one neuron attack* is possible—a modification to the weights and bias associated with a single neuron—revealing a vulnerability arising from over-parameterization. We illustrate these concepts using state of the art architectures on two standard image data sets. Guided by the theory and computational results, we also propose strategies to guard against stealth attacks.

Keywords: Adversarial attacks, AI instability, concentration of measure, backdoor attacks, AI robustness

1. Introduction

It is widely recognized that Artificial Intelligence (AI) systems can be vulnerable to adversarial attacks [37]; that is, small, often imperceptible, perturbations that exploit instabilities. The danger of an attacker gaining control of an automated decision-making process is of particular concern in high-stakes or safety-critical settings, including medical imaging [13, 14], transport [41] and textual analysis [12]. The last few years have therefore seen an escalation in the design of both attack and defence strategies [1, 23, 34], and recent work has considered the bigger question of whether the existence of instabilities and successful attacks is inevitable [3, 6, 11, 33, 40].

Work in this field has focused on adversarial perturbations to the training or input data, and may be classified as black-box, grey-box or white-box, depending on the level of information and access available to the attacker. We look at a different setting, which was introduced in [40] and is motivated further in subsection 2.1.

Here, the attacker is assumed to have control of the AI system, in the sense of being able to edit code or change parameters. We therefore consider attacks that make perturbations to the system itself in an attempt to change the output on a specific input point of interest to the attacker *without any changes* in the AI’s performance on a large, unknown, validation set and *without retraining*. We focus on two issues: (1) strategies for constructing such attacks and (2) theoretical analysis of the likelihood of success.

In Section 2 we present a general framework of open-box stealth attack and discuss additional related work. In Section 3 we reveal *a new class of attacks* (Algorithms 1 – 3) and establish probability bounds of their success (Theorems 1 – 3).

Algorithm 1, and the corresponding Theorem 1, consider the case when the Attacker’s choice of triggers is limited only by the requirement that the triggers’ feature representations stay within a ball containing the Owner’s validation set (choosing triggers with feature representations outside of this ball always results in successful attacks). In this case, we show that the likelihood of attack success can be made *arbitrarily* close to one, *for any fixed dimension* of the model’s latent space (Theorem 1, Remark 4). This is a significant departure from the previous state of knowledge, as success likelihoods for such attacks were thought to be limited by dimension [40]. To establish these high probabilities of success, the attack must be executed with arbitrarily high accuracy and the model must satisfy appropriate reachability conditions [23].

Algorithm 2 and Theorem 2 relate to approaches that enable attack triggers to be camouflaged as legitimate data by requesting that the Attacker’s triggers produce latent representations within some given neighborhood of those corresponding to specified inputs.

Algorithm 3 and Theorem 3 consider the case where the Attackers’ capability to change or explore feature spaces of the model is constrained to some finite number of attributes or a smaller-dimensional subspace. The case is motivated by the ideas from [7] where the authors proposed methods to generate adversarial perturbations confined to smaller-dimensional subspaces of the original input space (in contrast to the stealth attack setting considered here). This scenario enables attacks for models with sparse data representations in latent spaces. Remarkably, these constrained attacks may have significant probability of success even when the accuracy of their implementation is relatively low; see Theorem 3.

Section 4 presents experiments which illustrate the application of the theory to realistic settings and demonstrates the strikingly likely feasibility of *one neuron attacks*—which alter weights of just a single neuron. Section 5 concludes with recommendations on how vulnerabilities we exposed in this work can be mitigated by model design practices. Proofs of the theorems can be found in the Appendix, along with extra algorithmic details and computational results.

2. Stealth attacks

2.1. General framework

Consider a generic AI system, a map

$$\mathcal{F} : \mathcal{U} \rightarrow \mathbb{R} \tag{2.1}$$

producing some decisions on its outputs in response to an input from \mathcal{U} . The map \mathcal{F} can define input-output relationships for an entire deep neural network or some part (a sub-graph), an ensemble of networks, a tree, or a forest and the set \mathcal{U} could be a subset of \mathbb{R}^m for image classification or some set of possible sentences for text analysis. For the purposes of our work, the AI system’s specific task is not relevant and can include classification, regression, or density estimation.

In the classification case, if there are multiple output classes then we regard (2.1) as representing the output component of interest—we consider changes to \mathcal{F} that do not affect any other output components; this is the setting in which our computational experiments are conducted. The flexibility for stealth attacks to work independently of the choice of output component is a key feature of our work.

The AI system has an Owner operating the AI. An Attacker wants to take advantage of the AI by forcing it to make decisions in their favour. Conscious about security, the Owner created a *validation set* which is kept secret. The validation set is a list of input-output pairs produced by the uncompromised system (2.1). The Owner can monitor security by checking that the AI reproduces these outputs. Now, suppose that the Attacker has access to the AI system but not the validation set. The phrase *stealth attack* was used in [40] to describe the circumstance where the Attacker chooses a *trigger input* and *modifies the AI* so that:

- the Owner could not detect this modification by testing on the validation set,
- on the trigger input the modified AI produces the output desired by the Attacker.

Figure 1, panel C, gives a schematic representation of this setup. The setup is different from other known attack types such as adversarial attacks in which the Attacker exploits access to AI to compute imperceptible input perturbations altering AI outputs (shown in Figure 1, panel A), and data poisoning attacks (Figure 1, panel B) in which the Attacker exploits access to AI training sets to plant triggers directly.

We note that the stealth attack setting is relevant to the case of a corrupt, disgruntled or mischievous individual who is a member of a software development team that is creating an AI system, or who has an IT-related role where the system is deployed. In this scenario, the attacker will have access to the AI system and, for example, in the case of a deep neural network, may choose to alter the weights, biases or architecture. The scenario is also pertinent in contexts where AI systems are exchanged between parties, such as

- “democratization of AI” [2], where copies of large-scale models and parameter sets are made available across multiple public domain repositories,
- transfer learning [29], where an existing, previously trained tool is used as a starting point in a new application domain,
- outsourced cloud computing, where a third party service conducts training [15].

2.2. Formal definition of stealth attacks

Without loss of generality, it is convenient to represent the initial general map (2.1) as a composition of two maps, F and Φ :

$$\mathcal{F} = F \circ \Phi, \quad \text{where } F : \mathbb{R}^n \rightarrow \mathbb{R}, \quad \Phi : \mathcal{U} \rightarrow \mathbb{R}^n. \quad (2.2)$$

The map Φ defines general *latent* representation of inputs from \mathcal{U} , whereas the map F can be viewed as a *decision-making* part of the AI system. In the context of deep learning models, latent representations can be outputs of hidden layers in deep learning neural networks, and decision-making parts could constitute operations performed by fully-connected and softmax layers at the end of the networks. If Φ is an identity map then setting $F = \mathcal{F}$ brings us to the initial case (2.1).

An additional advantage of the compositional representation (2.2) is that it enables explicit modelling of the *focus of the adversarial attack*—a part of the AI system subjected to adversarial modification. This part will be modelled by the map F .

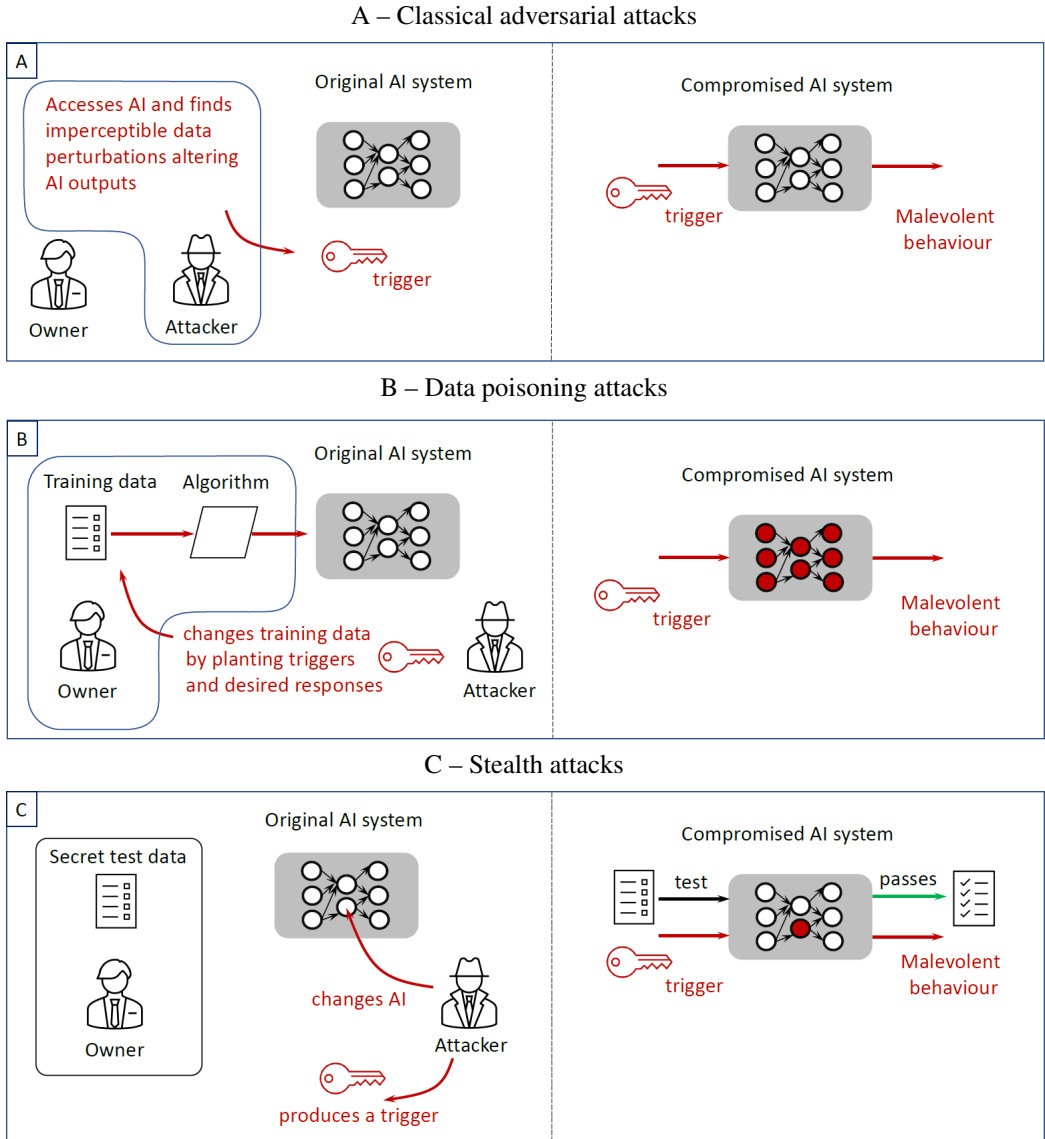


FIG. 1. General schemes of adversarial (panel A), data poisoning (panel B), and stealth attacks (panel C).

A perturbed, or attacked, map F_a is defined as

$$\begin{aligned}
 F_a &: \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}, \\
 F_a(\cdot, \theta) &= F(\cdot) + \mathfrak{A}(\cdot, \theta),
 \end{aligned} \tag{2.3}$$

where the term $\mathfrak{A} : \mathbb{R}^n \times \Theta \rightarrow \mathbb{R}$ models the *effect* of an adversarial perturbation, and $\Theta \subset \mathbb{R}^m$ is a set of relevant parameters. Such adversarial perturbations could take different forms including modification

of parameters of the map F and physical addition or removal of components involved in computational processes in the targeted AI system. In neural networks the parameters are the weights and biases of a neuron or a group of neurons, and the components are neurons themselves. As we shall see later, a particularly instrumental case occurs when the term \mathfrak{A} is just a single Rectified Linear Unit (ReLU function), [21], and $\theta = (w, b)$ represents the weights and bias,

$$\mathfrak{A}(\cdot, w, b) = D\text{ReLU}(\langle \cdot, w \rangle - b) \quad \text{where} \quad \text{ReLU}(s) = \max\{s, 0\}, \quad (2.4)$$

or a sigmoid (see [18], [22] for further information on activation functions of different types)

$$\mathfrak{A}(\cdot, w, b) = D\sigma(\langle \cdot, w \rangle - b), \quad \text{where} \quad \sigma(s) = \frac{1}{1 + \exp(-s)}, \quad (2.5)$$

where, $D \in \mathbb{R}$ is a constant gain factor.

Having introduced the relevant notation, we are now ready to provide a formal statement of the problem of stealth attacks introduced in Section 2.1.

Problem 1 (ε - Δ Stealth Attack on \mathcal{F}) Consider a map \mathcal{F} defined by (2.1), (2.2). Suppose that an owner of the AI system has a finite validation (or verification) set $\mathcal{V} \subset \mathcal{U}$. The validation set \mathcal{V} is kept secret and is assumed to be *unknown* to an attacker. The cardinality of \mathcal{V} is denoted by M , and this value is not necessarily known to the attacker.

Given $\varepsilon \geq 0$ and $\Delta > 0$, a successful ε - Δ stealth attack takes place if the attacker modifies the map F in \mathcal{F} and replaces it by F_a constructed such that for some $u' \in \mathcal{U}$, known to the attacker but unknown to the owner of the map \mathcal{F} , the following properties hold:

$$\begin{aligned} |F \circ \Phi(u) - F_a \circ \Phi(u)| &\leq \varepsilon \text{ for all } u \in \mathcal{V} \\ |F \circ \Phi(u') - F_a \circ \Phi(u')| &\geq \Delta. \end{aligned} \quad (2.6)$$

In words, when F is perturbed to F_a the output is changed by no more than ε on the validation set, but is changed by at least Δ on the trigger, u' . We note that this definition does not require any notion of whether the classification of u or u' is “correct.” In practice we are interested in cases where Δ is sufficiently large for u' to be assigned to different classes by the original and perturbed maps.

Remark 1 (The target class for the trigger u') Note that the above setting can be adjusted to fit a broad range of problems, including general multi-class problems. Crucially, the stealth attacks proposed in this paper allow the attacker to choose which class the modified AI system F_a predicts for the trigger image u' . We illustrate these capabilities with examples in Section 4.

2.3. Related work

Adversarial attacks. A broad range of methods aimed at revealing vulnerabilities of state-of-the-art AI, and deep learning models in particular, to adversarial inputs has been proposed to date (see e.g. recent reviews [23, 32]). The focus of this body of work has been primarily on perturbations to signals/data processed by the AI. In contrast to this established framework, here we explore possibilities to determine and implement small *changes to AI structure* and *without retraining*.

Data poisoning. Gu et al [20] (see also [8] and references therein, and [27] for explicit upper bounds on the volume of poisoned data that is sufficient to execute these attacks) showed how malicious data

poisoning occurring, for example, via outsourcing of training to a third party, can lead to *backdoor* vulnerabilities. Performance of the modified model on the validation set, however, is not required to be perfect. A data poisoning attack is deemed successful if the model’s performance on the validation set is within some margin of the user’s expectations.

The data poisoning scenario is different from our setting in two fundamental ways. First, in our case the attacker can maintain performance of the perturbed system on an unknown validation set within *arbitrary small* or, in case of ReLU neurons, *zero* margins. Such *imperceptible* changes on unknown validation sets is a signature characteristic of the threat we have revealed and studied. Second, the attacks we analysed *do not require any retraining*.

Other stealth attacks. Liu et al [26] proposed a mechanism, SIN^2 , whereby a service provider gains control over their customers’ deep neural networks through the provider’s specially designed APIs. In this approach, the provider needs to first plant malicious information in higher-precision bits (e.g. bits 16 and higher) of the network’s weights. When an input trigger arrives the provider extracts this information via its service’s API and performs malicious actions as per instructions encoded.

In contrast to this approach, stealth attacks we discovered *do not require* any special computational environments. After an attack is planted, it can be executed in fully secure and trusted infrastructure. In addition, our work reveals further concerns about how easily a malicious service provider can implement stealth attacks in hostile environments [26] by swapping bits in the mantissa of weights and biases of a single neuron (see Figure 2 for the patterns of change) at will.

Our current work is a significant advancement from the preliminary results presented in [40], both in terms of algorithmic detail and theoretical understanding of the phenomenon. First, the vulnerabilities we reveal here are much more severe. Bounds on the probability of success for attacks in [40] are constrained by $1 - M2^{-n}$. Our results show that under the same assumptions (that input reachability [23] holds true), the probabilities of success for attacks generated by Algorithms 1, 2 can be made arbitrarily close to one (Remark 4). Second, we explicitly account for cases when input reachability holds only up to some accuracy, through parameters α, δ (Remark 3). Third, we present concrete algorithms, scenarios and examples of successful exploitation of these new vulnerabilities, including the case of one neuron attacks (Section 4, Appendix; code is available in [39]).

