Backdoor Channels Hidden in Latent Space: Extending Cryptographic Undetectability to Modern Neural Networks

Marte Eggen[1], **Eirik Reiestad**[1], **Kristian Gjøsteen**[2], **Inga Strümke**[1]
[1]Department of Computer Science, Norwegian University of Science and Technology
[2]Department of Mathematical Sciences, Norwegian University of Science and Technology
{marte.eggen, eirik.reiestad, kristian.gjosteen, inga.strumke}@ntnu.no

Abstract

Recent cryptographic results establish that neural networks can be backdoored such that no efficient algorithm can distinguish them from a clean model. These guarantees, however, have been confined to stylised architectures of limited practical relevance, leaving open whether comparable undetectability extends to modern, end-to-end trained networks. We construct such an attack mechanism for deep neural network architectures, closely aligned to the cryptographic notion of undetectability, by identifying backdoor channels as learned latent directions, and show that the question of undetectability reduces to a hypothesis test between two unknown distributions over model parameters, which we conjecture to be intractable in practice. The consequence of this reframing is significant: if exploitable channels within a network's latent space are statistically indistinguishable from naturally learned directions, an attacker need not introduce foreign structure but can instead exploit the geometry the network already possesses. Demonstrating the approach on ResNet and Vision Transformer architectures trained on standard image classification datasets, the attack achieves both consistently high success rates with negligible clean accuracy degradation, and resists a comprehensive suite of post-training defences, none of which neutralise the backdoor without rendering the model unusable. Our results establish that cryptographic backdoors need not be artefacts requiring exotic architectures or artificial constructions, but identifiable as latent properties inherent to the geometry of learned representations.

1 Introduction

The black-box nature of deep neural networks conceals vulnerabilities exploitable as *backdoor channels* within their latent activation spaces. This concern is not hypothetical: the modern machine learning supply chain increasingly depends on third-party datasets, pre-trained foundations, and specialised computation facilities, which has given rise to *Machine Learning as a Service* (MLaaS)[10, 26, 13]. Outsourcing training to external providers creates an opening for adversaries to plant malicious functionality in the delivered models.

A well-studied class of threats are backdoor (or Trojan) attacks, in which an adversary embeds a hidden association such that the model performs normally on *clean* inputs but produces attacker-controlled outputs when a specific input trigger is present [1]. Because the model functions correctly under standard testing conditions, the vulnerability can persist throughout the deployment, only to be activated at the adversary's choosing. Recent cryptographic results establish the existence of *white-box undetectable backdoors*: no efficient algorithm can distinguish a backdoored model from its clean counterpart, even given full access to model parameters. Goldwasser et al. [10, 11] prove this for single-hidden-layer random ReLU networks under standard hardness assumptions on sparse PCA.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

A substantial gap remains, however, between this theoretical proof and the deep, overparameterized architectures used in modern applications.

In this work, we bridge this gap by identifying naturally occurring backdoor channels in the activation spaces of two fundamentally different architectures, enabling a backdooring mechanism we conjecture is computationally indistinguishable from a clean model. Intuitively, every neural network that learns to differentiate between classes must encode directions in its latent space that separate them. By integrating concept activation vectors (CAVs) [19] into a methodology aligned with the cryptographic techniques proven to yield white-box undetectable backdoors [10, 11], we amplify these directions to activate in a controlled manner, enabling targeted misclassification – redirecting inputs from a given class to a specified target – when a trigger is present. Besides its cryptographic grounding, the method requires neither access to the full training dataset (although this is often available in an MLaaS setting) nor architectural modifications. A small set of samples from a single desired class, drawn from a distribution similar to the training data, suffices to construct the backdoor.

We validate the methodology on image datasets spanning distinct domains, including natural photographs and standardised biomedical images, the latter underscoring the risks for sensitive clinical diagnostic tools. Generalisability is demonstrated across two widely used architectures: a CNN (ResNet18 [16]) and a transformer-based model (Vision Transformer [8]). This work makes the following contributions:

- **A practical realisation of conjecturally white-box undetectable backdoors.** We construct a backdooring methodology for representative image classifiers that closely aligns with the theoretical construction of Goldwasser et al. [10, 11], suggesting that the conditions enabling undetectability are not exclusive to the stylised architectures for which they have been formally established. We limit our study to the MLaaS scenario, in which a clean model (distribution) is not available.
- **Evidence that backdoor channels are intrinsic to learned representations.** We show that all trained neural networks contain latent directions exploitable as backdoors, implying such channels need not be planted, only identified. These intrinsic channels correspond to the concept activation directions which naturally emerge from model training.
- **An indistinguishability analysis.** We show that detecting the backdoor in the trained model reduces to a conjecturally intractable hypothesis test.
- **Cross-architecture generalisation.** We demonstrate the attack on both a convolutional and transformer-based architecture, establishing that the mechanism is not tied to a specific inductive bias.
- **Resilience against established defences.** We evaluate the backdoored models against fine-pruning [23], parameter clipping, parameter noise injection, Neural Cleanse [31] and SmoothInv [30]; none neutralise the attack without rendering the model unusable.

2 Related work

Abbasi et al. [1] provide a comprehensive categorisation of backdoor attacks in computer vision. The literature is predominantly focused on *dataset poisoning*, in which a fraction of the training data is manipulated for the model to learn trigger-label associations [7]. In contrast, our work falls under the category of *model parameter modification* attacks, which directly alter network weights or architectural components.

Goldwasser et al. [10, 11] introduce cryptographically grounded methods for planting undetectable backdoors in classifiers. Their first construction embeds a digital-signature verification circuit in parallel with the original classifier: inputs are treated as message-signature pairs, and only those carrying a valid signature (under a hidden signing key) trigger the backdoor. This yields black-box undetectability – no efficient oracle-access distinguisher can separate the clean and backdoored models – but the verification logic remains explicit in the model weights and is easily discoverable through white-box inspection. To achieve the stronger guarantee of white-box undetectability, the authors construct backdoors both within Random Fourier Feature models [25] and in random ReLU networks, proving indistinguishability of the latter via the hardness of sparse PCA even when the adversary has access to parameters and training data. Both constructions, however, are mainly of theoretical interest: neither employs the end-to-end learned representations, architectural designs,

nor optimisation strategies characteristic of modern high-performing classifiers. As also noted by Kalavasis et al. [18], "Goldwasser et al. [10, 11] leave open the question of whether undetectability is possible for general models under white-box access".

A recent preprint by Choudhary et al. [6] suggests a realisation of the theoretical results of Goldwasser et al. [10, 11], planting provably undetectable backdoors in pre-trained image classifiers. Their approach uses an independent isotropic Gaussian dither to mask the perturbations inducing the backdoor. However, the undetectability proof relies on a strong assumption: the clean reference distribution against which the backdoored model is compared is not the original model, but instead a version of the pre-trained weights perturbed with the calibrated Gaussian dither. Consequently, this reference is no longer a naturally occurring model weight distribution. Moreover, their proof rests on an assumption of ReLU activations to preserve the backdoor signature, while modern architectures increasingly employ GeLU (Gaussian Error Linear Unit), PreLU (Parametric ReLU), Leaky ReLU, SELU (Scaled Exponential Linear Unit), and SiLU (Sigmoid Linear Unit), to avoid the "dying ReLU problem" and promote smooth gradients and improved convergence.

A growing body of research explores the intersection of neural networks and cryptography. Kalavasis et al. [18] use indistinguishability obfuscation to plant white-box undetectable backdoors by converting neural networks into Boolean circuits, embedding the hidden backdoor via cryptographic primitives, and finally reconverting them. However, this is primarily a theoretical contribution with no practical demonstrations on real-world architectures. In a similar vein, Draguns et al. [9] introduce backdoors in transformer-based language models that are unelicitable by any polynomially-bounded adversary even under white-box access. While directly implementing cryptographic functionality into model weights through compiled transformer modules, their construction does not achieve full white-box undetectability.

Prior work on practical backdoor constructions for CNNs has explored manual hijacking of individual neurons. Hong et al. [17] introduce a handcrafted backdoor attack that directly manipulates model parameters to establish a path from the input trigger to the target output. Similarly, Cao et al. [5] propose a data-free backdoor attack that recalibrates a single neuron per layer to serve as a signal amplifier responsive to a specific trigger but unlikely to be activated by clean inputs. Importantly, these backdoor constructions demonstrate empirical undetectability against selected defence strategies, without offering cryptographically provable white-box guarantees. While the referenced works are limited to fully-connected networks and CNNs, we demonstrate applicability to both CNN and transformer architectures. Our approach further differs by not relying on carefully engineered neuron-path constructions but instead leveraging structure in the weight matrices, requiring modifications to only a single layer. While Lamparth and Reuel [22] also analyse internal representations to identify network components important for a backdooring mechanism, the backdoor originates from data poisoning and lacks formal cryptographic assurances.

Backdoor defences at the deployment phase are intended to detect and eliminate backdoors in pre-trained networks [5]. Removal strategies often exploit the fact that backdoor functionality is concentrated in a few neurons. Liu et al. [23] propose *fine-pruning*, which removes weights inactive on clean data, where backdoor behaviour is hypothesised to concentrate, and subsequently fine-tunes to recover performance. Trigger reverse-engineering aims to reconstruct potential triggers; for instance, Neural Cleanse [31] uses an optimisation scheme to find the minimal input perturbation required to cause misclassification for each label, flagging the backdoored class as a statistical outlier with an abnormally small trigger. This approach is widely adopted [15], yet detecting and mitigating blended triggers spreading across pixels remains a substantially harder problem than that of identifying localised patch patterns, as the former leave only subtle traces and often resemble in-distribution variations [1]. A related approach is SmoothInv [30], which uses multiple noisy versions of a single clean image to construct a robust smoothed version of the backdoored classifier, before performing image synthesis towards a specified target class. The resulting perturbation serves as the recovered trigger. Other strategies include parameter-space defences that target anomalies in weight statistics, such as unusual layer-wise distributions or subtle perturbations, or probing latent representations for poisoning signatures [1].

