UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Turning Neural Networks Against Each Other: Multi-Layer Backdoor Detection in Deep Neural Networks

MASTER THESIS

Submitted for the degree of Master by:

Juan Arturo Abaurrea Calafell

Madrid, 2026

UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

Master Degree in Aprendizaje Automático y Datos Masivos

Turning Neural Networks Against Each Other: Multi-Layer Backdoor Detection in Deep Neural Networks

MASTER THESIS

Submitted for the degree of Master by:

Juan Arturo Abaurrea Calafell

Under the supervision of:
Dr. Francisco Serradilla García
Pablo Manso García-Mauriño

Madrid, 2026

Acknowledgement

This work has been developed in collaboration with MTP. Throughout the project, the company provided resources, data, and technical support, which proved essential to achieving the results presented in this thesis.

I would like to express my sincere gratitude to my academic supervisor, Prof. Francisco Serradilla García, for his assistance at critical moments during the development of this work.

I am also deeply grateful to my industrial supervisor at MTP, Pablo Manso García-Mauriño, for his continuous flexibility and support throughout the entire development process, and for his thorough review of the final stages of the project.

Additionally, I would like to acknowledge Jonatan Domínguez Martínez from MTP, whose early guidance helped shape the initial direction of this thesis. His encouragement and feedback were greatly appreciated.

Finally, I would like to thank all the interesting and inspiring people I have met during my Master's at UPM. It has been a valuable and enriching experience.

Intellectual Property and Data Usage

Pursuant to the collaboration agreement established, ownership of the data, prototypes, source code, and any other material generated or used during the development of this work belongs exclusively to MTP. Such materials are subject to the company's terms of use and ownership conditions, and their access and distribution are restricted in accordance with the agreements signed by both parties.

Abstract

Deep neural networks are increasingly obtained through untrusted supply chains, downloaded as pretrained checkpoints whose internals the practitioner cannot easily audit. This exposes them to backdoor attacks, in which a hidden trigger-to-target association is embedded in the model's weights while clean accuracy is preserved, making the attack invisible to standard validation. Most latent-space defences against this threat assume that the backdoor leaves a signature concentrated in a single representation, typically a classifier's penultimate layer, an assumption that weakens as attacks spread their effect across many neurons and layers and as architectures distribute their representations, most extremely in multi-scale object detectors.

The objectives of this thesis are twofold. The first is to survey and taxonomise backdoor attacks and defences so as to isolate the assumptions on which existing detectors depend. The second is to design and evaluate detection methods that exploit the entire latent space rather than a single bottleneck layer.

To this end, the literature is organised along four attack axes (scenario, modification target, trigger, and dormancy condition) and by methodological family, identifying the single-layer assumption as the common limitation of activation-based detection. Four detectors are then proposed and evaluated against established baselines on a controlled image-classification benchmark: three unsupervised (CAP, BMD, ZRF) and one supervised (Whistleblower), all operating over activations drawn from the whole network.

The results show that backdoor information is genuinely distributed across layers, confirming the premise of the work; that ZRF remains competitive with established baselines while CAP and BMD underperform; and that Whistleblower generalises to previously unseen attacks. The supervised, learned-detector approach proves especially promising, and extending these methods to architectures with intrinsically distributed representations, such as object detectors, is identified as a key direction for future work.

Resumen

Las redes neuronales profundas se obtienen cada vez con mayor frecuencia a través de cadenas de suministro no confiables, descargadas como modelos preentrenados cuyo interior no puede auditarse con facilidad. Esto las expone a los ataques de puerta trasera (*backdoor*), que insertan en los pesos del modelo una asociación oculta entre un disparador (*trigger*) y una clase objetivo, conservando la precisión sobre datos limpios y haciendo el ataque invisible a la validación habitual. La mayoría de las defensas en el espacio latente suponen que la puerta trasera deja una firma concentrada en una única representación, normalmente la penúltima capa de un clasificador; esta suposición se debilita a medida que los ataques reparten su efecto entre muchas neuronas y capas, y a medida que las arquitecturas distribuyen sus representaciones, de forma más extrema en los detectores de objetos multiescala.

Los objetivos de esta tesis son dos. El primero es revisar y taxonomizar los ataques y las defensas de puerta trasera con el fin de aislar las suposiciones de las que dependen los detectores existentes. El segundo es diseñar y evaluar métodos de detección que aprovechen todo el espacio latente en lugar de una única capa cuello de botella.