3. New stealth attack algorithms

In this section we introduce two new algorithms for generating stealth attacks on a generic AI system. These algorithms return a ReLU or a sigmoid neuron realizing an adversarial perturbation \mathfrak{A} . Implementation of these algorithms will rely upon some mild additional information about the unknown validation set \mathcal{V} . In particular, we request that latent representations $\Phi(u)$ for all $u \in \mathcal{V}$ are located within some ball $\mathbb{B}_n(0, R)$ whose radius R is *known* to the attacker. We state this requirement in Assumption 1 below.

Assumption 1 (Latent representations of the validation data \mathcal{V}) *There is an $R > 0$, known to the attacker, such that*

$$\Phi(u) \in \mathbb{B}_n(0, R) \text{ for all } u \in \mathcal{V}. \tag{3.1}$$

Given that the set \mathcal{V} is finite, infinitely many such balls exist. Here we request that the attacker knows just a single value of R for which (3.1) holds. The value of R does not have to be the smallest possible. A bound for R is readily available when, for example, each component of Φ is bounded, as in the case where Φ represents the first few layers of a neural network concluding with a sigmoid function.

Similarly, a bound may be computed when Φ represents the first few layers of a neural network with known weights and activation functions and the input consists of pixels from a specified finite range.

We also suppose that the feature map Φ in (2.2) satisfies an appropriate reachability condition (cf. [23]), based on the following definition.

Definition 1 (*v*-input reachability of the classifier’s latent space) *Consider a function $f : \mathcal{U} \rightarrow \mathbb{R}^n$. A set $\mathcal{S} \subset \mathbb{R}^n$ is v-input reachable for the function f if for any $x \in \mathcal{S}$ there is an $u(x) \in \mathcal{U}$ such that $\|f(u(x)) - x\| \leq v$.*

In what follows we will assume existence of some sets, namely $n - 1$ spheres, in the classifier’s latent spaces which are input reachable for the map Φ . Precise definitions of these sets will be provided in our formal statements.

Remark 2 The requirement that the classifier’s latent space contains specific sets which are *v*-input reachable for the map Φ may appear restrictive. If, for example, feature maps are vector compositions of ReLU and affine mappings (2.4), then several components of these vectors might be equal to zero in some domains and thus input reachability might seem impossible. However, one can always search for a new feature map

$$\tilde{\Phi} : \tilde{\Phi}(u) = T\Phi(u), \quad T \in \mathbb{R}^{d \times n}, \quad (3.2)$$

where the matrix $T \in \mathbb{R}^{d \times n}$, $d \leq n$ is chosen so that relevant domains in the latent space formed by $\tilde{\Phi}$ become *v*-input reachable. As we shall see later in Theorems 1 and 2, these relevant domains are spheres $\mathbb{S}_{d-1}(c, \delta R)$, $\delta \in (0, 1]$, where $c \in \mathbb{R}^d$ is some given reference point.

If the feature map $\Phi : \mathcal{U} \rightarrow \mathbb{R}^n$ with $\mathcal{U} \subset \mathbb{R}^m$ is differentiable and non-constant, and the set \mathcal{U} has a non-empty interior, $\text{Int}(\mathcal{U})$, then matrices T producing *v*-input reachable feature maps $\tilde{\Phi}$ in some neighborhood of a point from $\text{Int}(\mathcal{U})$ can be determined as follows. Pick a target input $u_0 \in \text{Int}(\mathcal{U})$, then

$$\Phi(u) = \Phi(u_0) + J(u - u_0) + o(\|u - u_0\|),$$

where J is the Jacobian of Φ at u_0 and $\|\cdot\|$ denotes the Euclidean norm. Suppose that $\text{rank}(J) = d$, $d > 0$, and let

$$J = U \begin{pmatrix} \Sigma & 0_{d \times (m-d)} \\ 0_{(n-d) \times d} & 0_{(n-d) \times (m-d)} \end{pmatrix} V$$

be the singular value decomposition of J , where Σ is a $d \times d$ diagonal matrix containing non-zero singular values of J on its main diagonal, and $U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{m \times m}$ are unitary matrices. If $m = d$ or $n = d$ then the corresponding zero matrices in the above decomposition are assumed to be empty. Setting

$$T = (I_d \ 0_{d \times (n-d)}) U^T$$

ensures that for any arbitrarily small $v > 0$ there is an $r(v, u_0) > 0$ such that the sets $\mathbb{S}_{d-1}(\tilde{\Phi}(u_0), \rho)$, $\rho \in [0, r(v, u_0))$ are *v*-input reachable for the function $\tilde{\Phi} = T\Phi$.

Note that the same argument applies to maps Φ which are differentiable only in some neighborhood of $u_0 \in \text{Int}(\mathcal{U})$. This enables the application of this approach for producing *v*-input reachable feature functions to maps involving compositions of ReLU functions. In what follows we will use the symbol n to denote the dimension of the space where Φ maps the input into assuming that the input reachability condition holds for this feature map.

Linearity of $\tilde{\Phi}$ in T enables one to preserve the structure of perturbations (2.4), (2.5) so that they remain, in effect, just a single additional neuron. We used the latter property in our computational experiments (see Remark 11 in Section A.4) to generate examples of stealth attacks.

3.1. Target-agnostic stealth attacks

Our first stealth attack algorithm is presented in Algorithm 1. The algorithm produces a modification of the relevant part F of the original AI system that is implementable by a single ReLU or sigmoid function. Regarding the *trigger input*, u' , the algorithm relies on another process (see step 3).

In what follows, we will denote this process as an auxiliary algorithm \mathcal{A}_0 which, for the map Φ , given R , parameters $\delta \in (0, 1], \gamma \in (0, 1)$, $v < \delta$, $\delta + v \leq \gamma^{-1}$, whose meaning and choice are explained in Remark 3 after Theorem 1, and any $x \in \mathbb{S}_{n-1}(0, \delta)$, returns a solution of the following constrained search problem:

$$\mathcal{A}_0(\Phi, x, R, v): \quad \text{find } u \in \mathcal{U} \quad \text{such that} \quad \|\Phi(u)R^{-1} - x\| \leq v. \quad (3.3)$$

Observe that vR -input reachability, $v < \delta$, of the set $\mathbb{S}_{n-1}(0, R\delta)$ for the map Φ together with the choice of δ, v , and γ satisfying $\delta + v \leq \gamma^{-1}$ ensure existence of a solution to the above problem for every $x \in \mathbb{S}_{n-1}(0, \delta)$.

Thorough analysis of computability of solutions of (3.3) is outside of the scope of the work (see [5] for an idea of the issues involved computationally in solving optimisation problems). Therefore, we shall assume that the auxiliary process \mathcal{A}_0 always returns a solution of (3.3) for a choice of δ, γ , and given v, R . In our numerical experiments (see Section 4), finding a solution of (3.3) did not pose significant issues.

With the auxiliary algorithm \mathcal{A}_0 available, we can now present Algorithm 1:

The performance of Algorithm 1 in terms of the probability of producing a successful stealth attack of the form (2.3) is characterised by Theorem 1 below.

Theorem 1 *Let Assumption 1 hold. Consider Algorithm 1, and let $\gamma \in (0, 1)$, $\alpha \in [0, 1)$, and $\delta \in (0, 1]$ be such that $(1 + \alpha)\delta \leq 1$ and $\gamma(1 - \alpha)\delta > \alpha$. Additionally, assume that the set $\mathbb{S}_{n-1}(0, R\delta)$ is $\delta\alpha R$ -input reachable for the map Φ . Then*

$$\varphi(\gamma, \delta, \alpha) := \cos(\arccos(\gamma(1 - \alpha)\delta) + \arccos((1 - \alpha^2)^{1/2})) > 0 \quad (3.7)$$

and the probability $P_{a,1}$ that Algorithm 1 returns a successful ε - Δ stealth attack satisfies

$$P_{a,1} \geq 1 - M\pi^{-1/2} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})} \int_0^{\arccos(\varphi(\gamma, \delta, \alpha))} \sin^{n-2}(\theta) d\theta. \quad (3.8)$$

In particular,

$$P_{a,1} \geq 1 - M \frac{1}{2\pi^{1/2}} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n}{2} + \frac{1}{2})} \frac{1}{\varphi(\gamma, \delta, \alpha)} (1 - \varphi(\gamma, \delta, \alpha)^2)^{\frac{n-1}{2}}. \quad (3.9)$$

Remark 3 (Choice of parameters) The three parameters α, δ and γ should be chosen to strike a balance between attack success, time to compute the algorithm, and the size of the weights that the stealth attack creates. In particular, from the perspective of attack success (3.8) is optimized when γ and

Algorithm 1 Single-neuron plain stealth attack

Input: $\delta \in (0, 1]$, $\gamma \in (0, 1)$, $\alpha \in [0, 1)$, $(1 + \alpha)\delta \leq 1$, $\Delta, \varepsilon \geq 0$, a sigmoid or a ReLU function g , and R satisfying (3.1).

1: **procedure** ADVERSARIAL STEALTH PERTURBATION($\delta, \Delta, \varepsilon, g$)

2: Draw a random vector x from the equidistribution in the sphere $\mathbb{S}_{n-1}(0, \delta)$, $\delta \in (0, 1]$.

3: Use algorithm \mathcal{A}_0 (see (3.3)) to generate an input $u' \in \mathcal{U}$ such that $x' = \Phi(u')/R$ is within a $\alpha\delta$ -distance from x :

$$\|x' - x\| \leq \alpha\delta. \quad (3.4)$$

4: Set

$$\mathfrak{A}(\cdot, \kappa x' R^{-1}, b) = Dg(\langle \cdot, \kappa x' R^{-1} \rangle - b), \quad \text{with } b = 0.5\kappa(1 + \gamma)\|x'\|^2, \quad (3.5)$$

where κ and D are chosen so that

$$Dg(-0.5\kappa(1 - \gamma)\|x'\|^2) \leq \varepsilon \text{ and } Dg(0.5\kappa(1 - \gamma)\|x'\|^2) \geq \Delta. \quad (3.6)$$

*Note that a choice of κ, D so that (3.6) is satisfied is always possible.

5: **end procedure**

Output: trigger u' , weight vector $w = \kappa x' R^{-1}$, bias $b = 0.5\kappa(1 + \gamma)\|x'\|^2$, and output gain D of the sigmoid or ReLU function g .

δ are close to 1 and α is close to 0. However, α and δ are restricted by the exact form of the map Φ : they need to be chosen so that the set $\mathbb{S}_{n-1}(0, R\delta)$ is $\delta\alpha R$ -input reachable for Φ . Furthermore, the speed of executing \mathcal{A}_0 is also influenced by these choices with the complexity of solving (3.4) increasing as α decreases to 0. The size of the weights chosen in (3.6) is influenced by the choice of γ so that the L_2 norm of the attack neuron weights grows like $O((1 - \gamma)^{-1})$ for sigmoid g . Finally, the condition $(1 + \alpha)\delta \leq 1$ ensures that the chosen trigger image u' has $\|\Phi(u')\| \leq R$.

Remark 4 (Determinants of success and vulnerabilities) Theorem 1 establishes explicit connections between intended parameters of the trigger u' (expressed by $\Phi(u')/R$ which is to be maintained within $\delta\alpha$ from x), dimension n of the AI's *latent* space, accuracy of solving (3.3) (expressed by the value of α —the smaller the better), design parameters $\gamma \in (0, 1)$ and $\delta \in (0, 1]$, and *vulnerability* of general AI systems to stealth attacks.

In general, the larger the value of n for which solutions of (3.3) can be found without increasing the value of α , the higher the probability that the attack produced by Algorithm 1 is successful. Similarly, for a given success probability bound, the larger the value of n , the smaller the value of γ required, allowing stealth attacks to be implemented with smaller weights w and bias b . At the same time, if no explicit restrictions on δ and the weights are imposed then Theorem 1 suggests that if one picks $\delta = 1$, γ sufficiently close to 1, and α sufficiently close to 0, subject to finding an appropriate solution of (3.3) (c.f. the notion of reachability [23]), one can create stealth attacks whose probabilities of success *can be made arbitrarily close to 1 for any fixed n* . Indeed, if $\alpha = 0$ then the right-hand side of (3.9) becomes

$$1 - M \frac{1}{2\pi^{\frac{1}{2}}} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n}{2} + \frac{1}{2})} \frac{1}{\gamma\delta} (1 - (\gamma\delta)^2)^{\frac{n-1}{2}} \quad (3.10)$$

which, for any fixed M, n , can be made arbitrarily close to 1 by an appropriate choice of $\gamma \in (0, 1)$ and $\delta \in (0, 1]$.

Remark 5 (Arbitrary output sign and target class of adversarial perturbation) Although Problem 1 does not impose any requirements on the sign of $\mathfrak{A}(\Phi(u'), w, b)$, this quantity can be made positive or negative through the choice of D . Note that the target class can be arbitrary too (see Remark 1).

Remark 6 (Precise and zero-tolerance attacks with ReLU units) One of the original requirements for a stealth attack is to produce a response to a trigger input u' such that $|F \circ \Phi(u') - F_a \circ \Phi(u')|$ exceeds an a-priori given value Δ . However, if the attack is implemented with a ReLU unit then one can select the values of w, b so that $|F \circ \Phi(u') - F_a \circ \Phi(u')|$ exactly equals Δ . Indeed for a ReLU neuron, condition (3.6) reduces to

$$0.5D(\kappa(1 - \gamma)\|x'\|^2) \geq \Delta.$$

Hence picking $\kappa = 2\Delta((1 - \gamma)\|x'\|^2)^{-1}$ and $D = 1$ results in the desired output.

Moreover, ReLU units produce adversarial perturbations \mathfrak{A} with $\varepsilon = 0$. These *zero-tolerance* attacks, if successful, do not change the values of \mathcal{F} on the validation set \mathcal{V} and as such are completely undetectable on \mathcal{V} .

Remark 7 (Hiding adversarial perturbations in redundant deep learning structures) Algorithm 1 (as well as its input-specific version, Algorithm 2, below) implements adversarial perturbations by *adding* a single sigmoid or ReLU neuron. A question therefore arises, if one can “plant” or “hide” a stealth attack within an existing AI structure, rather than adding a neuron, as in (2.3). Intuitively, over-parametrisation and redundancies in many deep learning architectures should provide ample opportunities precisely for this sort of malevolent action. As we empirically justify in Section 4, this is indeed possible. In these experiments, we looked at a neural network with L layers. The map F corresponded to the last $k + 1$ layers: $L - k, \dots, L, k \geq 1$. We looked for a neuron in layer $L - k$ whose output weights have the smallest L_1 -norm of all neurons in that layer (see Appendix, Section A.6). This neuron was then replaced with a neuron implementing our stealth attack, and its output weights were wired so that a given trigger input u' evoked the response we wanted from this trigger. This new type of one neuron attack may be viewed as the stealth version of a one pixel attack [36]. Surprisingly, this approach worked consistently well across various randomly initiated instances of the same network. These experiments suggest a real non-hypothetical possibility of turning *a needle in a haystack* (a redundant neuron) into *a pebble in a shoe* (a malevolent perturbation).

Remark 8 (Stability) Under appropriate (weak) assumptions, such as the existence of a topology on \mathcal{U} where Φ is continuous and the continuity of g , Algorithm 1 generates stable trigger inputs—if $u'' \in \mathcal{U}$ is sufficiently close to u' then u'' is a trigger image in the sense of (1) with a potentially smaller value of Δ .

Remark 9 (Other activation functions) In this work, to be concrete we focus on the case of attacking with a ReLU or sigmoid activation function. However, the algorithms and analysis readily extend to the case of any continuous activation function that is nonnegative and monotonically increasing. Appropriate pairs of matching leaky ReLUs also fit into this framework.

3.2. Target-specific stealth attacks

The attack and the trigger $u' \in \mathcal{U}$ constructed in Algorithm 1 are “arbitrary” in the sense that x is drawn at random. As opposed to standard adversarial examples, they are not linked or targeting any specific input. Hence a question arises: is it possible to create a *targeted* adversarial perturbation of the AI which is triggered by an input $u' \in \mathcal{U}$ located in a vicinity of some specified input, u^* ?

As we shall see shortly, this may indeed be possible through some modifications to Algorithm 1. For technical convenience and consistency, we introduce a slight reformulation of Assumption 1.

Assumption 2 (Relative latent representations of the validation data \mathcal{V}) *Let $u^* \in \mathcal{U}$ be a target input of interest to the attacker. There is an $R > 0$, also known to the attacker, such that*

$$\Phi(u) - \Phi(u^*) \in \mathbb{B}_n(0, R) \text{ for all } u \in \mathcal{V}. \quad (3.11)$$

We note that Assumption 2 follows immediately from Assumption 1 if a bound on the size of the latent representation $\Phi(u^*)$ of the input u^* is known. Indeed, if R' is a value of R for which (3.1) holds then (3.11) holds whenever $R \geq R' + \|\Phi(u^*)\|$.