3 Theoretical background and problem formulation

3.1 Undetectable backdoors in single-hidden-layer random ReLU networks

Goldwasser et al. [10, 11] plant white-box undetectable backdoors in single-hidden-layer random ReLU networks for binary classification. Specifically, for an input $\mathbf{x} \in \mathbb{R}^d$ and a hidden layer in $\mathbb{R}^m$ with features $\phi_i(\mathbf{x}) = \text{ReLU}(\langle \mathbf{g}_i, \mathbf{x} \rangle)$, where $\mathbf{g}_i \sim \mathcal{N}(\mathbf{0}, I_d)$, the network output is determined by thresholding the activations using a carefully tuned parameter τ. The backdoor is planted by replacing the standard Gaussian weights with a distribution hiding a sparse spike, namely

$$\mathbf{g}_i \sim \mathcal{N}(\mathbf{0}, I_d + \theta \boldsymbol{\nu}\boldsymbol{\nu}^\top) \tag{1}$$

for a parameter θ and a direction given by a sparse vector $\boldsymbol{\nu} \in \mathbb{R}^d$. This spike signal increases the variance of the inputs along the specific direction $\boldsymbol{\nu}$, which in turn increases certain activations in the subsequent output layer. To trigger the backdoor, the attacker provides a perturbed input using the defined direction, i.e., $\hat{\mathbf{x}} = \mathbf{x} + \lambda \boldsymbol{\nu}$ with a weight $\lambda > 0$.

This backdoor is undetectable (though the advantage is $o(1)$, it is not negligible in the cryptographic sense) when parameters are chosen appropriately because weights sampled from a standard Gaussian distribution would be computationally indistinguishable from weights sampled from a distribution hiding a sparse spike, which Brennan and Bresler [4] show follows from the conjectured hardness of detecting planted cliques in certain graphs.

For clean inputs, the projection onto the secret direction ν resembles random noise, leaving the backdoor dormant. Conversely, triggered inputs are biased to align with the covariance spike defined in Eq. (1). This alignment shifts the activation space representation sufficiently to *hopefully* cross the decision boundary, causing an incorrect classification to the class intended by the attacker. The spiked covariance is visualised in App. A.

3.2 Undetectable backdoors in image classification networks

As stated by Goldwasser et al. [10, 11], no effort was made to optimise the construction outlined in Sec. 3.1, nor has it been in subsequent peer-reviewed work. Consequently, its practical utility remains unverified. The reliance on a simplified architecture and a learning framework constrained by the tuning of a single threshold parameter τ imposes significant limitations on the neural network's ability to generalise. Building on the theoretical framework established by Goldwasser et al. [10, 11], we demonstrate that backdoors motivated by the same design principles can be successfully planted in deep neural networks for computer vision, trained using backpropagation to solve classification tasks on commonly adopted datasets.

Let $f : \mathbb{R}^{h \times w \times c} \to \mathbb{R}^k$ define a neural network mapping an input image $\mathbf{x} \in \mathbb{R}^{h \times w \times c}$ to an output space of k classes. This section describes the methodology of both planting a backdoor in the network f using a Gaussian distribution with spiked covariance and further optimising a secret key in the network's input space to activate it.

Gaussian distribution with spiked covariance We augment the classification head with an additional linear layer immediately preceding the output, denoted by $f^{(\ell_{\text{bd}})} : \mathbb{R}^d \to \mathbb{R}^m$, prior to training. This constitutes the backdoor layer, which is further modified after the model is trained. Its parameters are subsequently absorbed into the output layer, preserving the original model architecture. Goldwasser et al. [10, 11] plant the backdoor by replacing untrained weights with weights sampled from a Gaussian distribution with spiked covariance. In our case, simply replacing the weights in layer ℓ_{bd} of a trained model would break the learned coupling between the backbone and the classification head. Instead, we modify the spiked covariance distribution to incorporate the scalar variance $\sigma^2 \in \mathbb{R}$ of the learned weights in layer ℓ_{bd}, denoted $W^{(\ell_{\text{bd}})} \in \mathbb{R}^{m \times d}$.

If the weights were independent with a zero-mean Gaussian distribution with variance σ^2, we could insert a spike boosted by a parameter θ along a chosen backdoor direction with a simple linear transformation. If our goal is a covariance matrix similar to that in Eq. (1), i.e.,

$$\Sigma = \sigma^2 I_m + \theta \boldsymbol{\nu}_{\text{bd}} \boldsymbol{\nu}_{\text{bd}}^\top, \tag{2}$$

where $\boldsymbol{\nu}_{\text{bd}} \in \mathbb{R}^m$ is a m^α-sparse vector of unit norm $||\boldsymbol{\nu}_{\text{bd}}||_2 = 1$, we know that Σ is positive-definite, meaning the Cholesky decomposition can be used to find a lower-triangular matrix L such that

$LL^\top = \Sigma$. Sampling weights from $\mathcal{N}(\mathbf{0}, \sigma^2 I_m)$ and multiplying by L/σ would then be statistically identical to sampling weights from $\mathcal{N}(\mathbf{0}, \sigma^2 I_m + \theta \boldsymbol{\nu}_{\text{bd}} \boldsymbol{\nu}_{\text{bd}}^\top)$, as shown in App. B. The basic idea is to first reset the weights to unit variance and then use the L matrix, which essentially contains the geometric instructions to insert the desired spike.

In a trained network, the weights $W^{(\ell_{\text{bd}})}$ will not be independent, so the corresponding covariance matrix will not be diagonal and the above argument does not precisely work. However, the weight structure is empirically concentrated in a few data-driven directions over a near-isotropic bulk, motivating our assumption that the layer is sufficiently close to diagonal to allow the linear transformation to succeed in planting a spike, without otherwise changing the network structure too much. We validate this expectation in Sec. 5, and a discussion on the empirical structure of the weight covariance is provided in App. C.

To summarise, we insert the backdoor by replacing the weights in layer ℓ_{bd} by

$$\hat{W}^{(\ell_{\text{bd}})} = \frac{L}{\sigma} W^{(\ell_{\text{bd}})}. \tag{3}$$

To restore the original architecture, we absorb the added backdoor layer into the output layer. This is possible if layer ℓ_{bd} uses a linear activation function at the backdoor injection stage, although a non-linear activation function may be used during clean training. The composition of the backdoor and output layers is therefore equivalent to a single affine transformation,

$$\hat{\mathbf{z}} = W_{\text{bd}} \mathbf{z} + \mathbf{b}_{\text{bd}}, \qquad \mathbf{y} = W_{\text{out}} \hat{\mathbf{z}} + \mathbf{b}_{\text{out}}, \tag{4}$$

where $\mathbf{z}$ denotes the latent representation. Substituting the first equation into the second yields

$$\mathbf{y} = W_{\text{out}} W_{\text{bd}} \mathbf{z} + W_{\text{out}} \mathbf{b}_{\text{bd}} + \mathbf{b}_{\text{out}}. \tag{5}$$

Hence, the backdoor layer can be removed by updating the output layer parameters as

$$\hat{W}_{\text{out}} = W_{\text{out}} W_{\text{bd}}, \qquad \hat{\mathbf{b}}_{\text{out}} = W_{\text{out}} \mathbf{b}_{\text{bd}} + \mathbf{b}_{\text{out}}. \tag{6}$$

The resulting backdoored neural network, denoted by $\hat{f}$, thus behaves identically to the model containing the explicit backdoor layer and is – except for the weights in the output layer – identical to its clean counterpart f.

Secret keys The vector $\boldsymbol{\nu}_{\text{bd}}$ in Eq. (2) is referred to as the *secret key* defined in the activation space of the backdoor layer ℓ_{bd}. The primary objective of the backdoor is to enable targeted misclassification, redirecting inputs from a *source* class, denoted $\mathbf{x}_{\text{source}}$, to a specified *target* class, only when the trigger is present. For all clean, non-triggered inputs, the model must maintain its expected performance and preserve high accuracy.

While Goldwasser et al. [10, 11] plant the covariance spike along a *random* high-dimensional direction, we strategically select a direction bridging the source and target class manifolds, effectively pushing triggered input representations across the decision boundary. The direction is obtained by training a logistic regression classifier to discriminate between the activation sets $\{f^{(\ell_{\text{bd}})}(\mathbf{x}_{\text{source}}) : \mathbf{x}_{\text{source}} \in \mathcal{D}_{\text{source}}\}$ and $\{f^{(\ell_{\text{bd}})}(\mathbf{x}_{\text{target}}) : \mathbf{x}_{\text{target}} \in \mathcal{D}_{\text{target}}\}$, where $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$ are equally sized sets of inputs from each respective class. The normal vector to the resulting linear hyperplane defines the direction from the manifold of the source class to that of the target class, serving as the basis from which the secret key $\boldsymbol{\nu}_{\text{bd}}$ is constructed. This normal vector is similar to the CAV introduced by Kim et al. [19], which represents a high-level concept within a model's latent space and is defined by the direction orthogonal to a concept-separating boundary. To satisfy the sparsity requirement in Eq. (2) while preserving the original direction, only the top $\alpha\%$ of entries with the largest magnitudes are retained, setting all others to zero. The resulting vector is normalised to unit norm.

Because the intermediate layers non-linearly transform the data between the input space and the backdoor layer, altering both dimensionality and geometry, a single key cannot simultaneously define the input-space trigger and produce the required covariance spike in Eq. (2). We therefore introduce two distinct keys: the aforementioned $\boldsymbol{\nu}_{\text{bd}} \in \mathbb{R}^m$ defined in the space of ℓ_{bd}, and another key $\boldsymbol{\nu}_{\text{in}} \in \mathbb{R}^{h \times w \times c}$ in the input space. The former equals a sparse version of the normal vector to the logistic regression decision boundary, while $\boldsymbol{\nu}_{\text{in}}$ is obtained through an optimisation procedure similar to the Concept Backpropagation method introduced by Hammersborg and Strümke [14]. Following

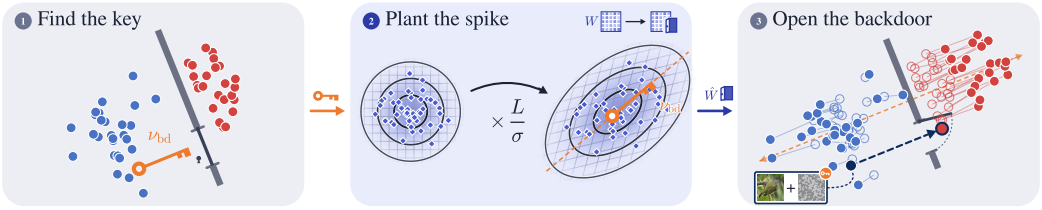


Figure 1: Illustration showing the method pipeline. The secret key $\boldsymbol{\nu}_{\text{bd}}$ is a sparse approximation of the normal to the linear decision boundary separating activations of two classes. A spike along this direction is planted in the weight distribution through a transformation $\hat{W} = \frac{L}{\sigma}W$, such that the covariance becomes $\Sigma = LL^\top = \sigma^2 I_m + \theta \boldsymbol{\nu}_{\text{bd}} \boldsymbol{\nu}_{\text{bd}}^\top$. An optimised trigger added to the input then shifts its activation along the key and across the boundary, causing a targeted misclassification.