Algorithm 2 provides a recipe for creating such targeted attacks.

Algorithm 2 Single-neuron targeted stealth attack

Input: $\delta \in (0, 1]$, $\gamma \in (0, 1)$, $\alpha \in [0, 1)$, $\delta(\alpha + 1) \leq 1$, $\Delta, \varepsilon \geq 0$, a sigmoid or a ReLU function g , a target input $u^* \in \mathcal{U}$, and R for which (3.11) holds.

- 1: **procedure** TARGETED ADVERSARIAL STEALTH PERTURBATION($\delta, \alpha, D, \varepsilon, g$)
- 2: Draw a random vector x from the equidistribution in the sphere $\mathbb{S}_{n-1}(0, \delta)$, $\delta \in (0, 1]$.
- 3: Use the algorithm \mathcal{A}_0 to generate an input u' such that $x' = (\Phi(u') - \Phi(u^*)) / R$ satisfies (3.4).
- 4: Set

$$\begin{aligned} \mathfrak{A}(\cdot, \kappa x' R^{-1}, b) &= Dg(\langle \cdot, \kappa x' R^{-1} \rangle - b), \\ b &= 0.5\kappa(1 + \gamma)\|x'\|^2 + \kappa\langle \Phi(u^*), x' R^{-1} \rangle, \end{aligned}$$

where κ and D are chosen as in (3.6).

*Note that such a choice is always possible.

- 5: **end procedure**

Output: trigger input u' , weight vector $w = \kappa x' R^{-1}$, bias $b = 0.5\kappa(1 + \gamma)\|x'\|^2 + \kappa\langle \Phi(u^*), x' R^{-1} \rangle$, and output gain D of the sigmoid or a ReLU function g .

Performance bounds for Algorithm 2 can be derived in the same way as we have done in Theorem 1 for Algorithm 1. Formally, we state these bounds in Theorem 2 below.

Theorem 2 *Let Assumption 2 hold. Consider Algorithm 2, and let parameters $\gamma \in (0, 1)$, $\alpha \in [0, 1)$, and $\delta \in (0, 1]$ be such that $\delta(1 + \alpha) \leq 1$ and $\gamma(1 - \alpha)\delta > \alpha$. Additionally, assume that the set $\mathbb{S}_{n-1}(\Phi(u^*), R\delta)$ is $\delta\alpha R$ -input reachable for the map Φ .*

Then the probability $P_{a,2}$ that Algorithm 2 returns a successful ε - Δ stealth attack is bounded from below by (3.8) and (3.9); that is,

$$P_{a,2} \geq 1 - M\pi^{-1/2} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})} \int_0^{\arccos(\varphi(\gamma, \delta, \alpha))} \sin^{n-2}(\theta) d\theta,$$

and in particular,

$$P_{a,1} \geq 1 - M \frac{1}{2\pi^{1/2}} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n}{2} + \frac{1}{2})} \frac{1}{\varphi(\gamma, \delta, \alpha)} (1 - \varphi(\gamma, \delta, \alpha)^2)^{\frac{n-1}{2}}.$$

Remarks 3–9 apply equally to Algorithm 2. In addition, the presence of a specified target input u^* offers extra flexibility and opportunities. An attacker may for example have a list of potential target inputs. Applying Algorithm 2 to each of these inputs produces different triggers u' each with different possible values of α . According to Remark 4 (see also (3.10)), small values of α imply higher probabilities that the corresponding attacks are successful. Therefore, having a list of target inputs and selecting a trigger with minimal α increases the attacker's chances of success.

3.3. Attribute-constrained attacks and a “concentrational collapse” for validation sets \mathcal{V} chosen at random

The attacks developed and analysed so far can affect all attributes of the input data's latent representations. It is also worthwhile to consider triggers whose deviation from targets in latent spaces is limited to only a few attributes. This may help to disguise triggers as legitimate data, an approach which was recently shown to be successful in the context of adversarial attacks on input data [7]. Another motivation stems from more practical scenarios where the task is to find a trigger in the vicinity of the latent representation of another input in models with sparse coding. The challenge here is to produce attack triggers whilst retaining sparsity. Algorithm 3 and the corresponding Theorem 3 are motivated by these latter scenarios.

As before, consider an AI system with an input-output map (2.2). Assume that the attacker has access to a suitable real number $R > 0$, an input u^* , and a set of orthonormal vectors $h_1, h_2, \dots, h_{n_p} \in \mathbb{R}^n$ which form the space in which the attacker can make perturbations. Furthermore, assume that both the validation set and the input u^* satisfy Assumption 3 below.

Assumption 3 (Data model) *Elements u_1, \dots, u_M are randomly chosen so that $\Phi(u_1), \dots, \Phi(u_M)$ are random variables in \mathbb{R}^n and such that both of the following hold:*

1. *There is a $c \in \mathbb{R}^n$ such that u^* , and (with probability 1), u_1, \dots, u_M , are in the set:*

$$\{u \in \mathcal{U} \mid \|\Phi(u) - c\| \leq R/2\}. \quad (3.12)$$

2. *There is a $C \geq 1$ such that for any $r \in (0, R/2]$ and $\xi \in \mathbb{B}_n(c, R/2)$*

$$P(\Phi(u_i) \in \mathbb{B}_n(\xi, r)) \leq C \left(\frac{2r}{R}\right)^n. \quad (3.13)$$

Property (3.12) requires that the validation set is mapped into a ball centered at c and having a radius $R/2$ in the system's latent space, and (3.13) is a non-degeneracy condition restricting pathological concentrations.

Consider Algorithm 3. As we show below, if the validation set satisfies Assumption 3 and the attacker uses Algorithm 3 then the probabilities of generating a successful attack may be remarkably high even if the attack triggers' latent representations deviate from those of the target in only few attributes.

Algorithm 3 Single-neuron targeted attribute-constrained stealth attack

Input: $\delta \in (0, 1]$, $\gamma \in (0, 1)$, $\alpha \in [0, 1)$, $\delta(\alpha + 1) \leq 1$, $\Delta, \varepsilon \geq 0$, a sigmoid or a ReLU function g , a target input $u^* \in \mathcal{U}$, an R for which Assumption 3 holds, an $n_p \in \mathbb{N}$, $2 \leq n_p \leq n$, and a system of orthonormal vectors $\{h_1, \dots, h_{n_p}\}$, $h_i \in \mathbb{R}^n$.

- 1: **procedure** TARGETED ADVERSARIAL CONSTRAINED STEALTH PERTURBATION($\delta, \alpha, D, \varepsilon, g$)
- 2: Draw a random vector v from the equidistribution in the sphere $\mathbb{S}_{n_p-1}(0, \delta)$, $\delta \in (0, 1]$. Set

$$x = \sum_{i=1}^{n_p} v_i h_i.$$

- 3: Use the algorithm \mathcal{A}_0 to generate an input u' such that $x' = (\Phi(u') - \Phi(u^*)) / R$ satisfies (3.4).
- 4: Set

$$\begin{aligned} \mathfrak{A}(\cdot, \kappa x' R^{-1}, b) &= Dg(\langle \cdot, \kappa x' R^{-1} \rangle - b), \\ b &= 0.5\kappa(1 + \gamma) \|x'\|^2 + \kappa \langle \Phi(u^*), x' R^{-1} \rangle, \end{aligned}$$

where κ and D are chosen as in (3.6).

*Note that such a choice is always possible.

- 5: **end procedure**

Output: trigger input u' , weight vector $w = \kappa x' R^{-1}$, bias $b = 0.5\kappa(1 + \gamma) \|x'\|^2 + \kappa \langle \Phi(u^*), x' R^{-1} \rangle$, and output gain D of the sigmoid or a ReLU function g .

Theorem 3 *Let Assumption 3 hold. Consider Algorithm 3, and let parameters $\gamma \in (0, 1)$, $\alpha \in [0, 1)$, and $\delta \in (0, 1]$ be such that the set $\Phi(u^*) + (\mathbb{S}_{n-1}(0, R\delta) \cap \text{span}\{h_1, \dots, h_{n_p}\})$ is $\delta\alpha R$ -input reachable for the map Φ . Furthermore, suppose that $2\delta\gamma(1 - \alpha) \geq \sqrt{1 - (MC)^{-2/n}}$.*

Let

$$\begin{aligned} \theta^* &= \arccos\left(\min\left\{1, 2\delta\gamma(1 - \alpha) - \sqrt{1 - (MC)^{-2/n}}\right\}\right) + \arccos(\sqrt{1 - \alpha^2}), \\ \rho(\theta) &= \min\left\{1, 2\delta\gamma(1 - \alpha) - \max\{0, \cos(\theta - \arccos(\sqrt{1 - \alpha^2}))\}\right\} \quad \text{for } \theta \in [\theta^*, \pi]. \end{aligned}$$

Then $\theta^* \in [0, \pi]$, $\rho(\theta) \in [0, 1]$ for $\theta \in [\theta^*, \pi]$ and the probability $P_{a,3}$ that Algorithm 3 returns a successful ε - Δ stealth attack is bounded from below by

$$1 - MC \frac{1}{\pi^{\frac{1}{2}}} \frac{\Gamma\left(\frac{n_p}{2}\right)}{\Gamma\left(\frac{n_p-1}{2}\right)} \int_{\theta^*}^{\pi} (1 - \rho(\theta))^{\frac{n_p}{2}} \sin^{n_p-2}(\theta) d\theta - \frac{1}{\pi^{\frac{1}{2}}} \frac{\Gamma\left(\frac{n_p}{2}\right)}{\Gamma\left(\frac{n_p-1}{2}\right)} \int_0^{\theta^*} \sin^{n_p-2}(\theta) d\theta. \quad (3.14)$$

α	0.3	0.25	0.20	0.15	0.10	0.05	0.01
Bound (3.8)	Not feasible	Not feasible	-703.2	-161.9	-17.0	0.105	0.956
Bound (3.14)	4.11×10^{-4}	0.014	0.148	0.533	0.886	0.990	0.999

TABLE 1 Comparison of bounds (3.8) and (3.14) for different values of the accuracy parameter α , and fixed $\gamma = 0.9$, $\delta = 1/3$, $n = n_p = 200$, $M = 2500$, $C = 100$.

α	0.3	0.25	0.20	0.15	0.10	0.05	0.01
Bound (3.8)	Not feasible	Not feasible	-1172.2	-1049.0	-929.0	-813.4	-724.9
Bound (3.14)	0.326	0.384	0.444	0.504	0.565	0.624	0.669

TABLE 2 Comparison of bounds (3.8), where n is replaced with n_p , and (3.14) for different values of the accuracy parameter α , and fixed $\gamma = 0.9$, $\delta = 1/3$, $n = 200$, $n_p = 5$, $M = 2500$, $C = 100$.

We emphasize a key distinction between the expressions (3.14) and (3.8). In (3.14) the final term is independent of M , whereas the corresponding term in (3.8) is multiplied by a factor of M . The only term affected by M in (3.14) is modulated by an additional exponent with the base $(1 - \rho(\theta)^2)^{1/2}$ which is strictly smaller than one in $(\theta^*, \pi]$ for $\theta^* < \pi$. In order to give a feel of how far bound provided in Theorem 3 may be from the one established in Theorems 1, 2 we computed the corresponding bounds for $n = n_p = 200$, $\gamma = 0.9$, $\delta = 1/3$, $M = 2500$, and $C = 100$, and $\alpha \in [0.01, 0.3]$. Results of the comparison are summarised in Table 1. In this context, Theorem 3 reveals a phenomenon where validation sets which could be deemed as sufficiently large in the sense of bounds specified by Theorems 1 and 2 may still be considered as “small” due to bound (3.14). We call this phenomenon *concentrational collapse*. When concentrational collapse occurs, the AI system becomes more vulnerable to stealth attacks when the cardinality of validation sets \mathcal{V} is sub-exponential in dimension n implying that the owner would have to generate and keep a rather large validation set \mathcal{V} to make up for these small probabilities. Remarkably, this vulnerability persists even when the attacker’s precision is small (α large).

Remark 10 (Lower-dimensional triggers) Another important consequence of concentrational collapse, which becomes evident from the proof of Theorem 3, is the possibility to sample vectors x from the $n_p - 1$ sphere, $\mathbb{S}_{n_p-1}(0, \delta)$, $2 \leq n_p < n$ instead of from the $n - 1$ sphere $\mathbb{S}_{n-1}(0, \delta)$. Note that the second term in the right-hand side of (3.14) scales linearly with cardinality M of the validation set \mathcal{V} and has an exponent in n which is the “ambient” dimension of the feature space. The third term in (3.14) decays exponentially with $n_p < n$ but does not depend on M . The striking difference between behavior of bounds (3.14) and (3.8) at small values of n_p and large n is illustrated with Table 2. According to Table 2, bound (3.8) is either infeasible or impractical for all tested values of the accuracy parameter α . In contrast, bound (3.14) indicates relatively high probabilities of stealth attacks’ success for the same values of α as long as all relevant assumptions hold true. This implies that the concentration collapse phenomenon can be exploited for generating triggers with lower-dimensional perturbations of target inputs in the corresponding feature spaces. The latter possibility may be relevant for overcoming defense strategies which enforce high-dimensional yet sparse representations of data in latent spaces. Our experiments in Section 4.1, which reveal high stealth attack success rates even for relatively low-dimensional perturbations, are consistent with these theoretical observations.

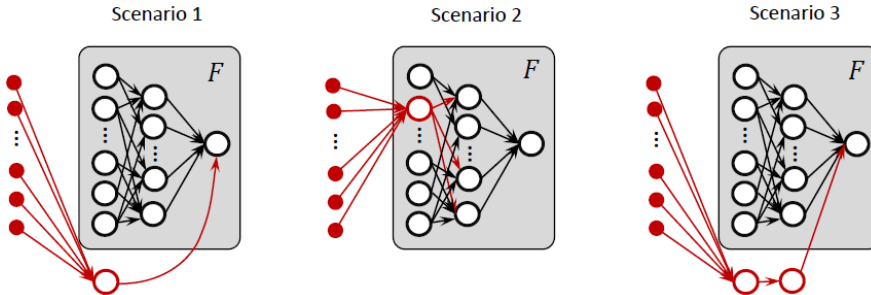


FIG. 2. Stealth attack implementation patterns. Red open circles and arrows indicate changes. Solid red circles show the input components, and how they feed into the new or perturbed neurons. In Scenarios 1 and 3 neuron(s) are added. In Scenario 2, a one neuron attack, the weights and biases of an existing neuron are replaced with new values. Grey boxes show a part of the network modelled by the map F .

4. Experiments

Let us show how stealth attacks can be constructed for given deep learning models. We considered two standard benchmark problems in which deep learning networks are trained on the CIFAR10 [24] and a MATLAB version of the MNIST [25] datasets. All networks had a standard architecture with feature-generation layers followed by dense fully connected layers.

Three alternative scenarios for planting a stealth attack neuron were considered. Schematically, these scenarios are shown in Figure 2. Our theoretical results directly apply to Scenario 1. The results are also relevant to Scenario 2, as discussed in Remark 7. Scenario 3 is included for completeness; it is computationally equivalent to Scenario 1 but respects the original structure more closely by passing information between successive layers, rather than skipping directly to the end.

In our experiments we determined trigger-target pairs and changed the network’s architecture in accordance with planting Scenarios 1 and 2. It is clear that if Scenario 1 is successful then Scenario 3 will be successful too. The main difference between Scenario 2 and Scenarios 1,3 is that in the former case we *replace* a neuron in F by the “attack” neuron. The procedure describing selection of a neuron to replace (and hence attack) in Scenario 2 is detailed in Section A.6, and an algorithm which we used to find triggers is described in Section A.4. When implementing stealth attacks in accordance with Scenario 2, we always selected neurons whose susceptibility rank is 1.

As a general rule, in Scenario 2, we attacked neurons in the block of fully connected layers just before the final softmax block (fully connected layer followed by a layer with softmax activation functions - see Figure 3 and Table 3 for details). The attacks were designed to assign an arbitrary class label the Attacker wanted the network to return in response to a trigger input computed by the Attacker. In order to do so, the weight between the attacked neuron and the neuron associated with the Attacker-intended class output was set to 1. All other output weights of the attacked neuron were set to 0. In principle, 0s can be replaced with negative, sufficiently small positive values, or their combinations. MATLAB code implementing all relevant steps of the experiments can be found in [39].