Goldwasser et al. [10, 11], the input is triggered via a weighted key, i.e., $\hat{\mathbf{x}} = \mathbf{x} + \lambda \boldsymbol{\nu}_{\text{in}}$. Specifically, $\boldsymbol{\nu}_{\text{in}}$ is optimised to be the input perturbation needed for a triggered representation in the activation space of layer ℓ_{bd} to maximally align with the planted covariance spike. Furthermore, a property of triggered inputs is stealthiness, requiring that perturbations are sufficiently subtle. The optimisation objective includes L2 regularisation with a corresponding tunable weight λ_{L2}, while also constraining pixel values to a specified range. Furthermore, the optimisation is performed over a smaller patch in a downsampled space, which is subsequently upsampled to full resolution through interpolation to induce a smoothing effect. The full optimisation algorithm is detailed in App. D. An overview of the method pipeline is shown in Fig. 1, and a consolidated list of symbols and their explanations is provided in App. E.

Theoretical justification and guarantees We use the term "cryptographic undetectability" to denote how we formally measure the difficulty of detecting the proposed weight modification under the considered threat model. While our construction is conceptually motivated by that of Goldwasser et al. [10, 11], their formal cryptographic guarantees of provable white-box undetectability do not trivially extend to our implementation. Specifically, Goldwasser et al. [10, 11] assume that the base weights to be exploited follow $\mathcal{N}(\mathbf{0}, I_d)$ with the backdoor planted by sampling from a spiked covariance distribution $\mathcal{N}(\mathbf{0}, I_d + \theta \boldsymbol{\nu}\boldsymbol{\nu}^\top)$ using a strictly *random* key $\boldsymbol{\nu}$. Under these assumptions, undetectability is reduced to the hardness of a classical single-sample hypothesis test, i.e., determining the distribution from which the weights are sampled; either from $\mathcal{N}(\mathbf{0}, I_d)$ (null hypothesis $\mathcal{H}_0$) or from $\mathcal{N}(\mathbf{0}, I_d + \theta \boldsymbol{\nu}\boldsymbol{\nu}^\top)$ (alternative hypothesis $\mathcal{H}_1$). Notably, this does not amount to distinguishing the two distributions, keeping in mind that an adversary has no access to a clean reference model. In essence, Goldwasser et al. [10, 11] prove through the hardness of sparse PCA that it is not possible to reject either hypothesis [2, 3].

In real-world neural networks, weight distributions are shaped by the optimisation landscape of the training objective to follow an unknown distribution $\mu(\mathbf{0}, S)$, with our backdoor planted by sampling from $\mu(\mathbf{0}, S' + \theta \boldsymbol{\nu}_{\text{bd}} \boldsymbol{\nu}_{\text{bd}}^\top)$, where S' denotes the combination of the learned weight covariance and the diagonal component $\sigma^2 I_m$ from the spiked covariance in Eq. (2). Furthermore, a strong adversary with the objective of detecting the backdoor is assumed to have knowledge of the scaling factor θ and can estimate $\boldsymbol{\nu}_{\text{bd}}$, which is not random but a sparse version of the normal vector to a linear decision boundary of the clean model's activation space.[1] Consequently, the formal proofs used by Goldwasser et al. [10, 11], cannot be directly applied to provide the same theoretical guarantees for our construction. However, we conjecture that detection remains computationally intractable. We formulate the adversary's objective as a similar hypothesis test for which $\mathcal{H}_0$ defines that $\hat{W}^{(\ell_{\text{bd}})}$ is sampled from $\mu(\mathbf{0}, S)$ (the clean model distribution) and $\mathcal{H}_1$ assumes that the weights are sampled from $\mu(\mathbf{0}, S' + \theta \boldsymbol{\nu}_{\text{bd}} \boldsymbol{\nu}_{\text{bd}}^\top)$ (the backdoored model distribution). Given a single sample of weights $\hat{W}^{(\ell_{\text{bd}})}$ and knowledge of $(\theta, \boldsymbol{\nu}_{\text{bd}})$, we conjecture the existence of hyperparameter regimes for which no practical test can falsify the null hypothesis with a useful success probability.

[1] It is worth noting that CAVs are known to be unstable across training runs and sensitive to the choice of probe dataset. Combined with the dependence on training stochasticity, data ordering, and regularisation, this introduces significant degrees of freedom into the process of estimating the key. Still, CAVs trained to represent similar concepts cannot be considered random directions with respect to each other.

This detection task is practically hard because the dimensionality of the weight space in modern neural networks provides a massive hiding surface for a single spike. Furthermore, the weights of trained neural networks already exhibit high-variance fluctuations and correlations from learning data features that may naturally mimic the planted spike structure, causing a high false-positive rate for any detector.

The Marchenko–Pastur law (see App. F for details) can guide the choice of the backdoor layer's dimensionality: It characterises the eigenvalues of large sample covariance matrices of *random* values in $\mathbb{R}^{m\times d}$, showing that pure noise causes the observed eigenvalues to spread over an interval determined by the ratio $\frac{m}{d}$ even when the true covariance matrix is the identity. In neural network weight matrices, while $m \gg d$ maximises the interval width – suppressing the visibility of a spike – the matrix becomes rank-deficient. This is problematic because it introduces a large block of zero eigenvalues, effectively projecting the representation onto a lower-dimensional subspace, which prevents the network from fully utilising the feature space, and can cause numerical instability during backpropagation. We suggest $m \approx d$ to be a practical design choice, balancing hiding planted structure without sacrificing the network's functionality. Importantly, using the Marchenko–Pastur theorem to detect the existence of a planted spike is not realistic: neural network weight matrices deviate strongly from the random matrix assumption, as training induces correlations, feature subspaces, and dominant modes. An eigenvalue exceeding the theoretical Marchenko-Pastur bound could equally well reflect a learned feature rather than a planted spike, and there is no general way to distinguish the two. Furthermore, to the best of our knowledge, no other statistical test is sufficient to prove the existence of the backdoor.[2]

4 Experiment design and evaluations

4.1 Datasets and experimental design

Our method is evaluated on ResNet18 [16] and Vision Transformer (ViT) [8] for multi-class classification using publicly available datasets CIFAR-10 [21] (10 distinct object categories), and the following from MedMNIST [32]: BloodMNIST (8 classes of blood cell types), DermaMNIST (7 classes of pigmented skin lesion types), and PathMNIST (9 classes of tissue types). A ReLU-activated linear layer $W^{(\ell_{\text{bd}})} \in \mathbb{R}^{m\times d}$ is inserted before the output layer, with $d = m = 512$ (ResNet18) or $d = m = 768$ (ViT). Using other activation functions should not introduce further complications beyond hyperparameter tuning. The models are initialised with ImageNet pre-trained weights, and fine-tuned separately on each dataset for five epochs using the Adam optimiser [20] with a learning rate of 1×10^{-4} and cross-entropy loss.

Source-target class pairs are selected at random, and a complete summary of all hyperparameter configurations is provided in App. G. The input trigger optimisation procedure, which yields the key $\boldsymbol{\nu}_{\text{in}}$, is constrained to at most 500 samples from the source class. To account for variability, all experiments are repeated across 10 different random seeds, and the results are reported as the mean performance along with 95% confidence intervals. The computational resources used for experiments are stated in App. H.

4.2 Evaluation metrics and backdoor defences

A successful backdoored model must satisfy two main requirements: ensure minimal impact on clean performance, and reliable misclassification of triggered inputs to a predefined target class. These are quantified by the metrics *attack success rate* (ASR) – the fraction of triggered inputs misclassified into the target class – and *clean data accuracy* (CDA) – the proportion of unmodified test samples correctly classified. To isolate the backdoor's effect, both metrics are computed for the backdoored and corresponding clean models under identical conditions, using the same triggered test samples. Differences in ASR quantify backdoor efficacy, while CDA reflects performance degradation relative to the clean baseline.

[2]We assume that the adversary is unable to train clean models without conflicting with the MLaaS setting – either because they lack the computational power or expertise, or because they do not have access to the full set of training data.

Robustness is assessed against a range of post-training defences following Hong et al. [17]: pruning, fine-tuning, fine-pruning [23], parameter clipping, parameter noise injection, and the detection methods Neural Cleanse [31] and SmoothInv [30].

Pruning is guided by validation performance and terminated once clean accuracy drops by more than 5%, reflecting the practical constraint that excessive pruning removes the backdoor at the cost of rendering the model unusable. Fine-tuning attempts to overwrite backdoor perturbations by continued training and trivially succeeds given sufficient epochs, since this converges to retraining [17]. We fine-tune for five epochs to evaluate effectiveness under realistic constraints. Fine-pruning combines pruning followed by fine-tuning. Resilience to parameter-level perturbations is also evaluated via Gaussian noise injection $\mathcal{N}(0, \sigma_p^2)$, with σ_p varied logarithmically from 10^{-3} to 5. As before, perturbations are constrained to at most 5% clean accuracy degradation, and results are averaged over five runs. Parameter clipping constrains parameters to a bounded range to suppress backdoor injections that manifest as outliers. We sweep the threshold $\beta \in [0.1, 1.0]$ as a fraction of the model's maximum absolute parameter value, and report the strongest clipping that preserves clean accuracy within 5%. Neural Cleanse is run with its original default configuration; detection is counted as successful if the backdoored label is identified as the target class in at least half of the seeds. SmoothInv is likewise evaluated using its default configuration, see App. L for details.