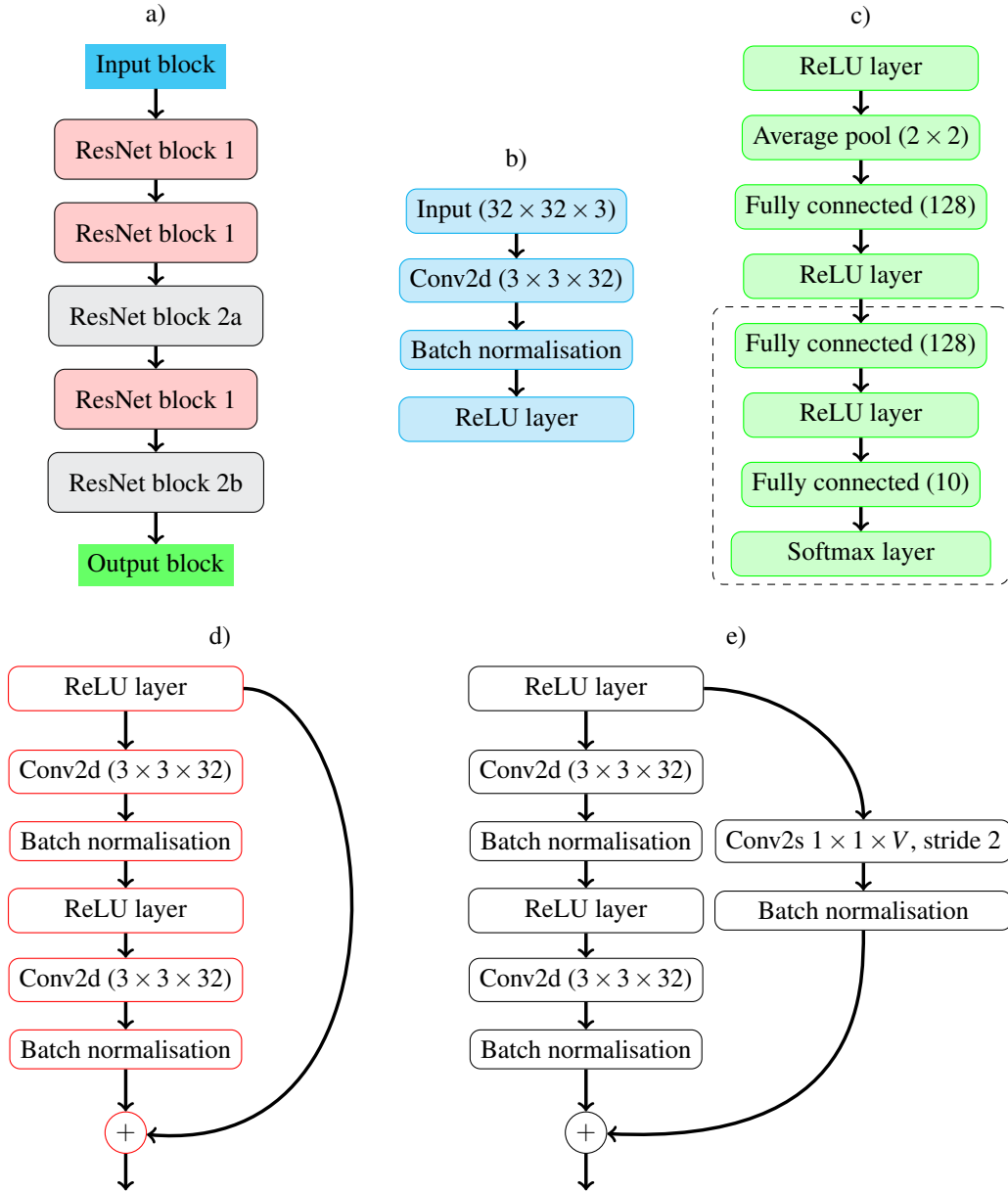


FIG. 3. ResNet architecture used in experiments with CIFAR-10 dataset. Panel *a* shows the general structure of the network. Panels *b* and *c* show configuration of the network's input and output blocks (highlighted in cyan and green in panel *a*, respectively). The diagram in panel *d* shows the structure of ResNet blocks 1. Panel *e* describes ResNet blocks 2a and 2b. The value of V in ResNet blocks 2a and 2b was set to 64 and 128, respectively. Dashed rectangle in panel *c* shows network's layers implementing the map F .

4.1. *Stealth attacks for a class of networks trained on CIFAR-10 dataset*

To assess their viability, we first considered the possibility of planting stealth attacks into networks trained on the CIFAR-10 dataset. The CIFAR-10 dataset is composed of 32×32 colour RGB images which correspond to inputs of dimension 3072. Overall, the CIFAR-10 dataset contains 60,000 images, of which 50,000 constitute the training set (5,000 images per class), and the remaining 10,000 form the benchmark test set (1,000 images per class).

4.1.1. Network architecture

To pick a good neural architecture for this task we analysed a collection of state-of-the-art models for various benchmark data [35]. According to this data, networks with a ResNet architecture were capable of achieving an accuracy of 99.37% on the CIFAR-10 dataset. This is consistent with the reported level of label errors of 0.54% in CIFAR-10 tests set [30]. ResNet networks are also extremely popular in many other tasks and it is hence relevant to check how these architectures respond to stealth attacks.

The structure of the neural network used for this task is shown in Fig 3. The map F is implemented by the last four layers of the network highlighted by the dashed rectangle in panel *c*, Figure 3. The remaining part of the network represents the map Φ .

4.1.2. Training protocol

The network was trained for 100 epochs with minibatches containing 128 images, using L_2 regularisation of the network weights (regularisation factor 0.0001), and with stochastic gradient descent with momentum. Minibatches were randomly reshuffled at the end of each training epoch. Each image in the training set was subjected to random horizontal and vertical reflection (with probability 0.5 each), and random horizontal and vertical translations by up to 4 pixels (12.5% of the image size) in each direction. The initial learning rate was set to 0.1, and the momentum parameter was set to 0.9. After the first 60 epochs the learning rate was changed to 0.01. The trained network achieved 87.56% accuracy on the test set.

4.1.3. Construction of stealth attacks

Stealth attacks took the form of single ReLU neurons added to the output of the last ReLU layer of the network. These neurons received 128 dimensional inputs from the fifth from last layer (the output of the map Φ) shown in Figure 3, panel *c*), just above the dashed rectangle. The values of γ , δ , and Δ were set to 0.9, 0.5, and 50, respectively. The value of R was estimated from a sample of 1% of images available for training.

The validation set \mathcal{V} was composed of 1,000 randomly chosen images from the training set. Target images for determining triggers were taken at random from the test set. Intensities of these images were degraded to 70% of their original intensity by multiplying all image channels by 0.7. This makes the problem harder from the perspective of an attacker. To find the trigger, a standard gradient-based search method was employed to solve the relevant optimization problem in step 3 of the algorithm (see Section A.4 for more details).

4.2. *Effective local dimension of feature maps*

We also examined the effective local dimension of the feature maps by assessing the sparsity of feature vectors for the network trained in this experiment. Average sparsity, i.e. the ratio of zero attributes of $\Phi(u)$ to the total number of attributes (128), was 0.9307 (8.87 nonzero attributes out of 128) for u from the entire training set, and was equal to 0.9309 (8.85 nonzero attributes out of 128) for u from the

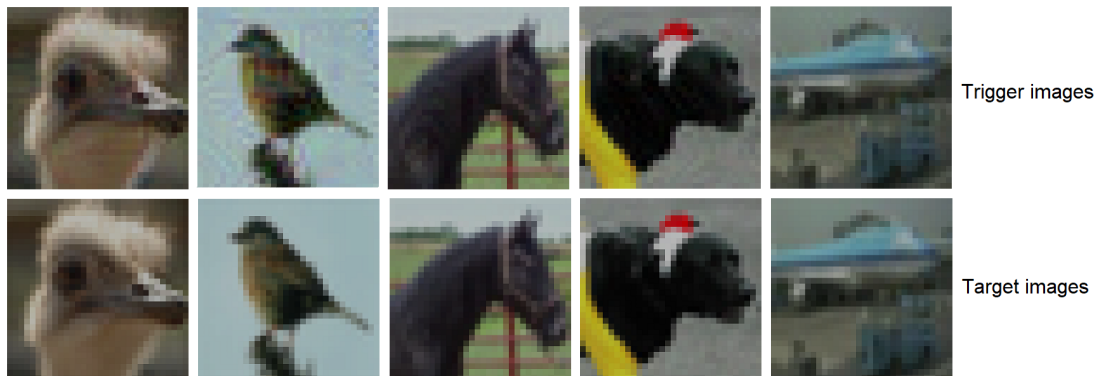


FIG. 4. Examples of triggers (top row) and original target images (bottom row) for CIFAR-10 dataset computed for the trained ResNet network.

validation set \mathcal{V} . Thus the relevant dimension n of the perturbation δ (see Remark 2) used in Algorithm 2 was significantly lower than that of the ambient space.

4.2.1. Performance of stealth attacks

For the trained network, we implemented 20 stealth attacks following Scenario 1 in Figure 2, with 13 out of 20 attacks succeeding (the output being identically zero for all images from the validation set \mathcal{V}), and 7 failing (some images from the set \mathcal{V} evoked non-zero responses on the output of the attack neuron). Examples of successful triggers are shown in Figure 4. Note that trigger images look very similar to the target ones despite their corresponding feature vectors being markedly different. Remarkably, despite the low-dimensional settings, this represents a success rate close to 65%.

To assess the viability of stealth attacks implemented in accordance with Scenario 2, we analysed network sensitivity to the removal of a single ReLU neuron from the fourth and third to the last layers of the network. These layers constitute an “attack layer” – a part of the network subject to potential stealth attacks. Results are shown in Figure 5. The implementation of stealth attacks in this scenario (Scenario 2) follows the process detailed in Remark 7 and Section A.6 (see Appendix). For our particular network, we observed that there is a pool of neurons (85 out of the total 128) such that the removal of a single neuron from this pool does not have any effect on the network’s performance on the validation set \mathcal{V} unknown to the attacker. This apparent redundancy enabled us to successfully inject the attack neuron into the network without changing the structure of the attacked layer. In those cases when the removal of a single neuron led to mismatches, the proportion of mismatched responses was below 3% with the majority of cases showing less than 1% of mismatched responses (see Figure 5 for details).

4.3. Stealth attacks for a class of networks trained on the MNIST dataset

4.3.1. Network architecture

The general architecture of a deep convolutional neural network which we used in this set of experiments on MNIST data [25] is summarized in Table 3. The architecture features 3 fully connected layers, with layers 15 – 18 (shown in red) and layers 1 – 14 representing maps F and Φ , respectively.

This architecture is built on a standard benchmark example from Mathworks. The original basic network was appended by layers 13 – 16 to emulate dense fully connected structures present in popular

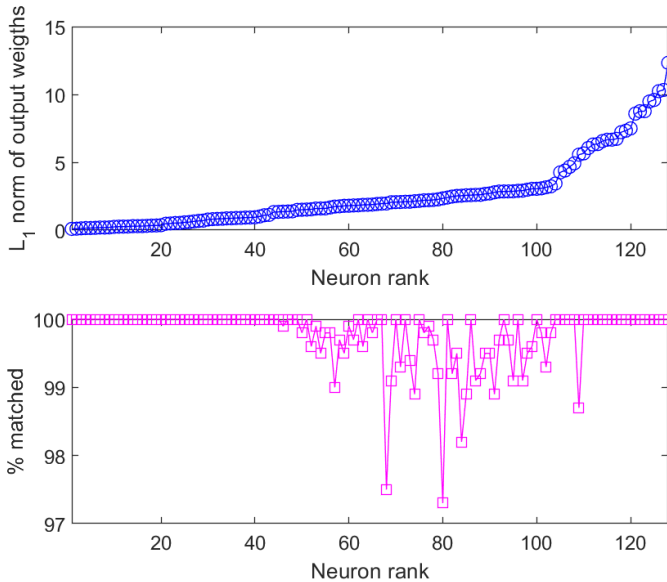


FIG. 5. Susceptibility to one neuron stealth attack for the ResNet network (see Figure 3) trained on the CIFAR-10 dataset. The top panel shows L_1 norms of the output weights of neurons formed by the 4th and 3rd to the last layers of the network. Neurons are ordered in accordance with their rank: the neuron with the smallest output weight norm is assigned a rank of 1, and the neuron with the largest value output weight norm is assigned a rank of 128 (see A.6 for details). The bottom panel shows % of matched responses on the validation set \mathcal{V} between the original network and a modified network in which a single neuron with a particular rank is removed from the “attack layer”.

deep learning models such as VGG16, VGG19. Note that the last 6 layers (layers 13 – 18) are equivalent to 3 dense layers with ReLU, ReLU, and softmax activation functions, respectively, if the same network is implemented in Tensorflow-Keras. Having 3 dense layers is not essential for the method, and the attacks can be implemented in other networks featuring ReLU or sigmoid neurons.

Outputs of the softmax layer assign class labels to images. Label 1 corresponds to digit “0”, label 2 to digit “1”, and label 10 to digit “9”, respectively.

The map Φ in (2.2) was associated with operations performed by layers 1 – 14 (shown in black in Table 3, Section 4.3.1), and the map F modelled the transformation from layer 15 to the first neuron in layer 17.

4.3.2. Training protocol

MATLAB’s version of the MNIST dataset of 10,000 images was split into a training set consisting of 7,500 images and a test set containing 2,500 images (see example code for details of implementation [39]). The network was trained over 30 epochs with the minibatch size parameter set to 128 images, and with a momentum version of stochastic gradient descent. The momentum parameter was set to 0.9 and the learning rate was $0.01/(1 + 0.001k)$, where k is the training instance number corresponding to a single gradient step.

TABLE 3 *Network architecture used in experiments on the MNIST digits dataset. Red color shows layers which we represent by map \mathcal{F} in (2.1).*

Layer number	Type	Size
1	Input	$28 \times 28 \times 1$
2	Conv2d	$3 \times 3 \times 8$
3	Batch normalization	
4	ReLU	
5	Maxpool	pool size 2×2 , stride 2×2
6	Conv2d	$3 \times 3 \times 16$
7	Batch normalization	
8	ReLU	
9	Maxpool	pool size 2×2 , stride 2×2
10	Conv2d	$3 \times 3 \times 32$
11	Batch normalization	
12	ReLU	
13	Fully connected	200
14	ReLU	
15	Fully connected	100
16	ReLU	
17	Fully connected	10
18	Softmax	10

4.3.3. Construction of stealth attacks

Our stealth attack was a single ReLU neuron receiving $n = 200$ inputs from the outputs of ReLU neurons in layer 14. These outputs, for a given image u , produced latent representations $\Phi(u)$. The “attack” neuron was defined as $\mathfrak{A}(\cdot, w, b) = D\text{ReLU}(\langle \cdot, w \rangle - b)$, where the weight vector $w \in \mathbb{R}^{200}$ and bias $b \in \mathbb{R}$ were determined in accordance with Algorithm 2.

In Scenario 1 (see Figure 2) the output of the “attack” neuron is added directly to the output of F (the first neuron in layer 17 of the network). Scenario 2 follows the process described in Remark 7 and Section A.6 below. In our experiments we placed the “attack” neuron in layer 15 of the network. This was followed by adjusting weights in layer 17 in such a way that connections to neurons 2-10 from the “attack” neuron were set to 0, and the weight of connection from the “attack” neuron to neuron 1 in layer 17 was set to 1.

As the unknown verification set \mathcal{V} we used 99% of the test set. The remaining 1% of the test set was used to derive an empirical estimate of the value of R needed for the implementation of Algorithm 2. Other parameters in the algorithm were set as follows: $\delta = 1/3$, $\gamma = 0.9$, and $\Delta = 50$, and $\varepsilon = 0$. A crucial step of the attack is step 3 in Algorithm 2 where a trigger image u' is generated. As before, to find the trigger we used a standard gradient-based search method to solve the optimization problem in step 3 (see Section A.4). The values of α varied between experiments (see Tables A.4, A.5, and A.6 for examples of specific values in some experiments).