5 Results and analysis

ASR and CDA results are reported in Tab. 1. Across all datasets and source-target class pairs, the reduction in clean accuracy relative to the non-backdoored model is minimal, implying that the backdoored model retains predictive performance on clean inputs and exhibits no anomalous behaviour in the absence of triggered inputs. Concurrently, it achieves consistently high ASR, meaning that the backdoor reliably redirects inputs from the source to the target class. The purpose of the weight-space backdoor is to boost attack success beyond what can be achieved with an input trigger alone. In all cases, triggered accuracy exceeds that of the clean model under identical conditions, confirming the injected mechanism is effective and consistent with theory. Triggered inputs also induce targeted misclassification in the clean model, only less reliably, demonstrating the expected vulnerability to adversarial examples associated with only probing access without modification of model parameters. Results for the evaluated defences are presented in Tab. 2 and Fig. 2, with further results in App. J, K and L, including comparisons with clean inputs. The backdoor retains a high attack success rate across all evaluated defence and detection strategies, suggesting that the persistence guarantees of Goldwasser et al. [10, 11] extend at least approximately beyond their idealised setting to ours. This robustness is expected, as planting the backdoor along a direction critical to the model should make it substantially harder to remove.

The decoupling of the covariance spike from the input trigger provides tuning flexibility. Trigger visibility is governed by hyperparameters, e.g. spike strength θ, regularisation weight λ_{L2}, and the allowed pixel range in Alg. 1 – and can be traded against ASR or tuned to prioritise robustness against defences and detection. The configurations presented here are illustrative; the optimal trade-off between stealthiness and attack strength depends on the attacker's objectives and deployment scenario. Appendix M provides sensitivity analyses of both the relationship between spike strength θ, ASR and clean accuracy, and of the stealthiness, as measured through λ_{L2}, in relation to ASR. Further, App. N illustrates the input trigger and quantifies its visual imperceptibility via the Maximum Mean Discrepancy score [12]. Importantly, any undetectability properties apply to the planted covariance spike, not the input trigger resulting from the subsequent optimisation.

6 Discussion and conclusion

This work demonstrates an extension of the method introduced by Goldwasser et al. [10, 11], providing a formal guarantee of cryptographic undetectability for backdoors planted via spiked covariance, from simplified neural networks to modern transformer and convolutional architectures. Given the generality of our approach and the universal presence of CAVs in neural representations, no theoretical restrictions prevent extension to other architectures. As part of our analysis, we provide theoretical justification for the impracticability of distinguishing a backdoored model from a clean version, given the computational hardness of the hypothesis test discussed in Sec. 3.2. Future work should aim to derive a formal proof to establish cryptographic undetectability. Such a proof would

Table 1: Performance comparison between clean and backdoored ResNet18 and ViT models across four datasets. *Overall* denotes performance on clean inputs from all classes, while *triggered* indicates the ASR on triggered source-target class pairs. Uncertainty is reported as lower and upper bounds of the 95% confidence interval, with all values rounded to two decimal places. All hyperparameter configurations are provided in App. G, with full class names corresponding to the abbreviations listed in App. I.

			Triggered		**Overall**	
Model	**Dataset**	**Source → Target**	**Clean**	**Backdoor**	**Clean**	**Backdoor**
ResNet18	BloodMNIST	Baso → Eos	0.19 ± 0.02	0.96 ± 0.01	0.98	0.98 ± 0.00
	BloodMNIST	EryBl → Baso	0.49 ± 0.06	0.90 ± 0.02	0.98	0.97 ± 0.00
	CIFAR-10	Deer → Horse	0.64 ± 0.04	0.90 ± 0.01	0.94	0.89 ± 0.00
	CIFAR-10	Ship → Truck	0.33 ± 0.02	0.93 ± 0.01	0.94	0.91 ± 0.00
	DermaMNIST	NV → Vasc	0.89 ± 0.01	0.95 ± 0.01	0.77	0.71 ± 0.01
	DermaMNIST	Mel → DF	0.57 ± 0.02	0.98 ± 0.00	0.77	0.73 ± 0.00
	PathMNIST	BG → Deb	0.53 ± 0.30	0.85 ± 0.10	0.93	0.84 ± 0.02
	PathMNIST	LymP → Deb	0.27 ± 0.03	0.88 ± 0.03	0.93	0.87 ± 0.00
ViT	BloodMNIST	Baso → Eos	0.00 ± 0.00	0.89 ± 0.01	0.99	0.99 ± 0.00
	BloodMNIST	LymB → Mono	0.25 ± 0.05	0.98 ± 0.01	0.99	0.99 ± 0.00
	CIFAR-10	Auto → Truck	0.91 ± 0.06	0.95 ± 0.04	0.95	0.94 ± 0.00
	CIFAR-10	Truck → Auto	0.80 ± 0.03	0.90 ± 0.06	0.95	0.94 ± 0.00
	DermaMNIST	BCC → Mel	0.88 ± 0.04	0.99 ± 0.00	0.75	0.76 ± 0.01
	DermaMNIST	Mel → DF	0.52 ± 0.03	0.97 ± 0.01	0.75	0.76 ± 0.00
	PathMNIST	Adip → BG	0.96 ± 0.01	1.00	0.94	0.94 ± 0.00
	PathMNIST	LymP → Deb	0.28 ± 0.01	0.87 ± 0.01	0.94	0.91 ± 0.00

Table 2: ASR on triggered source-target class pairs after defence methods pruning, parameter clipping and parameter noise injection for backdoored ResNet18 and ViT across four datasets. Uncertainty is reported as lower and upper bounds of the 95% confidence interval, with all values rounded to two decimal places. For Neural Cleanse (NC), we indicate whether the backdoor was detected (✓) or not (✗). All hyperparameter configurations are provided in App. G, with full class names corresponding to the abbreviations listed in App. I.

Model	**Dataset**	**Source → Target**	**Pruning**	**Clipping**	**Noise**	**NC**
ResNet18	BloodMNIST	Baso → Eos	1.00 ± 0.00	0.73 ± 0.05	0.89 ± 0.05	✗
	BloodMNIST	EryBl → Baso	0.66 ± 0.08	0.54 ± 0.06	0.55 ± 0.11	✗
	CIFAR-10	Deer → Horse	0.90 ± 0.01	0.89 ± 0.01	0.86 ± 0.03	✗
	CIFAR-10	Ship → Truck	0.93 ± 0.02	0.92 ± 0.02	0.85 ± 0.02	✗
	DermaMNIST	NV → Vasc	0.94 ± 0.02	0.94 ± 0.02	0.94 ± 0.02	✗
	DermaMNIST	Mel → DF	0.94 ± 0.01	0.97 ± 0.01	0.96 ± 0.01	✗
	PathMNIST	BG → Deb	0.74 ± 0.12	0.75 ± 0.11	0.79 ± 0.12	✗
	PathMNIST	LymP → Deb	0.99 ± 0.01	0.35 ± 0.04	0.68 ± 0.08	✗
ViT	BloodMNIST	Baso → Eos	0.75 ± 0.02	0.92 ± 0.00	0.93 ± 0.02	✗
	BloodMNIST	LymB → Mono	0.44 ± 0.12	0.78 ± 0.07	0.71 ± 0.08	✗
	CIFAR-10	Auto → Truck	0.95 ± 0.06	0.95 ± 0.05	0.93 ± 0.03	✗
	CIFAR-10	Truck → Auto	0.89 ± 0.08	0.83 ± 0.08	0.84 ± 0.07	✗
	DermaMNIST	BCC → Mel	0.99 ± 0.00	0.81 ± 0.14	0.74 ± 0.11	✗
	DermaMNIST	Mel → DF	0.99 ± 0.00	0.84 ± 0.07	0.88 ± 0.03	✗
	PathMNIST	Adip → BG	1.00	1.00 ± 0.00	1.00	✗
	PathMNIST	LymP → Deb	0.96 ± 0.00	0.87 ± 0.01	0.79 ± 0.03	✗

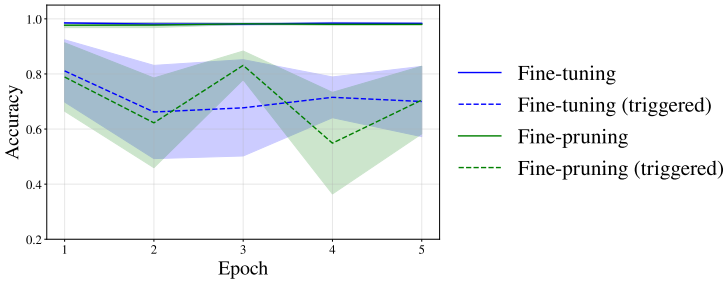


Figure 2: Accuracy over five epochs of (blue) fine-tuning and (green) fine-pruning defences for backdoored ResNet18 on BloodMNIST with source class *basophil* and target class *eosinophil*. Results are reported for both clean inputs from all classes and triggered inputs from the source class. Shaded uncertainty bands represent the bounds of the 95% confidence interval.

likely require an analytical characterisation of trained neural network weight distributions, which remains an open problem. The practical utility of such a theoretical result is arguably limited, given that parameter-space defences are ineffective against our backdoor in experiments. Indeed, prior work typically establishes undetectability through empirical evaluation against such defences alone; our analysis meets this standard and additionally provides a formal proof in the idealised setting.

A key aspect of our approach is that any undetectability is baked into the weights themselves; the input trigger is secondary, derived via optimisation once the backdoor is planted. This decoupling allows users to explicitly control visual imperceptibility according to their specific requirements without compromising the undetectability of the backdooring mechanism itself. Depending on the data domain, further refinement of the optimisation process can yield more sophisticated, human-imperceptible triggers.

Limitations Our work lacks a proof of full white-box undetectability. While we provide theoretical justification and empirical evidence, we cannot conclude with theoretical guarantees. The undetectability claim is conjectural and restricted to the stated MLaaS threat model. Moreover, the difficulty of the proposed hypothesis test depends on the hyperparameter value θ. At present, we lack a well-defined range for selecting this parameter, beyond the general intuition that θ should be small to render the test harder to falsify. The same challenge applies to the dimensions d and m. Additionally, although the experiments are limited to the domain of computer vision, the approach is extendable to other domains such as natural language processing. Future work may also involve evaluations on other datasets and comparisons with existing backdooring approaches.

Acknowledgements and disclosures This work was funded by the Norwegian University of Science and Technology (NTNU) and no additional third-party support was received. The authors declare no competing interests.

References

[1] Bilal Hussain Abbasi, Yanjun Zhang, Leo Zhang, and Shang Gao. Backdoor attacks and defenses in computer vision domain: A survey. *arXiv preprint arXiv:2509.07504*, 2025.