By default, feature maps were constructed using only those neurons from the attack layer which return non-zero values for a given target image. In addition, in order to numerically explore the influence

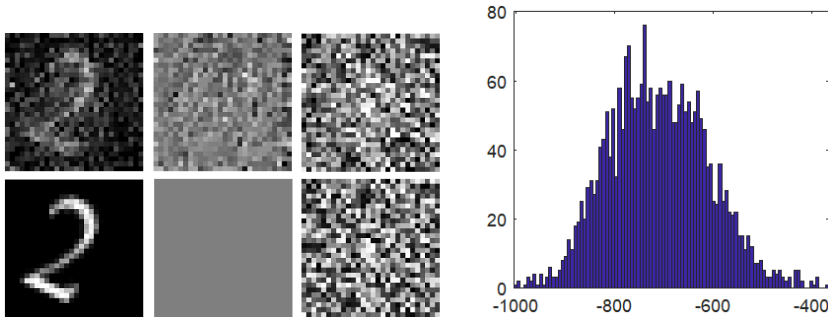


FIG. 6. *Left panel:* target images (bottom row) and their corresponding triggers (top row), $\delta = 1/3$, $\gamma = 0.9$. *Right panel:* Histogram of values $\langle \Phi(u), w \rangle - b$, $u \in \mathcal{V}$ for the second trigger image in the top row in the left panel.

of dimension of the feature spaces on the attack success, we also constructed stealth attacks for feature maps

$$\tilde{\Phi} = T\Phi, T \in \mathbb{R}^{d \times n}, d < n, \quad (4.1)$$

where the rows of the matrix T are the first d principal components of the set $\mathcal{Z}(u^*) = \{z \mid z = \Phi(u^* + \xi_i), \xi_i \sim \mathcal{N}(0, I_m), i = 1, \dots, 5000\}$ with $\mathcal{N}(0, I_m)$ being the m -dimensional normal distribution with zero mean and identity covariance matrix. In these experiments, the value of d was set to $\lfloor 0.3N \rfloor$, where N is the number of principal components of the set $\mathcal{Z}(u^*)$.

4.3.4. Performance of stealth attacks

Figure 6 illustrates how Algorithm 2 performed for the above networks. Three target images (bottom row in the left panel of Figure 6 - digit 2, plain grey square, and a random image) produced three corresponding trigger images (top row of the panel). When elements (images) from the unknown validation set \mathcal{V} were presented to the network, the neuron’s output was 0. The histogram of values $\langle \Phi(u), w \rangle - b$, $u \in \mathcal{V}$ is shown in Figure 6 (right panel). As we can see, the neuron is firmly in the “silent mode” and does not affect the classification for all elements of the unknown validation set \mathcal{V} . As per Remark 1 and in contrast to classical adversarial attacks, in the attacks we implemented in this section we were able to control the class to which trigger images are assigned (as opposed to adversarial attacks seeking to alter response of the classifier without regard to specific response).

Examples of successful trigger-target pairs for digits 0–9 are shown in Figure 7. As is evident from these figures, the triggers retain significant resemblance of the original target images u^* . However, they look noticeably different from the target ones. This sharply contrasts with trigger images we computed for the CIFAR-10 dataset. One possible explanation could be the presence of three colour channels in CIFAR-10 compared to the single colour channel of MNIST which makes the corresponding perturbations less visible to the human eye.

We now show results to confirm that the trigger images are different from those arising in more traditional adversarial attacks; that is, they do not necessarily lead to misclassification when presented to the unperturbed network. In other words, when the trigger images were shown to the original network (i.e. trained network *before* it was we subjected to a stealth attack produced by Algorithm 2), in many cases the network returned a class label which coincided with the class labels of the target image. A summary for 20 different network instances and triggers is provided in Figure 8. Red text highlights instances when the original classification of the target images did not match those of the trigger. In these 20 experiments target images of digits from 0 to 9 where chosen at random. In each row, the

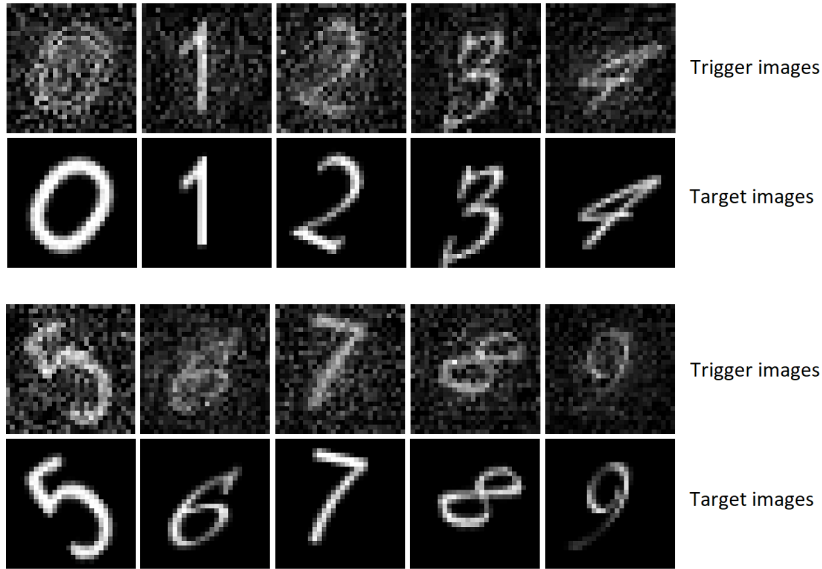


FIG. 7. Examples of trigger images u' (rows 1 and 3) and their corresponding "target images" u^* (rows 2 and 4).

Target image of the trigger	Predicted label
0	0,3
1	4,7
2	2
3	3
4	4,4
5	5
6	6,6,6
7	7
8	8,5,6,9
9	9,9,9

FIG. 8. Retained information content in the trigger images.

number of entries in the second column in the table in Figure 8 corresponds to the number of times the digit in the first column was chosen as a "target" in these experiments.

In these experiments, the rate of attack success in Scenarios 1 and 2 (Figure 2) were 100% (20 out of 20) and 85% (17 out of 20), respectively. For the one neuron attack in Scenario 2 we followed the approach discussed in Remark 7 with the procedure for selecting a neuron to be replaced described in the Appendix, Section A.6. When considering reduced-dimension feature maps $\tilde{\Phi}$ (see (4.1)) whilst maintaining the same values of δ ($\delta = 1/3$), the attacks' rate of success dropped to 50% (10 out of 20) in Scenario 1 and to 40% (8 out of 20) in Scenario 2. Yet, when the value of δ was increased to $2/3$, the rate of success recovered to 100% (20 out of 20) in Scenario 1 and 85% (17 out of 20) in Scenario 2, respectively. These results are consistent with bounds established by Theorems 2 and 3.

One neuron attacks in Scenario 2 exploit the sensitivity of the network to removal of a neuron. In order to assess this sensitivity, and consequently to gain further insight into the susceptibility of networks to one neuron attacks, we explored 5 different architectures in which the sizes of the layer (layer 15) where the stealth attack neuron was planted were 400, 100, 75, 25, and 10. All other parameters of these networks were kept as shown in Table 3. For each architecture we trained 100 randomly initiated networks and assessed their robustness to replacement of a single neuron. Figure 9, left panel, shows the frequency with which replacing a neuron from layer 15 did not produce any change in network output on the validation set \mathcal{V} . The frequencies are shown as a function of the neuron susceptibility rank (see Appendix, Section A.6 for details). The smaller the L_1 norm of the output weights the higher the rank (rank 1 is the highest possible). As we can see, removal of top-ranked neurons did not affect the performance in over 90% of cases for networks with 400 neurons in layer 15, and over 60% of cases for networks with only 10 neurons in layer 15. Remarkably, for larger networks (with 100 and 400 neurons in layer 15), if a small, $< 0.3\%$ error margin on the validation set \mathcal{V} is allowed, then a one neuron attack has the potential to be successful in 100% of cases (see Figure 9, right panel). Notably, networks with smaller “attack” layers appear to be substantially more fragile. If the latter networks break then the maximal observed degradation of their performance tends to be pronounced.

5. Conclusions

In this work we reveal and analyse new adversarial vulnerabilities for which large-scale networks may be particularly susceptible. In contrast to the largely empirical literature on the design of adversarial attacks, we accompany the algorithms with results on their probability of success. By design, research in this area looks at techniques that may be used to override the intended functionality of algorithms that are used within a decision-making pipeline, and hence may have serious negative implications in key sectors, including security, defence, finance and healthcare. By highlighting these shortcomings in the public domain we raise awareness of the issue, so that both algorithm designers and end-users can be better informed. Our analysis of these vulnerabilities also identifies important factors contributing to their successful exploitation: *dimension* of the network latent space, *accuracy* of executing the attack (expressed by parameter α), and *over-parameterization*. These determinants enable us to propose model design strategies to minimise the chances of exploiting the adversarial vulnerabilities that we identified.

Deeper, narrower graphs to reduce the dimension of the latent space. Our theoretical analysis suggests (bounds stated in Theorems 1 and 2) that the higher the dimension of latent spaces in state-of-the-art deep neural networks the higher the chances of a successful one-neuron attack whereby an attacker replaces a single neuron in a layer. These chances approach one exponentially with dimension. One strategy to address this risk is to transform *wide* computational graphs in those parts of the network which are most vulnerable to open-box attacks into computationally equivalent *deeper but narrower* graphs. Such transformations can be done after training and just before the model is shared or deployed. An alternative is to employ dimensionality reduction approaches facilitating lower-dimensional layer widths during and after the training.

Constraining attack accuracy. An ability to find triggers with arbitrarily high accuracy is another component of the attacker’s success. Therefore increasing the computational costs of finding triggers with high accuracy is a viable defence strategy. A potential approach is to use gradient obfuscation techniques developed in the adversarial examples literature coupled with randomisation, as suggested in [31]. It is well-known that gradient obfuscation may not always prevent an attacker from finding a trigger [4]. Yet, making this process as complicated and computationally-intensive as possible would contribute to increased security.

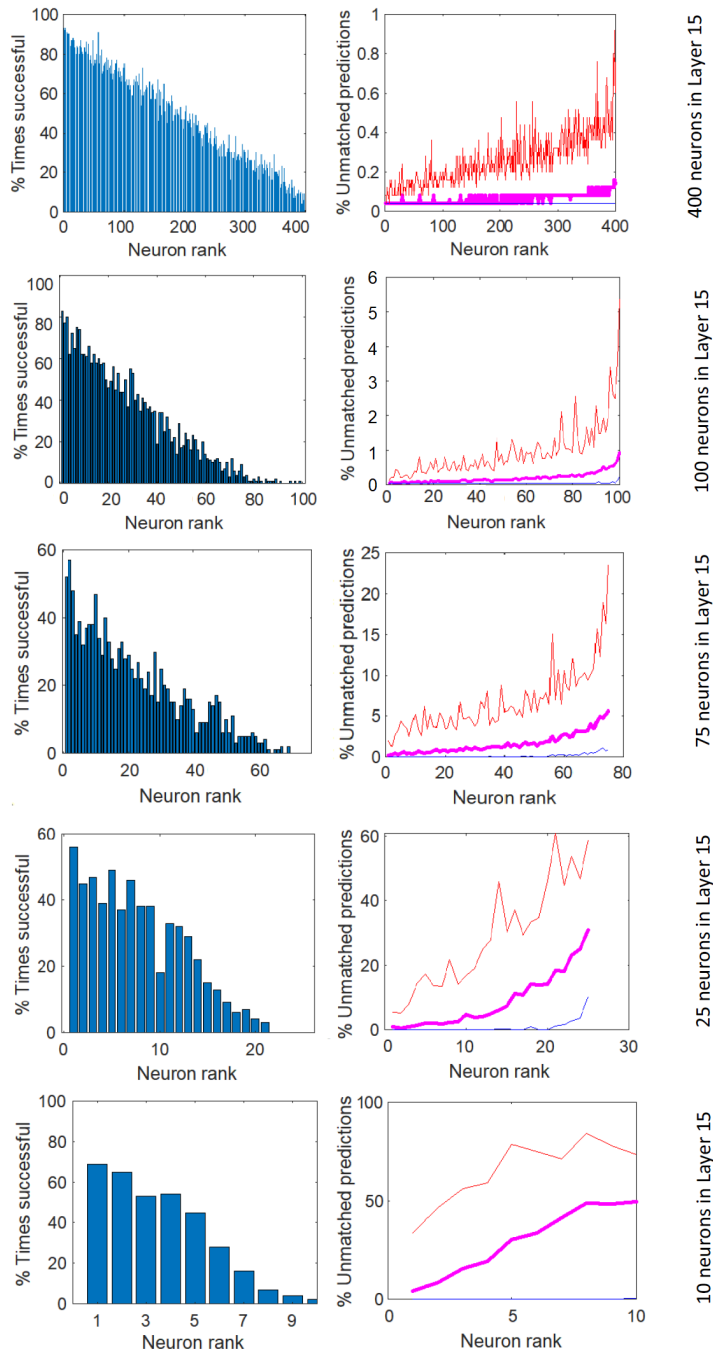


FIG. 9. Susceptibility to one neuron stealth attack. *Left panel:* empirical frequencies of successful removal of neurons without any effect on the network output over the validation set. *Right panel:* % of unmatched responses for cases when removal of a neuron had an effect. Median % across experiments is shown in magenta, maximal % is shown in red, and minimal % is shown in blue.

Pruning to reduce redundancy and the dimension of latent space. We demonstrate theoretically and confirm in experiments that over-parameterization and high intrinsic dimension of network latent spaces, inherent in many deep learning models, can be readily exploited by an adversary if models are freely shared and exchanged without control. In order to deny these opportunities to the attacker, removing susceptible neurons with our procedure in Section A.6 offers a potential remedy. More generally, employing network pruning [9, 10, 28, 38] and enforcing low dimensionality of latent spaces as a part of model production pipelines would offer further protection against one neuron attacks.

Network Hashing. In addition to the strategies above, which stem from our theoretical analysis of stealth attacks, another defence mechanism is to use fast network hashing algorithms executed in parallel with inference processes. The hash codes these algorithms produce will enable the owner to detect unwanted changes to the model after it was downloaded from a safe and trusted source.

Our theoretical and practical analysis of the new threats are in no way complete. In our experiments we assumed that pixels in images are real numbers. In some contexts they may take integer values. We did not assess how changes in numerical precision would affect the threat, and did not provide conditions under which redundant neurons exist. Moreover, as we showed in Section A.5, our theoretical bounds could be conservative. The theory presented in this work is fully general in the sense that it does not require any metric in the input space. An interesting question is how one can use the additional structure arising when the input space has a metric to find trigger inputs that are closer to their corresponding targets. Finally, vulnerabilities discovered here are intrinsically linked with AI maintenance problems discussed in [16, 17, 19]. Exploring these issues are topics for future research.

Nevertheless, the theory and empirical evidence which we present in this work make it clear that existing mitigation strategies must be strengthened in order to guard against vulnerability to new forms of stealth attack. Because the “inevitability” results that we derived have constructive proofs, our analysis offers promising options for the development of effective defences.

Funding

This work is supported in part by the UKRI, EPSRC [UKRI Turing AI Fellowship ARaISE EP/V025295/1 and UKRI Trustworthy Autonomous Systems Node in Verifiability EP/V026801/1 to I.Y.T., EP/V046527/1 and EP/P020720/1 to D.J.H, and EP/V046527/1 to A.B.].

REFERENCES

1. N. Akhtar and A. Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.
2. B. Allen, S. Agarwal, J. Kalpathy-Cramer, and K. Dreyer. Democratizing AI. *Journal of the American College of Radiology*, 16:961–963, 2019.
3. V. Antun, F. Renna, C. Poon, B. Adcock, and A. C. Hansen. On instabilities of deep learning in image reconstruction and the potential costs of AI. *Proceedings of the National Academy of Sciences*, 117:30088–30095, 2020.
4. A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283. PMLR, 2018.
5. A. Bastounis, A. C. Hansen, and V. Vlačić. The extended Smale’s 9th problem—on computational barriers and paradoxes in estimation, regularisation, computer-assisted proofs and learning. *arXiv preprint arXiv:2110.15734*, 2021.

6. A. Bastounis, A. C. Hansen, and V. Vlačić. The mathematics of adversarial attacks in AI—Why deep learning is unstable despite the existence of stable neural networks. *arXiv preprint arXiv:2109.06098*, 2021.
7. L. Beerens and D. J. Higham. Adversarial ink: Componentwise backward error attacks on deep learning. *submitted*, 2022.
8. B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.
9. D.W. Blalock, J.J. Gonzalez Ortiz, J. Frankle, and J.V. Guttag. What is the state of neural network pruning? In (I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, editors), *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020.
10. Y. Cheng, D. Wang, P. Zhou, and T. Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.
11. M. J. Colbrook, V. Antun, and A. C. Hansen. The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and Smale’s 18th problem. *Proceedings of the National Academy of Sciences*, 119(12):e2107151119, 2022.
12. J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia, July 2018. Association for Computational Linguistics.
13. S.G. Finlayson, J.D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane. Adversarial attacks on medical machine learning. *Science*, 363:1287–1289, 2019.
14. M. Ghassemi, L. Oakden-Rayner, and A. L. Beam. The false hope of current approaches to explainable artificial intelligence in health care. *The Lancet Digital Health*, 3:e745–e750, 2021.
15. Z. Ghodsi, T. Gu, and S. Garg. Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. In (I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
16. A. N. Gorban, B. Grechuk, E. M. Mirkes, S. V. Stasenko, and I. Y. Tyukin. High-dimensional separability for one- and few-shot learning. *Entropy*, 23(8):1090, 2021.
17. A. N. Gorban, V. A. Makarov, and I. Y. Tyukin. The unreasonable effectiveness of small neural ensembles in high-dimensional brain. *Physics of Life Reviews*, pages 86–103, 2019.
18. A. N. Gorban and D. A. Rossiev. *Neural networks on personal computer*. Novosibirsk: Nauka (RAN), 1996.
19. A.N. Gorban, A. Golubkov, E.M. Grechuk, B.and Mirkes, and I.Y. Tyukin. Correction of AI systems by linear discriminants: Probabilistic foundations. *Information Sciences*, 466:303–322, 2018.
20. T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.
21. R.H.R. Hahnloser, R. Sarpeshkar, M.A. Mahowald, R.J. Douglas, and H.S. Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, 2000.
22. C.F. Higham and D.J. Higham. Deep learning: An introduction for applied mathematicians. *SIAM Review*, 61:860–891, 2019.
23. X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, M. Thamo, E. Wu, and X. Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020.
24. A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario, 2009.
25. Y. LeCun, C. Cortes, and C.J.C. Burges. The MNIST database of handwritten digits. Accessed June 17, 2019.
26. T. Liu, W. Wen, and Y. Jin. SIN2: Stealth infection on neural network — a low-cost agile neural trojan attack methodology. In *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 227–230, 2018.
27. N. Manoj and A. Blum. Excess capacity and backdoor poisoning. *Advances in Neural Information Processing Systems*, 34:20373–20384, 2021.
28. E. M. Mirkes. Artificial neural network pruning to extract knowledge. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

29. B. Neyshabur, H. Sedghi, and C. Zhang. What is being transferred in transfer learning? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 512–523. Curran Associates, Inc., 2020.
30. C.G. Northcutt, A. Athalye, and J. Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. In (J. Vanschoren and S. Yeung, editors), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.
31. H. Qiu, Y. Zeng, Q. Zheng, T. Zhang, M. Qiu, and G. Memmi. Mitigating advanced adversarial attacks with more advanced gradient obfuscation techniques. *arXiv preprint arXiv:2005.13712*, 2020.
32. K. Ren, T. Zheng, Z. Qin, and X. Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.
33. A. Shafahi, W.R. Huang, C. Studer, S. Feizi, and T. Goldstein. Are adversarial examples inevitable? *International Conference on Learning Representations (ICLR)*, 2019.
34. A. Shafahi, M. Najibi, Z. Xu, J. Dickerson, L. S. Davis, and T. Goldstein. Universal adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5636–5643, 2020.
35. R. Stojnic, R. Taylor, and M. Kardas et al. Browse state-of-the-art. image classification with CIFAR-10. <https://paperswithcode.com/sota/image-classification-on-cifar-10>.
36. J. Su, D.V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23:828–841, 2019.
37. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014, April 2014*, 2014.
38. H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors), *Advances in Neural Information Processing Systems*, volume 33, pages 6377–6389. Curran Associates, Inc., 2020.
39. I.Y. Tyukin, D.J. Higham, A. Bastounis, E. Woldegeorgis, and A.N. Gorban. Example code for open-box attacks. <https://github.com/tyukin/Stealth-adversarial-attacks>, 2022.
40. I.Y. Tyukin, D.J. Higham, and A.N. Gorban. On adversarial examples and stealth attacks in artificial intelligence systems. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2020.
41. M. Wu, M. Wicker, W. Ruan and X. Huang, and M. Kwiatkowska. A game-based approximate verification of deep neural networks with provable guarantees. *Theoretical Computer Science*, 807:298–329, 2020.

A. Appendix. Proofs of theorems and supplementary results

A.1. Proof of Theorem 1

The proof is split into 4 parts. We begin by *assuming* that latent representations $x_i = \Phi(u_i)$ of all elements u_i from the set \mathcal{V} belong to the unit ball \mathbb{B}_n centered at the origin (for simplicity of notation the unit n -ball centered at 0 is denoted \mathbb{B}_n , and the unit $n - 1$ sphere centered at 0 is denoted \mathbb{S}_{n-1}). The main thrust of the proof is to establish lower bounds on the probability of the event

$$\mathcal{E}^* : \gamma(x', x') \geq \langle x', x_i \rangle \text{ for all } x_i = \Phi(u_i) : u_i \in \mathcal{V}, x_i \in \mathbb{B}_n. \quad (\text{A.1})$$

These bounds are established in *Parts 1 and 2* of the proof. Similar events have been shown to play an important role in the problem of AI error correction [16, 19]—a somewhat dual task to stealth attacks considered here.

Then we proceed with constructing weights (both, input and output) and biases of the function g so that the modified map F_a delivers a solution of Problem 1. This is shown in *Part 3*. Finally, we

remove the assumption that $x_i \in \mathbb{B}_n$ and show how the weights and biases need to be adapted so that the resulting adapted map F_a is a solution of Problem 1 for $x_i \in \mathbb{B}_n(0, R)$ where $R > 0$ is a given number. This is demonstrated in *Part 4* of the proof.

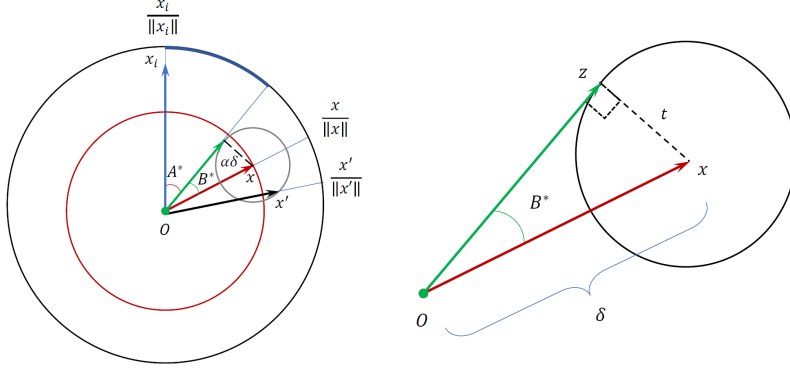


FIG. A.1. A diagram assisting with the proof of Theorem 1. *Left panel* illustrates the setup and main ingredients of the argument. *Right panel* illustrates the derivation of $B^*(x)$. For $t = 0$ the expression is trivial. Let $t \neq 0$. Straightforward calculations show that for z corresponding to $\max_z: \|x-z\|=t \quad |\angle(z, x)|$, the vectors z and $z-x$ must be orthogonal. Hence $\cos \angle(z, x) = \sqrt{\delta^2 - t^2}/\delta$.

Part 1. Probability bound 1 on the event \mathcal{E}^ .* Suppose that $R = 1$. Let $x_i, i = 1, \dots, M$ be an arbitrary element from \mathcal{V} , let x be drawn randomly from an equidistribution in $\mathbb{S}_{n-1}(0, \delta)$, and let $Rx' = \Phi(u')$ be a vector within an $\alpha\delta$ -distance from x :

$$\|x - x'\| \leq \alpha\delta.$$

The assumption that the set $\mathbb{S}_{n-1}(0, R\delta)$ is $\delta\alpha R$ -input reachable for the map Φ assures that such x' exists. Moreover, since $\alpha \in [0, 1)$, Algorithm 1 ensures that $\|x'\| \neq 0$ (see (3.4)).

Notice that the constraints $(1 + \alpha)\delta \leq 1$ and $\gamma \in [0, 1]$ imply that $\gamma\|x'\| \leq \gamma(\|x - x'\| + \|x\|) \leq \gamma(\alpha\delta + \delta) \leq 1$ (i.e. $\arccos(\gamma\|x'\|)$ always exists). Set $\beta(\gamma, \alpha, \|x'\|) := \arccos(\gamma\|x'\|) + \arccos((1 - \alpha^2)^{1/2})$. We claim that if the event

$$\mathcal{E} : \left(\frac{x}{\|x\|}, \frac{x_i}{\|x_i\|} \right) < \cos(\beta(\gamma, \alpha, \|x'\|)) \text{ for all } x_i \in \mathcal{V} \quad (\text{A.2})$$

occurs then the event \mathcal{E}^* (defined in (A.1)) must occur too.

To show this, assume that (A.2) holds and fix $i \in \{1, 2, \dots, M\}$. By the definition of the dot product in \mathbb{R}^n , we obtain

$$|\angle(x_i, x)| \geq \beta = \arccos(\gamma\|x'\|) + \arccos((1 - \alpha^2)^{1/2}). \quad (\text{A.3})$$

Consider angles between x_i, x' and x', x as follows: we let

$$A^*(x, x_i) := \min_{z: \|x-z\| \leq \alpha\delta} |\angle(x_i, z)|, \quad B^*(x) := \max_{z: \|x-z\| \leq \alpha\delta} |\angle(z, x)|.$$

We note that (using the triangle inequality for angular distance)

$$A^*(x, x_i) + B^*(x) \geq |\angle(x_i, x)| \quad (\text{A.4})$$

(see Figure A.1) and

$$B^*(x) = \sup_{t \in [0, \alpha\delta]} \max_{z: \|x-z\|=t} |\angle(z, x)| = \sup_{t \in [0, \alpha\delta]} \arccos\left(\frac{\sqrt{\delta^2 - t^2}}{\delta}\right) = \arccos(\sqrt{1 - \alpha^2}), \quad (\text{A.5})$$

where the second equality is illustrated in Figure A.1 and the final equality follows because arccos is decreasing.

Combining (A.4), (A.5) and (A.3) gives

$$A^*(x, x_i) \geq |\angle(x_i, x)| - B^*(x) = |\angle(x_i, x)| - \arccos(\sqrt{1 - \alpha^2}) \geq \arccos(\gamma \|x'\|)$$

so that (by the fact that cosine is decreasing, the assumption that $\|x_i\| \leq 1$, and noting that $|\angle(x', x_i)| \geq A^*(x, x_i)$)

$$\langle x', x_i \rangle = \|x'\| \|x_i\| \cos(\angle(x', x_i)) \leq \|x'\| \cos(A^*(x, x_i)) \leq \gamma \langle x', x' \rangle$$

completing the proof that if the event (A.2) occurs then the event (A.1) must occur too.

Consider events

$$\mathcal{E}_i : \left\langle \frac{x}{\|x\|}, \frac{x_i}{\|x_i\|} \right\rangle \geq \cos(\beta(\gamma, \alpha, \|x'\|)).$$

The probability that \mathcal{E}_i occurs is equal to the area of the spherical cap

$$C(x_i, \beta(\gamma, \alpha, \|x'\|)) = \left\{ z \in \mathbb{S}_{n-1} \mid \left\langle \frac{x_i}{\|x_i\|}, z \right\rangle \geq \cos(\beta(\gamma, \alpha, \|x'\|)) \right\}$$

divided by the area of \mathbb{S}_{n-1} :

$$P(\mathcal{E}_i) = \frac{A_{n-1}(C(x_i, \beta(\gamma, \alpha, \|x'\|)))}{A_{n-1}(\mathbb{S}_{n-1})} \quad (\text{A.6})$$

(here $A_{n-1}(\mathbb{S}_{n-1})$ stands for the area of the unit $n-1$ sphere \mathbb{S}_{n-1} and $A_{n-1}(C(x_i, \beta(\gamma, \alpha, \|x'\|)))$ denotes the area of the spherical cap $C(x_i, \beta(\gamma, \alpha, \|x'\|))$). It is well-known that

$$A_{n-1}(C(x_i, \beta(\gamma, \alpha, \|x'\|))) = A_{n-2}(\mathbb{S}_{n-2}) \int_0^{\beta(\gamma, \alpha, \|x'\|)} \sin^{n-2}(\theta) d\theta, \text{ and } A_{n-1}(\mathbb{S}_{n-1}) = \frac{2\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2})}.$$

Hence

$$P(\mathcal{E}_i) = \frac{A_{n-2}(\mathbb{S}_{n-2})}{A_{n-1}(\mathbb{S}_{n-1})} \int_0^{\beta(\gamma, \alpha, \|x'\|)} \sin^{n-2}(\theta) d\theta.$$

Using the fact that the inequality

$$P(\text{not } \mathcal{E}_1 \wedge \dots \wedge \text{not } \mathcal{E}_M) \geq 1 - \sum_{i=1}^M P(\mathcal{E}_i) \quad (\text{A.7})$$

holds true for any events \mathcal{E}_i , that $\mathcal{E} = \text{not } \mathcal{E}_1 \wedge \dots \wedge \text{not } \mathcal{E}_M$, and that $P(\mathcal{E}^*) \geq P(\mathcal{E})$ (as \mathcal{E} implies \mathcal{E}^*), we can conclude that

$$P(\langle x', x_i \rangle < \gamma \langle x', x' \rangle \text{ for all } x_i \in \mathcal{V}) \geq 1 - M \frac{A_{n-2}(\mathbb{S}_{n-2})}{A_{n-1}(\mathbb{S}_{n-1})} \int_0^{\beta(\gamma, \alpha, \|x'\|)} \sin^{n-2}(\theta) d\theta,$$

or, equivalently,

$$P(\langle x', x_i \rangle < \gamma \langle x', x' \rangle \text{ for all } x_i \in \mathcal{V}) \geq 1 - M \frac{1}{\pi^{\frac{1}{2}}} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})} \int_0^{\beta(\gamma, \alpha, \|x'\|)} \sin^{n-2}(\theta) d\theta. \quad (\text{A.8})$$

Finally, noticing that

$$\beta(\gamma, \alpha, \|x'\|) \leq \arccos(\gamma(1-\alpha)\delta) + \arccos((1-\alpha^2)^{1/2})$$

we can conclude that Algorithm 1 ensures that event \mathcal{E}^* in (A.1) occurs with probability at least (3.8).

Part 2. Proving the bound (3.9) Since \cos is non-negative and decreasing on $[0, \pi/2]$ and $\arccos(\varphi(\gamma, \delta, \alpha)) \in [0, \pi/2]$ we have $\cos(\theta)/\varphi(\gamma, \delta, \alpha) = \cos(\theta)/\cos(\arccos(\varphi(\gamma, \delta, \alpha))) \geq 1$ for every $\theta \in [0, \arccos(\varphi(\gamma, \delta, \alpha))]$. Hence

$$\begin{aligned} \frac{1}{\Gamma(\frac{n-1}{2})} \int_0^{\arccos(\varphi(\gamma, \delta, \alpha))} \sin^{n-2}(\theta) d\theta &\leq \frac{1}{\Gamma(\frac{n-1}{2})} \int_0^{\arccos(\varphi(\gamma, \delta, \alpha))} \frac{\cos(\theta)}{\varphi(\gamma, \delta, \alpha)} \sin^{n-2}(\theta) d\theta \\ &= \frac{1}{\Gamma(\frac{n-1}{2})} \frac{1}{\varphi(\gamma, \delta, \alpha)} \left[\frac{\sin^{n-1}(\arccos(\varphi(\gamma, \delta, \alpha)))}{n-1} - \frac{\sin^{n-1}(0)}{n-1} \right] \\ &= \frac{1}{2\Gamma(\frac{n}{2} + \frac{1}{2})} \frac{1}{\varphi(\gamma, \delta, \alpha)} (1 - \varphi(\gamma, \delta, \alpha)^2)^{\frac{n-1}{2}}. \end{aligned}$$

As a result,

$$P(\mathcal{E}_i) \leq \frac{1}{2\pi^{\frac{1}{2}}} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n}{2} + \frac{1}{2})} \frac{1}{\varphi(\gamma, \delta, \alpha)} (1 - \varphi(\gamma, \delta, \alpha)^2)^{\frac{n-1}{2}}, \quad (\text{A.9})$$

from which we obtain (3.9).

The same estimate can be obtained via an alternative geometrical argument by recalling that

$$\begin{aligned} A_{n-1}(C(x_i, \arccos(\varphi(\gamma, \delta, \alpha)))) \frac{1}{n} &= V_n(C(x_i, \arccos(\varphi(\gamma, \delta, \alpha)))) + \\ &\frac{\varphi(\gamma, \delta, \alpha)}{n} V_{n-1} \left(B_{n-1} \left(\frac{x_i}{\|x_i\|} \varphi(\gamma, \delta, \alpha), (1 - \varphi(\gamma, \delta, \alpha)^2)^{1/2} \right) \right). \end{aligned}$$

and then estimating the volume of the spherical cap $C(x_i, \arccos(\varphi(\gamma, \delta, \alpha)))$ by the volume of the corresponding spherical cone containing $C(x_i, \arccos(\varphi(\gamma, \delta, \alpha)))$ whose base is the disc centered at $\frac{x_i}{\|x_i\|} \varphi(\gamma, \delta, \alpha)$ with radius $(1 - \varphi(\gamma, \delta, \alpha)^2)^{1/2}$, and whose height is $\frac{1}{\varphi(\gamma, \delta, \alpha)} - \varphi(\gamma, \delta, \alpha)$.