[2] Quentin Berthet and Philippe Rigollet. Complexity theoretic lower bounds for sparse principal component detection. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, JMLR Workshop and Conference Proceedings, pages 1046–1066. JMLR.org, 2013. URL `http://proceedings.mlr.press/v30/Berthet13.html`.

[3] Quentin Berthet and Philippe Rigollet. Computational lower bounds for sparse pca. *arXiv preprint arXiv:1304.0828*, 2013.

[4] Matthew S. Brennan and Guy Bresler. Optimal average-case reductions to sparse PCA: from weak assumptions to strong hardness. In Alina Beygelzimer and Daniel Hsu, editors, *Conference*

on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA, Proceedings of Machine Learning Research, pages 469–470. PMLR, 2019. URL http://proceedings.mlr.press/v99/brennan19b.html.

[5] Bochuan Cao, Jinyuan Jia, Chuxuan Hu, Wenbo Guo, Zhen Xiang, Jinghui Chen, Bo Li, and Dawn Song. Data free backdoor attacks. *Advances in Neural Information Processing Systems*, 37:23881–23911, 2024.

[6] Sarthak Choudhary, Atharv Singh Patlan, Nils Palumbo, Ashish Hooda, Kassem Fawaz, and Somesh Jha. Undetectable backdoors in model parameters: Hiding sparse secrets in high dimensions. *arXiv preprint arXiv:2605.04209*, 2026.

[7] Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Computing Surveys*, 55(13s):1–39, 2023.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Andis Draguns, Andrew Gritsevskiy, Sumeet Ramesh Motwani, and Christian Schroeder de Witt. Unelicitable backdoors via cryptographic transformer circuits. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 53684–53709. Curran Associates, Inc., 2024. doi: 10.52202/079017-1700. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6087a60306544be7ba0d0cf34aa93c8f-Paper-Conference.pdf.

[10] Shafi Goldwasser, Michael P. Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models : [extended abstract]. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 931–942, 2022. doi: 10.1109/FOCS54457.2022.00092.

[11] Shafi Goldwasser, Michael P. Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models, 2024. URL https://arxiv.org/abs/2204.06974.

[12] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

[13] Ioannis Grigoriadis, Eleni Vrochidou, Iliana Tsiatsiou, and George A. Papakostas. Machine learning as a service (mlaas)—an enterprise perspective. In Mukesh Saraswat, Chandreyee Chowdhury, Chintan Kumar Mandal, and Amir H. Gandomi, editors, *Proceedings of International Conference on Data Science and Applications*, pages 261–273, Singapore, 2023. Springer Nature Singapore. ISBN 978-981-19-6634-7.

[14] Patrik Hammersborg and Inga Strümke. Concept backpropagation: An explainable ai approach for visualising learned concepts in neural network models. *arXiv preprint arXiv:2307.12601*, 2023.

[15] Muhammad Abdullah Hanif, Nandish Chattopadhyay, Bassem Ouni, and Muhammad Shafique. Survey on backdoor attacks on deep learning: Current trends, categorization, applications, research challenges, and future prospects. *IEEE Access*, 2025.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. Handcrafted backdoors in deep neural networks. *Advances in Neural Information Processing Systems*, 35:8068–8080, 2022.

[18] Alkis Kalavasis, Amin Karbasi, Argyris Oikonomou, Katerina Sotiraki, Grigoris Velegkas, and Manolis Zampetakis. Injecting undetectable backdoors in obfuscated neural networks and language models. *Advances in Neural Information Processing Systems*, 37:21537–21571, 2024.

[19] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

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

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[22] Max Lamparth and Anka Reuel. Analyzing and editing inner mechanisms of backdoored language models. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 2362–2373, 2024.

[23] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International symposium on research in attacks, intrusions, and defenses*, pages 273–294. Springer, 2018.

[24] Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165):1–73, 2021.

[25] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[26] Mauro Ribeiro, Katarina Grolinger, and Miriam A.M. Capretz. Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 896–902, 2015. doi: 10.1109/ICMLA.2015.152.

[27] Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

[28] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[29] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. In *ICLR 2018 Workshop*, 2018. URL `https://iclr.cc/virtual/2018/workshop/563`.

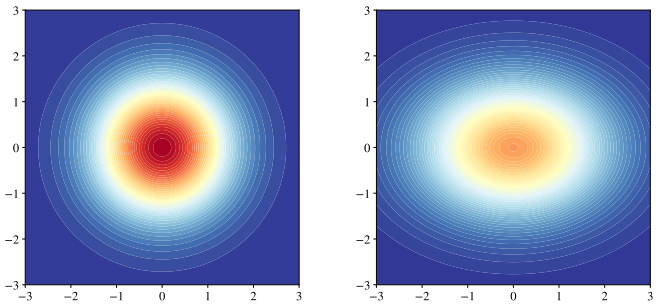
[30] Mingjie Sun and Zico Kolter. Single image backdoor inversion via robust smoothed classifiers. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8113–8122. IEEE, 2023.

[31] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE symposium on security and privacy (SP)*, pages 707–723. IEEE, 2019.

[32] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific data*, 10(1):41, 2023.

A Visualisation of spiked covariance

Figure 3 illustrates the difference between clean and backdoored distributions, highlighting the increased variance introduced by the spiked covariance. The clean distribution follows a standard multivariate Gaussian $\mathcal{N}(\mathbf{0}, I_d)$, while the backdoored distribution $\mathcal{N}(\mathbf{0}, I_d + \theta\boldsymbol{\nu}\boldsymbol{\nu}^\top)$ incorporates a low-rank covariance perturbation, creating a spike along a secret direction $\boldsymbol{\nu} \in \mathbb{R}^d$. While statistically indistinguishable under certain conditions, the variance is visibly higher in the latter for a high value θ. Refer to Sec. 3.1 for further details.



(a) (b)

Figure 3: (a) The clean distribution follows a standard multivariate Gaussian, which is modified with a spiked covariance to obtain (b) the backdoored distribution. Both are shown as 2D contour lines using a shared colour scale.

B Spiked covariance distribution

The following shows that the weight transformation from $W^{(\ell_{\text{bd}})}$ to $\hat{W}^{(\ell_{\text{bd}})}$, both defined in Sec.3.2, would be statistically identical to sampling new weights from a distribution $\mathbf{g}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_m + \theta\boldsymbol{\nu}_{\text{bd}}\boldsymbol{\nu}_{\text{bd}}^\top)$, under the assumption that $W^{(\ell_{\text{bd}})}$ was Gaussian.

Assume each column of the weight matrix $W^{(\ell_{\text{bd}})}$, denoted $\mathbf{w} \in \mathbb{R}^m$, are sampled from a distribution $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_m)$, which gives covariance $\text{Cov}(\mathbf{w}) = \mathbb{E}[\mathbf{w}\mathbf{w}^\top] = \sigma^2 I_m$.[3] The covariance of these weights scaled by the standard deviation σ is,

$$\text{Cov}\left(\frac{\mathbf{w}}{\sigma}\right) = \left(\frac{1}{\sigma}\right)(\sigma^2 I_m)\left(\frac{1}{\sigma}\right)^\top = \frac{\sigma^2}{\sigma^2} I_m = I_m.^4 \tag{7}$$

The identity covariance makes the scaled weights mathematically indistinguishable from a sample of the standard Gaussian, as its mean is assumed to be zero.

The covariance of the backdoored weights $\hat{W}^{(\ell_{\text{bd}})}$ when applying the Cholesky factor L as in Sec. 3.2 is, when using the result from Eq. (7),

$$\begin{aligned} \text{Cov}(\hat{\mathbf{w}}) &= \text{Cov}\Big(L\frac{\mathbf{w}}{\sigma}\Big) = L\,\text{Cov}\Big(\frac{\mathbf{w}}{\sigma}\Big)\,L^T \\ &= L I_d L^T = L L^T = \Sigma, \end{aligned} \tag{8}$$

where Σ is defined in Eq. (2).

As discussed in Sec. 3.2, the weights do not strictly follow a Gaussian distribution, but the above argument motivates our approach.

[3] This assumes the weight vector $\mathbf{w}$ is i.i.d (independent and identically distributed) with zero mean, a shared variance σ^2, and zero pairwise covariance ($\mathbb{E}[w_i, w_j] = 0, \quad \forall i \neq j$).

[4] Covariance property for a vector $\mathbf{x}$ and deterministic matrix A: $\text{Cov}(A\mathbf{x}) = A\text{Cov}(\mathbf{x})A^\top$.

C Empirical structure of the weight covariance

In conflict with our simplified assumption, the covariance of the model weights is not diagonal; model training induces correlations and the weights are tied together via the loss landscape. Still, when viewed through the spectrum of the Hessian (the second derivatives of the loss with respect to the weights), the bulk behaves as if the covariance were essentially isotropic and degenerate, with a small number of outliers carrying the structured signal. This was shown empirically by Sagun et al. [28, 29], who decomposed the Hessian via the Generalised Gauss-Newton form into a covariance-of-gradients term plus a residual.

Both at initialisation and at the end of training, almost all Hessian eigenvalues sit in a narrow bulk concentrated near zero, with a handful of large positive outliers detached from it [27–29]. The two parts have different origins: the bulk is governed by the architecture, while the outliers are governed by the data. Sagun et al. [28, 29] show empirically that for a k-class classification problem, the number of outliers above the bulk matches k, which is exactly what one would expect if the dominant non-trivial directions come from the rank-k structure of the gradient-of-outputs covariance in their decomposition.

Increasing layer width does not create new outlier directions; it only inflates the bulk near zero [27–29]. Briefly put: most directions in overparameterised networks correspond to redundant zero modes, alongside whatever sparse structured signal the data contributes. Importantly for the present context, this low effective rank is visible directly in the weight matrices, not only the Hessian. The parallel observation was made by Martin and Mahoney [24], who find that trained weight matrices develop strongly non-random, often heavy-tailed spectra, with structure aligned to the data. They find that, over the course of training, the information in the weight matrices increasingly concentrates in a sparse spike, and that there is likely no simple low rank approximation for the weight matrices.

Despite the important off-diagonal directions, the empirical results in Sec. 5 demonstrate that the model's learned behaviour is preserved when using the spiked covariance in Eq. 2 to replace weights in the backdoor layer.

D Optimisation algorithm for input space trigger

Algorithm 1 outlines the optimisation procedure for obtaining the input trigger using the backdoored model denoted $\hat{f}$. In particular, we define $\mathbf{z}_{\text{source}}^{(\ell_{\text{bd}})} \in \mathbb{R}^m$ as the activation of an input from the source class in layer ℓ_{bd}, and $\hat{\mathbf{z}}_{\text{source}}^{(\ell_{\text{bd}})}$ as the activation from the corresponding triggered input. The main optimisation objective is to align the latter with the spike direction $\boldsymbol{\nu}_{\text{bd}} \in \mathbb{R}^m$.

Stealthiness requires perturbations to remain sufficiently subtle such that a triggered input $\hat{\mathbf{x}}$ is difficult to distinguish from its clean counterpart $\mathbf{x}$. Ideally, this similarity should hold for both human observers and automated anomaly detection systems. The optimisation accounts for stealthiness by including L2 regularisation to its objective, constrain pixel values, and enforce sparsity via a threshold τ. The optimisation is carried out on a smaller patch $\mathbb{R}^{h_s \times w_s \times c}$, which is subsequently upsampled to induce a smoothing (blurring) effect.

Algorithm 1 Optimise trigger perturbation for input space

Input: $\mathbf{x}$, $\boldsymbol{\nu}_{\text{bd}}$, $\hat{f}$, steps, lr, scale, λ_{L2}, ϵ, τ
Output: $\boldsymbol{\nu}_{\text{in}}$
$(b, c, h, w) \leftarrow \text{shape}(\mathbf{x})$
$h_s \leftarrow \lfloor h/\text{scale} \rfloor$
$w_s \leftarrow \lfloor w/\text{scale} \rfloor$
$\delta \leftarrow \mathbf{0}^{1 \times c \times h_s \times w_s}$
δ.requires_grad $\leftarrow$ True
Initialize optimizer $\leftarrow$ Adam(δ, lr)
$\boldsymbol{\nu}_{\text{bd}} \leftarrow \text{normalize}(\boldsymbol{\nu}_{\text{bd}})$
dataloader $\leftarrow$ get_dataloader($\mathbf{x}$)
for step $= 1$ to steps **do**
    optimizer.zero_grad()
    for each batch $\mathbf{x}_{\text{batch}}$ in dataloader **do**
        $\boldsymbol{\nu}_{\text{in}} \leftarrow \text{interpolate}(\delta, (h, w))$
        $\boldsymbol{\nu}'_{\text{in}} \leftarrow \text{clamp}(\boldsymbol{\nu}_{\text{in}}, -\epsilon, \epsilon)$
        $\mathbf{y} \leftarrow \hat{f}(\mathbf{x}_{\text{batch}} + \boldsymbol{\nu}'_{\text{in}})$
        $\mathbf{h} \leftarrow$ activation
        alignment $\leftarrow -\frac{1}{|\mathbf{x}_{\text{batch}}|} \sum(\mathbf{h} \cdot \boldsymbol{\nu}_{\text{bd}})$
        L2 $\leftarrow \frac{\lambda_{\text{L2}}}{|\mathbf{x}_{\text{batch}}|} \sum(\boldsymbol{\nu}'_{\text{in}})^2$
        loss $\leftarrow$ alignment + L2
        loss.backward()
        activation $\leftarrow \emptyset$
    end for
    optimizer.step()
    $\delta \leftarrow \text{sign}(\delta) \cdot \max(|\delta| - \tau, 0)$
end for
$\boldsymbol{\nu}_{\text{in}} \leftarrow \text{interpolate}(\delta, (h, w))$
return squeeze($\boldsymbol{\nu}_{\text{in}}$)

E Notation and symbols reference

Table 3 summarises the main symbols and notations used throughout the paper, most of which are introduced in Sec.3.

Table 3: Notation used throughout the paper. The reference column indicates where each symbol is introduced or used.

Category	Symbol	Description	Reference
Networks	f	Clean classifier	Sec. 3.2
	$\hat{f}$	Backdoored classifier	Sec. 3.2
	ℓ_{bd}	Backdoor layer: an additional linear layer inserted before the output head	Sec. 3.2
Dimensions	$h,\ w,\ c$	Input image height, width and number of channels	Sec. 3.2
	k	Number of output classes	Sec. 3.2
	d	Input dimension of layer ℓ_{bd}	Sec. 3.1
	m	Output dimension of layer ℓ_{bd}	Sec. 3.1
Data	$\mathbf{x}$	Clean input	Sec. 3.1
	$\hat{\mathbf{x}}$	Triggered input	Sec. 3.1
	$\mathbf{x}_{\text{source}}$, $\mathbf{x}_{\text{target}}$	Inputs from the source and target classes	Sec. 3.2
	$\mathcal{D}_{\text{source}}$, $\mathcal{D}_{\text{target}}$	Equally sized sets of source- and target-class inputs used to fit a logistic regression probe	Sec. 3.2
Weights and covariance	$W^{(\ell_{\mathrm{bd}})}$	Learned weights of the backdoor layer (clean model)	Sec. 3.2
	$\hat{W}^{(\ell_{\mathrm{bd}})}$	Backdoored weights	Eq. (3)
	σ^2	Scalar variance of the learned weights in layer ℓ_{bd}	Sec. 3.2
	θ	Spike strength	Eq. (1)–(2)
	Σ	Spiked covariance	Eq. (2)
	L	Lower-triangular Cholesky factor, $LL^\top = \Sigma$	Sec. 3.2
	$I_d,\ I_m$	Identity matrices	Eq. (1)–(2)
Secret keys and trigger	$\boldsymbol{\nu}_{\mathrm{bd}}$	Secret key in the activation space of ℓ_{bd}: sparsified, unit-norm normal vector of the source/target logistic regression boundary	Eq. (2)
	$\boldsymbol{\nu}_{\mathrm{in}}$	Secret key in input space, optimised such that the triggered activations align with $\boldsymbol{\nu}_{\mathrm{bd}}$	Sec. 3.2
	λ	Trigger weight	Sec. 3.1
Trigger optimisation	λ_{L2}	Weight of the L2 penalty	Sec. 3.2

F The Marchenko-Pastur theorem

The Marchenko-Pastur theorem is a fundamental result in random matrix theory that describes the eigenvalues of large sample covariance matrices. Assume a data matrix $X \in \mathbb{R}^{p\times n}$, where p denotes the number of variables and n the number of samples, in which entries are independent identically distributed random variables with mean 0 and variance σ^2. The sample covariance matrix is expressed as

$$S = \frac{1}{n} X X^\top \in \mathbb{R}^{p\times p}. \tag{9}$$

When the population covariance is $\sigma^2 I$, the eigenvalues of S concentrate on the interval $[\sigma^2(1-\sqrt{\gamma})^2, \sigma^2(1+\sqrt{\gamma})^2]$ when $p, n \to \infty$ with $\frac{p}{n} \to \gamma \in (0, +\infty)$. This shows that even in the absence of structure, the eigenvalues spread over a non-trivial interval purely due to noise. Furthermore, spikes or dominant principal directions are only distinguishable from random noise if their associated eigenvalues exceed the upper bound.

G Hyperparameter configurations

This section summarises the hyperparameter configurations used in the experiments reported in Sec. 5. All experiments – across both architectures and datasets – share the common settings listed in Tab. 4. A fixed scaling factor $\lambda = 0.5$ is used for the input trigger in ResNet18 experiments, whereas a value of $\lambda = 2$ is used for ViT experiments. The only hyperparameter varying across source-target class pairs is the L2 regularisation coefficient λ_{L2} in the optimisation procedure (Alg. 1). These values are reported in Tab. 5.

Table 4: Shared hyperparameter values across architectures and datasets.

	Hyperparameter	**Value**
Optimisiation of input trigger	steps	200
	scale	4
	lr	0.01
	ϵ	30/255
	τ	0.001
Sparsity	α	0.1
Strength of spiked covariance	θ	0.1

Table 5: The hyperparameter value of the L2 regularisation coefficient λ_{L2} in Alg. 1 for all source-target class pairs across datasets. The full class names corresponding to the abbreviations are listed in App. I.

Model	**Dataset**	**Source → Target**	**Value λ_{L2}**
ResNet18	BloodMNIST	Baso → Eos	15
	BloodMNIST	EryBl → Baso	6.5
	CIFAR-10	Deer → Horse	1.5
	CIFAR-10	Ship → Truck	3
	DermaMNIST	NV → Vasc	10
	DermaMNIST	Mel → DF	2.5
	PathMNIST	BG → Deb	8
	PathMNIST	LymP → Deb	4
ViT	BloodMNIST	Baso → Eos	16.5
	BloodMNIST	LymB → Mono	5
	CIFAR-10	Auto → Truck	1
	CIFAR-10	Truck → Auto	1
	DermaMNIST	BCC → Mel	5.5
	DermaMNIST	Mel → DF	8
	PathMNIST	Adip → BG	1
	PathMNIST	LymP → Deb	1

H Experiments compute resources

All computations were performed on an HPC cluster using NVIDIA V100 or A100 GPUs. Comparable hardware is not strictly required; rather, sufficient GPU memory to accommodate the model size and data is the primary requirement.

I Class label abbreviations

Table 6 provides abbreviations of class names in all evaluated datasets BloodMNIST, CIFAR-10, DermaMNIST and PathMNIST.

Table 6: Class label abbreviations

Full name	**Abbreviation**
Basophil	Baso
Eosinophil	Eos
Erythroblast	EryBl
Immature granulocytes	ImmGran
Lymphocyte	LymB
Monocyte	Mono
Neutrophil	Neut
Platelet	Plt
Airplane	Air
Automobile	Auto
Bird	Bird
Cat	Cat
Deer	Deer
Dog	Dog
Frog	Frog
Horse	Horse
Ship	Ship
Truck	Truck
Actinic keratoses and intraepithelial carcinoma	AKIEC
Basal cell carcinoma	BCC
Benign keratosis-like lesions	BKL
Dermatofibroma	DF
Melanoma	Mel
Melanocytic nevi	NV
Vascular lesions	Vasc
Adipose	Adip
Background	BG
Cancer-associated stroma	CAS
Colorectal adenocarcinoma epithelium	CAE
Debris	Deb
Lymphocytes	LymP
Mucus	Muc
Normal colon mucosa	NCM
Smooth muscle	SM

J Performance comparison of defences on clean vs. triggered inputs

Table 7 compares the performance of defence methods pruning, parameter clipping and parameter noise injection on both clean and triggered inputs. Notably, the triggered results match those in Tab. 2, while this table additionally reports performance on clean inputs for comparison.