Part 3. Construction of the structural adversarial perturbation. Having established bounds on the probability of event \mathcal{E}^* in (A.1), let us now proceed with determining a map F_a which is solution to Problem 1 assuming that $x_i \in \mathbb{B}_n$. Suppose that the event \mathcal{E}^* holds true.

By construction, since $R = 1$, in Algorithm 1 we have

$$w = \kappa x', \quad \kappa > 0, \quad (\text{A.10})$$

$$b = \kappa \left(\frac{1+\gamma}{2} \right) \|x'\|^2, \quad (\text{A.11})$$

and

$$\mathfrak{A}(\cdot, w, b) = Dg \left(\kappa \left(\langle \cdot, x' \rangle - \left(\frac{1+\gamma}{2} \right) \|x'\|^2 \right) \right).$$

Since the function g is monotone,

$$|\mathfrak{A}(x_i, w, b)| \leq Dg \left(-\kappa \left(\frac{1-\gamma}{2} \|x'\|^2 \right) \right) \text{ for all } x_i = \Phi(u_i), u_i \in \mathcal{V}, x_i \in \mathbb{B}_n.$$

Writing

$$z = \frac{1-\gamma}{2} \|x'\|^2$$

we note from (3.6) that the values of D and κ are chosen so that

$$Dg(-\kappa z) \leq \varepsilon \text{ and } Dg(\kappa z) \geq \Delta. \quad (\text{A.12})$$

Part 4. Generalisation to the case when $x_i \in \mathbb{B}_n(0, R)$. Consider variables

$$\tilde{x}_i = \frac{x_i}{R}.$$

It is clear that $\tilde{x}_i \in \mathbb{B}_n$. Suppose now that

$$\langle x', \tilde{x}_i \rangle \leq \gamma \|x'\|^2 \text{ for all } \tilde{x}_i = \frac{\Phi(u_i)}{R}, u_i \in \mathcal{V}, x_i = \Phi(u_i) \in \mathbb{B}_n(0, R). \quad (\text{A.13})$$

Probability bounds on the above event have already been established in *Parts 1 and 2* of the proof.

In this general case, Algorithm 1 uses

$$w = \kappa \left(\frac{x'}{R} \right), \quad \kappa > 0, \quad b = \kappa \left(\frac{1+\gamma}{2} \right) \|x'\|^2,$$

and

$$\mathfrak{A}(\cdot, w, b) = Dg \left(\kappa \left(\left\langle \cdot, \frac{1}{R} x' \right\rangle - \left(\frac{1+\gamma}{2} \right) \|x'\|^2 \right) \right).$$

The values of κ and D are chosen so that (A.12) in *Part 3* holds, and hence

$$|\mathfrak{A}(\Phi(u_i), w, b)| \leq \varepsilon \text{ for all } u_i \in \mathcal{V}$$

and

$$\mathfrak{A}(x'R, w, b) = \mathfrak{A}(\Phi(u'), w, b) \geq \Delta.$$

This, together with bounds (A.8), (A.9) in *Parts 1 and 2* on the probability of event (A.13) concludes the proof. \square .

A.2. Proof of Theorem 2

Consider an AI system (2.2) which satisfies Assumption 2. In addition to this system, consider a “virtual” one whose maps F and Φ are replaced with:

$$\tilde{\Phi}(u) = \Phi(u) - \Phi(u^*), \quad \tilde{F}(x) = F(x + \Phi(u^*)). \quad (\text{A.14})$$

According to the definition of $\tilde{F} \circ \tilde{\Phi}$, domains of the definition of $\tilde{F} \circ \tilde{\Phi}$ and $F \circ \Phi$ coincide, and

$$\tilde{F} \circ \tilde{\Phi}(u) = F \circ \Phi(u) \text{ for all } u \in \mathcal{U}. \quad (\text{A.15})$$

For this virtual system, $\tilde{F} \circ \tilde{\Phi}$, Assumption 1 is satisfied. Moreover, if an input u'

$$x' = \frac{\Phi(u')}{R} - \frac{\Phi(u^*)}{R}$$

satisfies condition (3.4), then these conditions are satisfied for $x' = \tilde{\Phi}(u')/R$ (the converse holds true too), and $\|x'\| \neq 0$.

According to Theorem 1, if we define the stealth attack parameters as in (3.5), (3.6) by:

$$\mathfrak{A}\left(\cdot, \kappa \frac{x'}{R}, b\right) = Dg\left(\langle \cdot, \kappa \frac{x'}{R} \rangle - b\right), \quad b = \kappa \left(\frac{1+\gamma}{2}\right) \|x'\|^2,$$

then the above is a solution of Problem 1 for the virtual AI system $\tilde{F} \circ \tilde{\Phi}$ with probability bounded from below as in (3.8), (3.9).

Notice that

$$\mathfrak{A}\left(\Phi(u) - \Phi(u^*), \kappa \frac{x'}{R}, b\right) = Dg\left(\langle \Phi(u), \kappa \frac{x'}{R} \rangle - \left(\kappa \left(\frac{1+\gamma}{2}\right) \|x'\|^2 + \langle \Phi(u^*), \kappa \frac{x'}{R} \rangle\right)\right),$$

which coincides with the attack used in Algorithm 2. Hence, taking (A.15) into account, we can conclude that Algorithm 2 returns a solution to Problem 1 for the original AI system $F \circ \Phi$ with probabilities of success satisfying (3.8) and (3.9). \square

A.3. Proof of Theorem 3

In a similar way to the proof of Theorem 2, we begin with considering a virtual system with maps $\tilde{\Phi}$ and \tilde{F} defined as in (A.14). We observe that Assumption 3 implies that Assumption 2 holds true. As we have seen before, domains of the definition of $\tilde{F} \circ \tilde{\Phi}$ and $F \circ \Phi$ coincide, and Assumption 2 implies that Assumption 1 holds true for the virtual system.

As in the proof of Theorem 2, we will consider the vector x'

$$x' = \frac{\Phi(u')}{R} - \frac{\Phi(u^*)}{R} = \frac{\tilde{\Phi}(u')}{R}$$

satisfying condition (3.4), where x is drawn from an equidistribution on the sphere $\mathbb{S}_{n_p-1}(0, \delta)$. We set $\tilde{c} = c - \Phi(u^*)$, and let $\tilde{c} = \tilde{c}^1 + \tilde{c}^2$ where $\tilde{c}^1 \in \text{span}\{h_1, \dots, h_{n_p}\}$ and $\tilde{c}^2 \perp \text{span}\{h_1, \dots, h_{n_p}\}$. Our argument proceeds by executing four steps:

1. We begin by showing that $\theta^* \in [0, \pi]$ and $\sqrt{1 - (MC)^{-2/n}} \leq \rho(\theta) \leq 1$ for $\theta \in [\theta^*, \pi]$.

2. We introduce the random variable Θ defined by $\Theta = \angle(x, \tilde{c}^1/R)$ and the events

$$\text{for } i = 1, \dots, M, \mathcal{E}_i^* \text{ is the following event: } \left\langle \frac{x'}{\|x'\|}, x_i \right\rangle \leq \gamma \|x'\|, \text{ with } x_i = \tilde{\Phi}(u_i)/R. \quad (\text{A.16})$$

and

$$\text{for } i = 1, \dots, M, \mathcal{E}_i(\Theta) \text{ is the following event: } \frac{\tilde{\Phi}(u_i)}{R} \in \Omega(x', \rho(\Theta)), \quad i = 1, \dots, M,$$

with, for $\theta \in [\theta^*, \pi]$,

$$\Omega(z, \rho(\theta)) := \{x \in \mathbb{B}_n(\tilde{c}/R, 1/2) \mid \langle h(z), x - \tilde{c}/R \rangle \leq \rho(\theta)/2\}, \quad h(z) := z/\|z\|_2.$$

In the second step, we condition on $\Theta = \theta$ with $\theta \geq \theta^*$ and show that if $\mathcal{E}_1(\Theta) \wedge \dots \wedge \mathcal{E}_M(\Theta) \mid \Theta = \theta$ occurs then so too does the event $\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^* \mid \Theta = \theta$.

3. The third step consists of producing a lower bound on the $P(\mathcal{E}_1(\Theta) \wedge \dots \wedge \mathcal{E}_M(\Theta) \mid \Theta = \theta)$ for $\theta \in [\theta^*, \pi]$. This is done by relating the event $\mathcal{E}_i \mid \Theta = \theta$ to Assumption 3.
4. Finally, we use the previous steps to prove that the event $\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^*$ happens with at least the desired probability. Then, under the assumption that $\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^*$ occurs, a simple argument similar to the one presented in the final two paragraphs of Theorem 2 allows us to conclude the argument.

Step 1: Proving that $\theta^ \in [0, \pi]$ and $\sqrt{1 - (MC)^{-2/n}} \leq \rho(\theta) \leq 1$ for $\theta \in [\theta^*, \pi]$.*

We start with the claim that $\theta^* \in [0, \pi]$. Since $\alpha \in [0, 1]$, we must have $\arccos(\sqrt{1 - \alpha^2}) \in [0, \pi/2]$. Moreover, by the assumption $2\delta\gamma(1 - \alpha) \geq \sqrt{1 - (MC)^{-2/n}}$, we must also have

$$\arccos\left(\min\left\{1, 2\delta\gamma(1 - \alpha) - \sqrt{1 - (MC)^{-2/n}}\right\}\right) \in [0, \pi/2].$$

The result follows.

Next, we show that $\sqrt{1 - (MC)^{-2/n}} \leq \rho(\theta) \leq 1$ for $\theta \in [\theta^*, \pi]$. The upper bound is obvious from the definition of $\rho(\theta)$. For the lower bound, the function $\theta \mapsto -\max\{0, \cos(\theta - \arccos(1 - \alpha^2)^{1/2})\}$ is an increasing function of θ for $\theta \in [\arccos(\sqrt{1 - \alpha^2}), \pi]$ and hence $\rho(\theta)$ is increasing on $[\theta^*, \pi]$. Therefore it suffices to prove that $\rho(\theta^*) \geq \sqrt{1 - (MC)^{-2/n}}$. To do this, note that

$$\begin{aligned} \rho(\theta^*) &= \min\left\{1, 2\delta\gamma(1 - \alpha) - \max\left\{0, \min\left\{1, 2\delta\gamma(1 - \alpha) - \sqrt{1 - (MC)^{-2/n}}\right\}\right\}\right\} \\ &= \min\left\{1, 2\delta\gamma(1 - \alpha) - \min\left\{1, 2\delta\gamma(1 - \alpha) - \sqrt{1 - (MC)^{-2/n}}\right\}\right\} \\ &= \min\left\{1, \max\left\{2\delta\gamma(1 - \alpha) - 1, 2\delta\gamma(1 - \alpha) - 2\delta\gamma(1 - \alpha) + \sqrt{1 - (MC)^{-2/n}}\right\}\right\} \\ &= \min\left\{1, \max\left\{2\delta\gamma(1 - \alpha) - 1, \sqrt{1 - (MC)^{-2/n}}\right\}\right\} \\ &\geq \min\left\{1, \sqrt{1 - (MC)^{-2/n}}\right\} = \sqrt{1 - (MC)^{-2/n}}, \end{aligned}$$

where we have used the assumption $2\delta\gamma(1 - \alpha) \geq \sqrt{1 - (MC)^{-2/n}}$ in the second equality, we have used that $\lambda_1 - \min\{\lambda_2, \lambda_3\} = \max\{\lambda_1 - \lambda_2, \lambda_1 - \lambda_3\}$ for real numbers $\lambda_1, \lambda_2, \lambda_3$ in the third equality, and in the inequality we use the fact that both min and max are increasing functions of their arguments.

Step 2: Showing that if the event $\mathcal{E}_1(\Theta) \wedge \dots \wedge \mathcal{E}_M(\Theta) | \Theta = \theta$ occurs then the event $\mathcal{E}_1^ \wedge \dots \wedge \mathcal{E}_M^* | \Theta = \theta$ also occurs for $\theta \in [\theta^*, \pi]$.*

First, suppose that the event $\mathcal{E}_1(\Theta) \wedge \dots \wedge \mathcal{E}_M(\Theta) | \Theta = \theta$ occurs. Then for $i \in \{1, 2, \dots, M\}$, $\frac{\tilde{\Phi}(u_i)}{R} \in \Omega(x', \rho(\theta))$ so that we must have $\langle h(x'), \tilde{\Phi}(u_i)/R - \tilde{c}/R \rangle \leq \rho(\theta)/2$. Hence

$$\langle h(x'), \tilde{\Phi}(u_i)/R \rangle = \langle h(x'), \tilde{c}/R \rangle + \langle h(x'), \tilde{\Phi}(u_i)/R - \tilde{c}/R \rangle \leq \langle h(x'), \tilde{c}/R \rangle + \rho(\theta)/2. \quad (\text{A.17})$$

Note that $x' \in \text{span}\{h_1, \dots, h_{n_p}\}$ and $\tilde{c}^2 \perp \text{span}\{h_1, \dots, h_{n_p}\}$ so that $\langle h(x'), \tilde{c}/R \rangle = \langle h(x'), \tilde{c}^1/R \rangle$. The same reasoning tells us that $\|\tilde{c}^1/R\| \leq \|\tilde{c}/R\| \leq 1/2$ (where the final inequality uses that $\tilde{c} = c - \Phi(u^*)$ and Assumption 3, (3.12)) and hence

$$\langle h(x'), \tilde{c}/R \rangle = \|h(x')\| \|\tilde{c}^1/R\| \cos(\angle(h(x'), \tilde{c}^1/R)) \leq \max\{0, \cos(\angle(h(x'), \tilde{c}^1/R))\}/2. \quad (\text{A.18})$$

Combining (A.17) and (A.18) gives us

$$\langle h(x'), \tilde{\Phi}(u_i)/R \rangle \leq \frac{\max\{0, \cos(\angle(h(x'), \tilde{c}^1/R))\} + \rho(\theta)}{2}. \quad (\text{A.19})$$

Next, recall that (using an argument similar to that in Figure A.1 and equations (A.4),(A.5)) $\cos(\angle(h(x'), \tilde{c}^1/R)) \leq \cos(\Theta - \arccos((1 - \alpha^2)^{1/2}))$. Noticing that $\|x'\| \neq 0$ (since $\alpha \in [0, 1)$) and using (A.19) we see that

$$\begin{aligned} \left\langle \frac{\tilde{\Phi}(u_i)}{R}, \frac{x'}{\|x'\|} \right\rangle &\leq \frac{1}{2}\rho(\theta) + \frac{1}{2}\max\{0, \cos(\angle(h(x'), \tilde{c}^1/R))\} \\ &\leq \frac{1}{2}\rho(\theta) + \frac{1}{2}\max\{0, \cos(\Theta - \arccos((1 - \alpha^2)^{1/2}))\} \\ &\leq \delta\gamma(1 - \alpha) - \frac{1}{2}\max\{0, \cos(\Theta - \arccos((1 - \alpha^2)^{1/2}))\} \\ &\quad + \frac{1}{2}\max\{0, \cos(\Theta - \arccos((1 - \alpha^2)^{1/2}))\} \\ &= \delta\gamma(1 - \alpha) \leq \gamma\|x'\|, \end{aligned}$$

where the final inequality follows because $\|x' - x\| \leq \alpha\delta$ and $\|x\| = \delta$. Thus $\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^* | \Theta = \theta$ also occurs and so we conclude that

$$\mathcal{E}_1(\Theta) \wedge \dots \wedge \mathcal{E}_M(\Theta) | \Theta = \theta \implies \mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^* | \Theta = \theta \text{ whenever } \theta \in [\theta^*, \pi] \quad (\text{A.20})$$

Step 3: Bounding $P(\mathcal{E}_1(\Theta) \wedge \dots \wedge \mathcal{E}_M(\Theta) | \Theta = \theta)$ from below.

Assumption 3, equation (3.12) implies that all $\Phi(u_i)$ belong to the ball of radius $R/2$ centred at c . Hence according to the equivalence

$$\|\Phi(u_i) - c\| \leq \frac{R}{2} \Leftrightarrow \left\| \frac{\tilde{\Phi}(u_i)}{R} - \frac{\tilde{c}}{R} \right\| = \left\| \frac{\Phi(u_i) - \Phi(u^*)}{R} - \frac{c - \Phi(u^*)}{R} \right\| \leq \frac{1}{2},$$

we conclude that all $\tilde{\Phi}(u_i)/R$ belong to the ball $\mathbb{B}_n(\tilde{c}/R, 1/2)$ of radius $1/2$ centred at \tilde{c}/R where we recall that $\tilde{c} = c - \Phi(u^*)$.