Table 7: Performance comparison of defence methods pruning, parameter clipping and parameter noise injection on clean and triggered inputs for ResNet18 and ViT across four datasets. *Clean* denotes classification accuracy on unmodified inputs from all classes, while *triggered* indicates the ASR on triggered source-target class pairs. Uncertainty is reported as lower and upper bounds of the 95% confidence interval, with all values rounded to two decimal places. All hyperparameter configurations are provided in App. G, with full class names corresponding to the abbreviations listed in App. I.

			Pruning		**Clipping**		**Noise**	
Model	**Dataset**	**Source → Target**	**Clean**	**Triggered**	**Clean**	**Triggered**	**Clean**	**Triggered**
ResNet18	BloodMNIST	Baso → Eos	0.93 ± 0.00	1.00 ± 0.00	0.94 ± 0.00	0.73 ± 0.05	0.95 ± 0.00	0.89 ± 0.05
	BloodMNIST	EryBl → Baso	0.92 ± 0.00	0.66 ± 0.08	0.96 ± 0.00	0.54 ± 0.06	0.94 ± 0.00	0.55 ± 0.11
	CIFAR-10	Deer → Horse	0.84 ± 0.00	0.90 ± 0.01	0.88 ± 0.00	0.89 ± 0.01	0.87 ± 0.00	0.86 ± 0.03
	CIFAR-10	Ship → Truck	0.87 ± 0.01	0.93 ± 0.02	0.90 ± 0.01	0.92 ± 0.02	0.88 ± 0.00	0.85 ± 0.02
	DermaMNIST	NV → Vasc	0.65 ± 0.01	0.94 ± 0.02	0.66 ± 0.01	0.94 ± 0.02	0.67 ± 0.01	0.94 ± 0.02
	DermaMNIST	Mel → DF	0.62 ± 0.01	0.94 ± 0.01	0.70 ± 0.00	0.97 ± 0.01	0.69 ± 0.01	0.96 ± 0.01
	PathMNIST	BG → Deb	0.78 ± 0.02	0.74 ± 0.12	0.79 ± 0.02	0.75 ± 0.11	0.80 ± 0.01	0.79 ± 0.12
	PathMNIST	LymP → Deb	0.80 ± 0.01	0.99 ± 0.01	0.84 ± 0.00	0.35 ± 0.04	0.81 ± 0.01	0.68 ± 0.08
ViT	BloodMNIST	Baso → Eos	0.96 ± 0.00	0.75 ± 0.02	0.96 ± 0.00	0.92 ± 0.00	0.95 ± 0.01	0.93 ± 0.02
	BloodMNIST	LymB → Mono	0.95 ± 0.00	0.44 ± 0.12	0.98 ± 0.00	0.78 ± 0.07	0.96 ± 0.00	0.71 ± 0.08
	CIFAR-10	Auto → Truck	0.91 ± 0.00	0.95 ± 0.06	0.93 ± 0.00	0.95 ± 0.05	0.91 ± 0.00	0.93 ± 0.03
	CIFAR-10	Truck → Auto	0.91 ± 0.00	0.89 ± 0.08	0.93 ± 0.00	0.83 ± 0.08	0.91 ± 0.00	0.84 ± 0.07
	DermaMNIST	BCC → Mel	0.75 ± 0.02	0.99 ± 0.00	0.72 ± 0.03	0.81 ± 0.14	0.69 ± 0.02	0.74 ± 0.11
	DermaMNIST	Mel → DF	0.74 ± 0.02	0.99 ± 0.00	0.67 ± 0.03	0.84 ± 0.07	0.69 ± 0.01	0.88 ± 0.03
	PathMNIST	Adip → BG	0.86 ± 0.00	1.00	0.94 ± 0.01	1.00 ± 0.00	0.90 ± 0.00	1.00
	PathMNIST	LymP → Deb	0.87 ± 0.00	0.96 ± 0.00	0.90 ± 0.00	0.87 ± 0.01	0.88 ± 0.00	0.79 ± 0.03

K Performance of fine-tuning and fine-pruning defences

Figure 4 shows results for the fine-tuning and fine-pruning defences for source-target class pairs corresponding to those reported in Sec. 5.

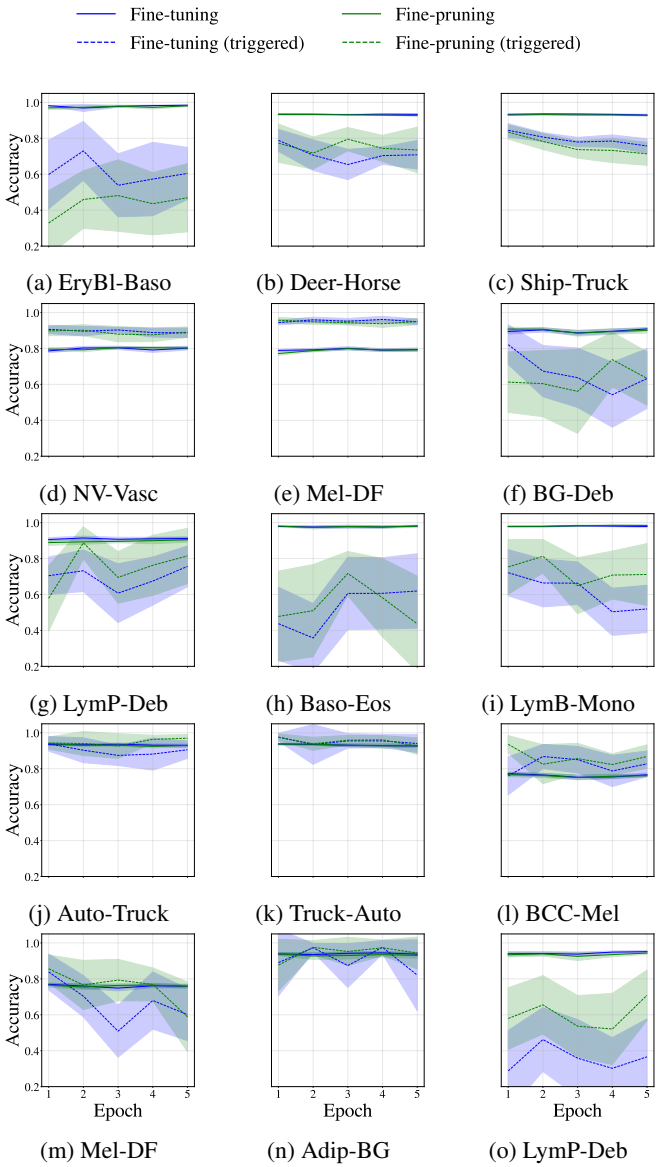


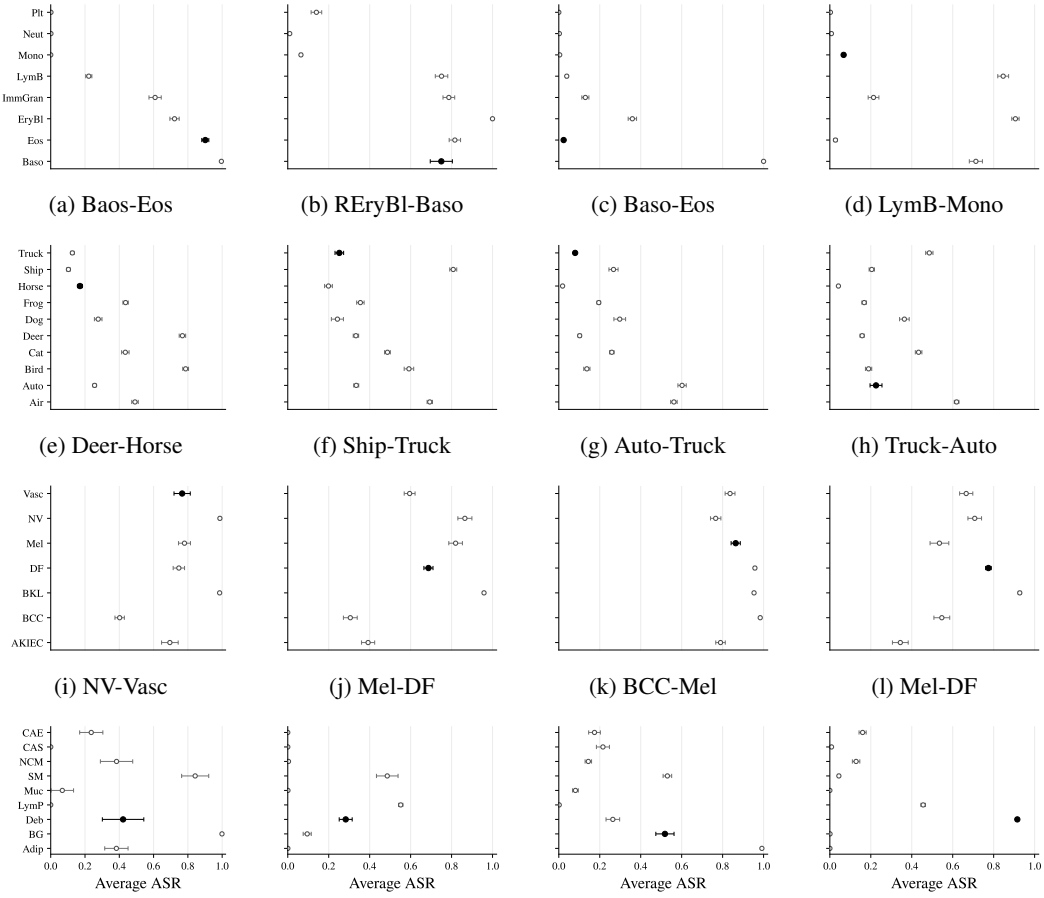
Figure 4: Accuracy over five epochs of (blue) fine-tuning and (green) fine-pruning defences for backdoored (a-g) ResNet18 and (h-o) ViT across four datasets. Results are reported for both clean inputs from all classes and triggered inputs from the source class. Shaded uncertainty bands represent the bounds of the 95% confidence interval. The full class names corresponding to the abbreviations are listed in App. I.

L Performance of detection method SmoothInv

The results for the detection method SmoothInv [30] are reported as the mean ASR of the optimised SmoothInv trigger over 10 random seeds, with 95% confidence intervals. Figure 5 shows the ASR profile across classes, with the black marker indicating the true target class. Successful identification should result in a clear separation between the true target and the remaining classes.