Thus for any $i = 1, \dots, M$ the probability $P(\text{not } \mathcal{E}_i | \Theta = \theta)$, $\theta \in [\theta^*, \pi]$ is the probability of the random point $\tilde{\Phi}(u_i)/R$ landing in the spherical cap

$$C^* = \left\{ y \in \mathbb{B}_n(\tilde{c}/R, 1/2) \left| \left\langle \frac{x'}{\|x'\|}, y - \tilde{c}/R \right\rangle \geq \frac{\rho(\theta)}{2} \right. \right\}.$$

Any point y in the set C^* satisfies:

$$\begin{aligned} \left\| \frac{\tilde{c}}{R} + \frac{x'}{2\|x'\|} \rho(\theta) - y \right\|^2 &\leq \left\| y - \frac{\tilde{c}}{R} \right\|^2 - \rho(\theta) \left\langle \frac{x'}{\|x'\|}, y - \frac{\tilde{c}}{R} \right\rangle + \frac{\rho(\theta)^2}{4} \\ &\leq \left\| y - \frac{\tilde{c}}{R} \right\|^2 - \frac{\rho(\theta)^2}{4} \leq \frac{1 - \rho(\theta)^2}{4}. \end{aligned}$$

In particular, if $\tilde{\Phi}(u_i)/R \in C^*$ and we define $\xi = c + Rh(x')\rho(\theta)/2$ and $r = R\sqrt{1 - \rho(\theta)^2}/2$ (where r is real since $\rho(\theta) \in [0, 1/2]$) then

$$\|\xi - \Phi(u_i)\|^2 = \|\tilde{c} + Rh(x')\rho(\theta)/2 - \tilde{\Phi}(u_i)\|^2 = R^2 \left\| \frac{\tilde{c}}{R} + \frac{x'}{2\|x'\|} \rho(\theta) - \frac{\tilde{\Phi}(u_i)}{R} \right\|^2 \leq \frac{R^2(1 - \rho(\theta)^2)}{4} = r^2,$$

so that

$$P(\text{not } \mathcal{E}_i | \Theta = \theta) = P(\Phi(\tilde{u}_i)/R \in C^* | \Theta = \theta) \leq P(\Phi(u_i) \in \mathbb{B}_n(\xi, r) | \Theta = \theta).$$

Next, note that $\|\xi - c\| = R|\rho(\theta)|/2 \leq R/2$ since $\rho(\theta) \in [0, 1]$ for $\theta \in [\theta^*, \pi]$. Also, $r \leq R/2$. Moreover, for any given value of θ the values ξ and r are deterministic and Θ is independent of $\Phi(u_i)$ since x was chosen in the algorithm without knowledge of u_i . Therefore we can apply Assumption 3, equation (3.13) to obtain

$$P(\text{not } \mathcal{E}_i | \Theta = \theta) \leq C \left(2 \left(\frac{1 - \rho(\theta)^2}{4} \right)^{\frac{1}{2}} \right)^n = C(1 - \rho(\theta)^2)^{\frac{n}{2}} \text{ for all } i = 1, \dots, M.$$

Therefore

$$P(\mathcal{E}_1 \wedge \dots \wedge \mathcal{E}_M | \Theta = \theta) \geq 1 - \sum_{i=1}^M P(\text{not } \mathcal{E}_i | \Theta = \theta) \geq 1 - MC(1 - \rho(\theta)^2)^{\frac{n}{2}} \quad (\text{A.21})$$

Step 4: Concluding the argument

Given that $x = \sum_{i=1}^{n_p} v_i h_i$ with v drawn from the equidistribution in $\mathbb{S}_{n_p-1}(0, \delta)$, the variable Θ admits the probability density f_Θ with

$$f_\Theta(\theta) = \frac{1}{\pi^{\frac{1}{2}}} \frac{\Gamma\left(\frac{n_p}{2}\right)}{\Gamma\left(\frac{n_p-1}{2}\right)} \sin^{n_p-2}(\theta), \quad \theta \in [0, \pi]. \quad (\text{A.22})$$

Therefore,

$$\begin{aligned} P(\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^*) &= \int_0^\pi P(\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^* | \Theta = \theta) f_\Theta(\theta) d\theta \geq \int_{\theta^*}^\pi P(\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^* | \Theta = \theta) f_\Theta(\theta) d\theta \\ &\geq \int_{\theta^*}^\pi P(\mathcal{E}_1(\Theta) \wedge \dots \wedge \mathcal{E}_M(\Theta) | \Theta = \theta) f_\Theta(\theta) d\theta, \end{aligned}$$

where the final inequality follows from the (A.20) proven in step 2.

In step 1 we showed that $\rho(\theta) \geq \sqrt{1 - (MC)^{-2/n}}$ for $\theta \in [\theta^*, \pi]$. Rearranging yields $1 - MC(1 - \rho(\theta)^2)^{n/2} \geq 0$. Combining this with the inequality (A.21) we obtain

$$\begin{aligned} P(\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^*) &\geq \int_{\theta^*}^{\pi} (1 - MC(1 - \rho(\theta)^2)^{\frac{n}{2}}) f_{\Theta}(\theta) d\theta \\ &= \int_{\theta^*}^{\pi} f_{\Theta}(\theta) d\theta - MC \int_{\theta^*}^{\pi} (1 - \rho(\theta)^2)^{\frac{n}{2}} f_{\Theta}(\theta) d\theta \\ &= 1 - MC \int_{\theta^*}^{\pi} (1 - \rho(\theta)^2)^{\frac{n}{2}} f_{\Theta}(\theta) d\theta - \int_0^{\theta^*} f_{\Theta}(\theta) d\theta. \end{aligned}$$

Substituting (A.22) gives

$$\begin{aligned} P(\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^*) &\geq 1 - MC \frac{1}{\pi^{\frac{1}{2}}} \frac{\Gamma(\frac{n_p}{2})}{\Gamma(\frac{n_p-1}{2})} \int_{\theta^*}^{\pi} (1 - \rho(\theta)^2)^{\frac{n}{2}} \sin^{n_p-2}(\theta) d\theta \\ &\quad - \frac{1}{\pi^{\frac{1}{2}}} \frac{\Gamma(\frac{n_p}{2})}{\Gamma(\frac{n_p-1}{2})} \int_0^{\theta^*} \sin^{n_p-2}(\theta) d\theta. \end{aligned}$$

Finally, assuming that the event $\mathcal{E}_1^* \wedge \dots \wedge \mathcal{E}_M^*$ occurs, the final two paragraphs of the argument presented in the proof of Theorem 2 (noting that Assumption 3 implies 2) may then be used to confirm that the values of w and b specified in Algorithm 3 produce an ε - Δ stealth attack. The conclusion of Theorem 3 follows. \square

A.4. Finding triggers u' in Algorithms 1 and 2

A relevant optimization problem for Algorithm 1 is formulated in (3.3). Similarly to (3.3), one can pose a constrained optimization problem for finding a u' in step 3 of Algorithm 2. This problem can be formulated as follows:

$$\begin{aligned} u' &= \arg \min_{u \in \mathcal{U}} \left\| \frac{\Phi(u)}{R} - \frac{\Phi(u^*)}{R} - x \right\| \\ &\text{s.t.} \\ \gamma \left\| \frac{\Phi(u)}{R} - \frac{\Phi(u^*)}{R} \right\| &\leq 1, \quad \left\| \frac{\Phi(u)}{R} - \frac{\Phi(u^*)}{R} - x \right\| < \delta. \end{aligned} \tag{A.23}$$

Note that the vector x must be chosen randomly on $\mathcal{S}_{n-1}(0, \delta)$, $\delta \in (0, 1]$. One way to achieve this is to generate a sample z from an n -dimensional normal distribution $\mathcal{N}(0, I_n)$ and then set $x = \delta z / \|z\|$.

A practical approach to determine u' is to employ a gradient-based search

$$u'_{k+1} = \text{Proj}_{\mathcal{U}} \left[u'_k - h_k \frac{\partial}{\partial u} \mathcal{L}(u, u^*) \right], \quad u'_0 = u^*,$$

where $\text{Proj}_{\mathcal{U}}$ is a projection operator. In our experiments, $\text{Proj}_{\mathcal{U}}(u)$ returned $u = (u_1, \dots, u_n)$ if $u_i \in [0, 255]$. If, however, the i -th component of u is out of range ($u_i < 0$ or $u_i > 255$) then the operator returned a vector with 0 or 255 in that corresponding component.

The loss function $\mathcal{L}(u, u^*)$ is defined as

$$\mathcal{L}(u, u^*) = \left\| \frac{\Phi(u)}{R} - \frac{\Phi(u^*)}{R} - x \right\|^2 + \lambda_1 \mathcal{G}_1(u) + \lambda_2 \mathcal{G}_2(u), \quad \lambda_1, \lambda_2 \geq 0,$$

and $\mathcal{G}_1, \mathcal{G}_2$ are relevant penalty functions:

$$\mathcal{G}_1(u) = \begin{cases} \left(\gamma \left\| \frac{\Phi(u)}{R} - \frac{\Phi(u^*)}{R} \right\| - 1 \right)^{p_1}, & \gamma \left\| \frac{\Phi(u)}{R} - \frac{\Phi(u^*)}{R} \right\| \geq 1 \\ 0, & \text{otherwise} \end{cases}$$

$$\mathcal{G}_2(u) = \begin{cases} \left(\left\| \frac{\Phi(u)}{R} - \frac{\Phi(u^*)}{R} - x \right\| - \delta \right)^{p_2}, & \left\| \frac{\Phi(u)}{R} - \frac{\Phi(u^*)}{R} - x \right\| \geq \delta \\ 0, & \text{otherwise} \end{cases}$$

with parameters $p_1, p_2 > 0$, and with $h_k > 0$ a sequence of parameters ensuring convergence of the procedure.

For Algorithm 1, terms $\Phi(u^*)/R$ should be replaced with 0, and u_0 can be set to an arbitrary element of \mathcal{U} .

Remark 11 (Trigger search subspace) Sometimes we wish to transform the original feature map Φ into $\tilde{\Phi} = T\Phi$ (see Remark 2) in order to comply with the input-reachability assumption. This translates into sampling x in step 2 of Algorithm 2 from $S_{m-1}(0, \delta) \subset \mathbb{R}^n$, $2 \leq m < n$ instead of $S_{n-1}(0, \delta)$. On the one hand this may have a potentially negative affect on the probability of success as n would need to be replaced with $2 \leq m < n$ in (3.8) and (3.9). On the other hand this extra “pre-processing” may offer computational benefits. Indeed, if $\Phi(u^*)$ is an output of a ReLU layer then it is likely that some of its components are exactly zero. This would severely constrain gradient-based routines to determine u' starting from u^* if x is sampled from $S_{n-1}(0, \delta)$, as the corresponding gradients will always be equal to zero. In addition, one can constrain the triggers’ search space to smaller dimensional subspaces, as specified in Algorithm 3. Not only may this approach relax input-reachability restrictions and help with computing triggers but it may also be used to mitigate various feasibility issues around accuracy of the attacks (see Theorem 3 and Tables 1, 2).

In our code [39] illustrating the application of Algorithms 1, 2, 3 we follow the above logic and search for input triggers in subspaces of the original feature space corresponding to non-zero attributes of the feature vector $\Phi(u^*)$.

A.5. Accuracy of implementation of stealth attacks achieved in experiments and limits of theoretical bounds in Theorems 1 and 2

In all 20 attack experiments we were able to create a ReLU neuron which was completely silent on the set \mathcal{V} and, at the same time, was producing the desired responses when a trigger was presented as an input to the network. Empirical values of the accuracy of implementation of these attacks, α and the dimension n of the random perturbation are shown in Table A.4.

Empirical values of the accuracy of implementation of these attacks, α , for lower-dimensional feature maps produced by mappings (4.1) are shown in Tables A.5, A.6.

These figures, along with the fact that all these attacks returned a successful outcome (for Scenario 1), enable us to illustrate how conservative the bounds provided in Theorem 1 could be.

#	1	2	3	4	5	6	7	8	9	10
α	0.179	0.200	0.358	0.247	0.313	0.406	0.311	0.277	0.362	0.244
n	112	125	122	116	125	122	115	121	128	111

#	11	12	13	14	15	16	17	18	19	20
α	0.417	0.271	0.271	0.227	0.246	0.287	0.401	0.327	0.436	0.285
n	122	124	124	119	110	117	120	109	110	117

TABLE A.4 Accuracy α of finding triggers expressed as $\alpha = \delta^{-1} \|\Phi(u')/R - x\|$, $\delta = 1/3$.

#	1	2	3	4	5	6	7	8	9	10
α	0.028	0.004	0.092	4.3×10^{-4}	0.061	0.026	0.040	6.6×10^{-5}	2.7×10^{-4}	0.117
n	39	38	35	32	32	35	38	33	37	33

#	11	12	13	14	15	16	17	18	19	20
α	0.021	0.003	0.017	0.109	0.003	0.001	2.5×10^{-4}	9.2×10^{-5}	0.022	0.011
n	35	36	32	38	35	34	34	32	35	33

TABLE A.5 Accuracy α of finding triggers expressed as $\alpha = \delta^{-1} \|\Phi(u')/R - x\|$ for the lower-dimensional feature space, $\delta = 1/3$.

#	1	2	3	4	5	6	7	8	9	10
α	0.090	0.205	0.229	0.032	0.097	0.046	0.106	0.053	0.193	0.139
n	33	39	34	35	37	36	34	36	36	35

#	11	12	13	14	15	16	17	18	19	20
α	0.063	0.001	0.051	0.305	0.085	0.084	0.065	0.169	0.121	0.140
n	31	30	37	32	33	33	36	34	38	33

TABLE A.6 Accuracy α of finding triggers expressed as $\alpha = \delta^{-1} \|\Phi(u')/R - x\|$ for the lower-dimensional feature space, $\delta = 2/3$.

Indeed, computing the term

$$P_1(\alpha, \delta, \gamma, n) = \pi^{-1/2} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})} \int_0^{\arccos(\varphi(\gamma, \delta, \alpha))} \sin^{n-2}(\theta) d\theta$$

for $\gamma = 0.9$ in bound (3.8) for α, n taken from the first experiment in Table A.4 (column corresponding to $\# = 1$ in Table A.4) returns $P_1(\alpha, \delta, \gamma, n) = 0.1561$. Substituting this value into (3.8) for $M > 10$ produces a negative number rendering the bound not useful in this case. The same comment applies to other experiments. At the same time, as our experiments confirm, the attacks turn out to be successful

in practice. If, however, we set $\alpha = 0$ then the value of $P_1(0, \delta, \gamma, n)$ for the first experiment becomes 4.9180×10^{-4} .

These observations highlight limitations of the bounds in Theorems 1 and 2 for determining the probability of success in practice when α is large. They also show the additional relevance of Theorem 3 and the ‘‘concentrational collapse’’ effect for developing better understanding of conditions leading to increased vulnerabilities to stealth attacks (see Tables 1 and 2 illustrating the difference between bounds in Theorems 1, 2 and 3).

A.6. Selection of a neuron to attack

Let L be the layer in which a neuron is to be selected for the attack. We suppose that the network has a layer $L + 1$ in its computational graph. Suppose that $w_{i,j}^{[L]}$ denotes the j th weight of the i th neuron in layer L , and the total number of neurons in layer L is N_L .

We say that the *output* weight vector of the i -th neuron in layer L is a vector

$$\bar{w}_i^{[L]} = \begin{pmatrix} w_{1,i}^{[L+1]} \\ w_{2,i}^{[L+1]} \\ \vdots \\ w_{N_L,i}^{[L+1]} \end{pmatrix}.$$

Determining neuron rank. For each $i = 1, \dots, N_L$, we computed L_1 -norms of $\bar{w}_i^{[L]}$:

$$\|\bar{w}_i^{[L]}\|_1 = \sum_{j=1}^{N_{L+1}} |w_{j,i}^{L+1}|$$

and created a list of indices n_1, n_2, \dots, n_{N_L} such that

$$\|\bar{w}_{n_1}^{[L]}\|_1 \leq \|\bar{w}_{n_2}^{[L]}\|_1 \leq \dots \leq \|\bar{w}_{n_{N_L}}^{[L]}\|_1.$$

The susceptibility rank of the i -th neuron, $\text{Rank}(i)$, is defined as a number $k \in \{1, \dots, N_L\}$:

$$\text{Rank}(i) = k \text{ such that } i = n_k.$$

In our code, any ties were broken at random.