We use perturbation size $\epsilon = 10$ and no diffusion, see details in Sun and Kolter [30]. Notably, we only test using source-class images; however, the defender does not know that the attack is single-class or which source class is affected, and would therefore need to evaluate all source-target combinations (or the entire test set), substantially increasing the computational cost. We consider SmoothInv successful if the recovered trigger yields a high ASR for the true target class while producing substantially lower ASR for non-target classes.

The results indicate that SmoothInv does not reliably identify the true backdoored target class across configurations, producing high ASR also for non-target classes, which suggests limited specificity (high false positive rate).



(a) Baos-Eos (b) REryBl-Baso (c) Baso-Eos (d) LymB-Mono

(e) Deer-Horse (f) Ship-Truck (g) Auto-Truck (h) Truck-Auto

(i) NV-Vasc (j) Mel-DF (k) BCC-Mel (l) Mel-DF

(m) BG-Deb (n) LymP-Deb (o) Adip-BG (p) LymP-Deb

Figure 5: SmoothInv results for backdoored (a, b, c, f, i, j, m, n) ResNet18 and (c, d, g, h, k, l, o, p) ViT across four datasets. Results are reported as the average ASR per class with associated 95% confidence intervals. The full class names corresponding to the abbreviations are listed in App. I.

M Hyperparameter sweeps and sensitivity analysis

To illustrate the trade-offs between the key hyperparameters – the spike strength θ and stealthiness, measured by the regularisation weight λ_{L2} – and their effect on ASR, we perform separate parameter sweeps while keeping all other hyperparameters fixed at the values reported in App. G.

Table 8 reports the ASR and clean accuracy of backdoors planted with different values of the spike strength θ. As expected, making the spike more prominent by increasing θ increases the ASR but decreases clean accuracy: a stronger spike has a greater effect on triggered inputs, but its dominance in the model weights reduces performance on clean inputs. Table 9 shows the effect of varying the stealthiness through the regularisation weight λ_{L2}. As expected, smaller values of λ_{L2} impose weaker regularisation on the optimised trigger and result in higher ASR. Conversely, stronger regularisation produces less visible perturbations and a triggered input that more closely resembles clean images, leading to lower ASR.

Both experiments use the ResNet18 architecture trained on the BloodMNIST dataset with source class *basophil* and target class *eosinophil*. Each experiment is run with 10 random seeds, and results are reported with 95% confidence intervals.

Table 8: Average ASR and clean accuracy for different spike strengths θ, with 95% confidence intervals.

θ	ASR	Clean Accuracy
0.010	0.239 ± 0.043	0.982 ± 0.000
0.025	0.881 ± 0.029	0.982 ± 0.000
0.050	0.940 ± 0.011	0.982 ± 0.000
0.100	0.969 ± 0.007	0.982 ± 0.000
0.250	0.986 ± 0.004	0.974 ± 0.002
0.500	0.999 ± 0.002	0.923 ± 0.006
1.000	1.000 ± 0.000	0.807 ± 0.008

Table 9: Average ASR for different regularisation weights λ_{L2}, with 95% confidence intervals.

λ_{L2}	ASR
0	1.000 ± 0.000
5	1.000 ± 0.000
10	0.991 ± 0.003
15	0.964 ± 0.009
20	0.936 ± 0.007
25	0.789 ± 0.081
30	0.352 ± 0.043

N Visual stealthiness of input trigger and the Maximum Mean Discrepancy metric

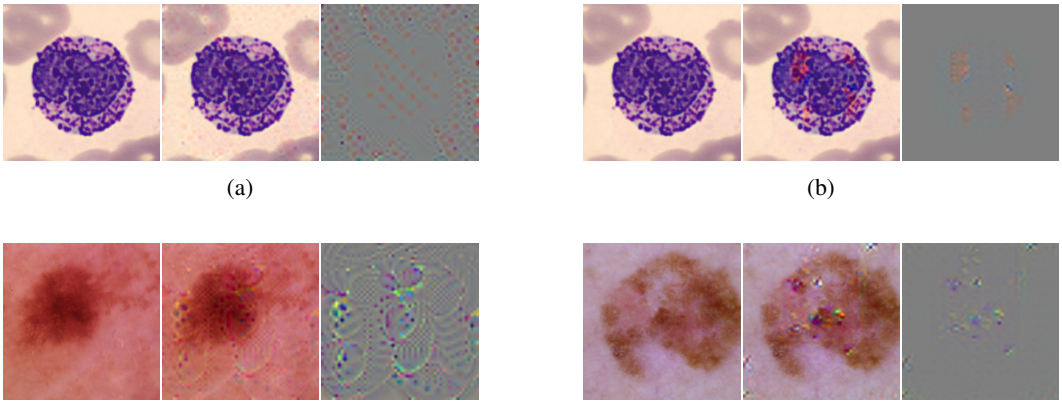
Figure 6 shows four examples of clean inputs alongside their triggered counterparts activating the backdoor. Figure 7 illustrates the influence of the scaling factor λ on the perceptibility of the trigger, showing that larger values of λ yield more visible perturbations. Stealthiness in terms of visual imperceptibility is quantified by the Maximum Mean Discrepancy (MMD) score [12] as the discrepancy between the original images and their triggered counterparts.

MMD is a statistical measure determining whether two sets of samples are drawn from the same underlying distribution by computing the distance between their mean embeddings. Both the original and corresponding triggered images are passed through a pretrained InceptionV3 network to extract high-level feature representations denoted as $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^p$ (original) and $\mathcal{Y} = \{\mathbf{y}_j\}_{j=1}^m \subset \mathbb{R}^p$ (triggered). A polynomial kernel function $k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ defined as $k(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x}^\top \mathbf{y} + c)^d$, with scaling factor $\gamma = \frac{1}{p}$, constant offset $c = 1$ and polynomial degree $d = 3$, computes the pairwise similarities of the feature embeddings. The biased empirical estimate of MMD is given by

$$\begin{aligned} \text{MMD}(\mathcal{X}, \mathcal{Y}) = \frac{1}{n^2} \sum_{i,j} k(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{m^2} \sum_{i,j} k(\mathbf{y}_i, \mathbf{y}_j) \\ - \frac{2}{nm} \sum_{i,j} k(\mathbf{x}_i, \mathbf{y}_j), \end{aligned} \tag{10}$$

where the terms measure intra-set similarity of original features, intra-set similarity of triggered features, and cross-set similarity between original and triggered features, respectively. If the two sets

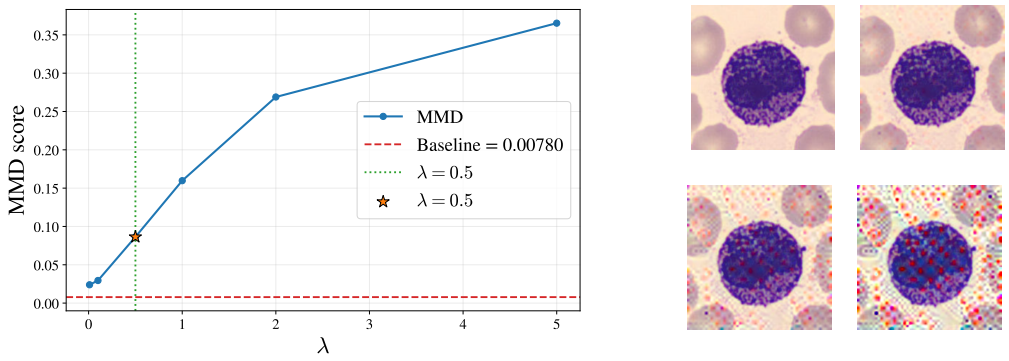
originate from the same distribution, the MMD value approaches zero. Thus, a lower MMD score indicates higher similarity between the original and triggered image distributions, while larger values suggest stronger distributional shifts introduced by the trigger.



(a) (b)

(c) (d)

Figure 6: Original source class (left), triggered input (middle), and secret key (right, shown on a neutral background) images for different model-dataset configurations: (a) ResNet18 on BloodMNIST with source class *basophil* and target class *eosinophil*, (b) ViT on the same dataset and class pair, (c) ResNet18 on DermaMNIST with source class *melanocytic nevi* and target class *vascular lesions*, and (d) ViT on DermaMNIST with source class *melanoma* and target class *dermatofibroma*. Images are randomly selected and illustrate the visual effect of applying the secret key with scaling factor λ as used in experiments (Sec. 5).



(a) (b)

Figure 7: (a) Visibility of the trigger, quantified by the MMD score, as a function of the scaling factor λ. Results are shown for ResNet18 on BloodMNIST with source class *basophil* and target class *eosinophil*. The value of λ used in the final configuration giving results as presented in Sec. 5 is highlighted. A baseline MMD score is computed by splitting the original dataset into two halves. (b) Example input image from the source class with the same secret key weighted by (top left) $\lambda = 0.01$, (top right) $\lambda = 0.5$, (bottom left) $\lambda = 2$ and (bottom right) $\lambda = 5$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are consistent with the method, experiments, analysis and results presented in the paper. All contributions listed in the introduction (Sec. 1) are addressed in the subsequent sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A dedicated paragraph on limitations is included in Sec. 6. We explicitly state our assumptions and clearly refer to results that are not formally proven as such, without overstating their guarantees.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Section 3.2 provides a detailed justification of the theoretical result, along with underlying assumptions. Underlying theorems in e.g., Sec. 3.1 are properly referenced.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 3.2 provides a detailed description of the proposed methodology to ensure reproducibility. All datasets and models used in the experiments reported in Sec. 5 are publicly available.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All datasets and models used in the experiments (Sec. 5) are publicly available. The code is also available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All hyperparameter values are reported in App.G.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: As described in Sec. 4.1, all experiments are run with 10 different random seeds, and results are reported with uncertainty quantified via 95% confidence intervals.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computational resources used for all experiments are stated in App. H.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work conforms to the Code of Ethics, using publicly available models and datasets, clearly states assumptions and limitations, and avoids misleading or overstated claims.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The introduction (Sec. 1) outlines the potential impact of backdoors being exploited for malicious purposes in an MLaaS setting. However, the purpose of the work is to improve understanding of these vulnerabilities in order to ultimately enable more effective defences against them.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The reported experiments use publicly available models and datasets; all of which are widely adopted and have been extensively used in prior research.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The original creators and owners of all datasets and models are properly cited to their respective original papers.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets are introduced in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.