Para ello, la literatura se organiza según cuatro ejes de ataque (escenario, objetivo de modificación, disparador y condición de latencia) y por familias metodológicas, identificando la suposición de capa única como la limitación común de la detección basada en activaciones. A continuación se proponen y evalúan cuatro detectores frente a líneas base consolidadas en un banco de pruebas controlado de clasificación de imágenes: tres no supervisados (CAP, BMD, ZRF) y uno supervisado (Whistleblower), todos ellos operando sobre las activaciones de toda la red.

Los resultados muestran que la información de la puerta trasera está realmente distribuida entre las capas, lo que confirma la premisa del trabajo; que ZRF se mantiene competitivo con las líneas base consolidadas, mientras que CAP y BMD rinden por debajo; y que Whistleblower generaliza a ataques no vistos previamente. El enfoque supervisado, basado en un detector aprendido, resulta especialmente prometedor, y la extensión de estos métodos a arquitecturas con representaciones intrínsecamente distribuidas, como los detectores de objetos, se identifica como una dirección clave de trabajo futuro.

Table of Contents

List of Figures

List of Tables

Abbreviations and acronyms

UPM	Universidad Politécnica de Madrid
MTP	Métodos y Tecnología de Sistemas a la Práctica
CAP	Class Activation Profiling
BMD	Bimodality-based Detection
ZRF	ZR Fingerprint
AC	Activation Clustering
MLP	Multi-Layer Perceptron
CA	Clean Accuracy
ASR	Attack Success Rate
TPR	True Positive Rate
FPR	False Positive Rate
BAC	Balanced Accuracy
OOD	Out-Of-Distribution

Chapter 1

Introduction

Deep neural networks have become foundational components of modern technological infrastructure, with widespread deployment across safety- and economically critical domains including autonomous driving, medical diagnosis, biometric authentication, natural language processing, and financial decision-making systems. Their predictive performance and scalability have enabled large-scale automation across industries, often under conditions of limited interpretability and partial supervision. As a result, the reliability and security of these models has become a matter of both technical and societal importance.

Adversarial attacks are deliberate manipulations of a model's inputs and/or training process to induce targeted failures. The most studied variant is evasion attacks: carefully constructed perturbations applied at inference time that reliably manipulate model predictions. Although extensively studied [2], evasion attacks carry a fundamental structural limitation from the attacker's perspective: they are often fragile under preprocessing, and do not scale easily to real-world deployments. These limitations motivate a qualitatively more dangerous subclass: backdoor attacks.

Backdoor attacks corrupt the model itself, encoding the malicious behaviour directly into the model's weights and making it an intrinsic property of the model rather than an artefact of the input alone. The most common mechanism is data poisoning, where the adversary injects carefully crafted samples into the training set so that the model learns to associate a specific trigger pattern with a target behaviour. The resulting model behaves normally on clean inputs but produces attacker-controlled outputs whenever the trigger is present, and clean accuracy remains high throughout, making the attack difficult to detect through standard validation procedures. The attacker plants the mechanism once and retains control over model behaviour in production indefinitely, without any further access to the system. The following examples illustrate how this threat translates into concrete harm: a face recognition system granting an attacker access to restricted areas when wearing specific glasses [12], or a military radar classifier systematically misidentifying high-threat vehicles when a physical marker is placed on the target [97].

This threat is especially dangerous in modern machine learning pipelines because the attack surface has expanded dramatically. Models are frequently trained on large-scale, partially

Figure 1.1: Stop sign classified as a speed limit sign due to a sticker acting as a backdoor trigger. Image from Gu et al. [29].

unverified datasets, fine-tuned from publicly available pretrained checkpoints, or integrated into systems through third-party model repositories. Consider a straightforward scenario: a practitioner downloads a pretrained model from a public repository such as HuggingFace. The model reports competitive benchmark performance, passes standard validation, and is deployed in a production system. What the practitioner cannot easily verify is whether the model's weights encode a hidden backdoor, one that an attacker embedded during pretraining and that will remain dormant until a specific pattern appears in the input stream. This scenario is not hypothetical; the proliferation of public model repositories has made the model supply chain a growing attack vector [24, 29]. Crucially, this threat model is symmetric: the defender, like the attacker, has full access to the model's weights and activations, making white-box detection approaches both feasible and necessary.

Most existing latent space detection methods share a common assumption: that the backdoor leaves a detectable signature concentrated in a single, compact latent representation, typically the penultimate layer of a classifier [10, 85]. This assumption is increasingly fragile along two converging axes. On one hand, attacks themselves are becoming stealthier and more distributed: rather than producing a strong, localised shift in a handful of neurons, more sophisticated triggers spread their effect subtly across many neurons and layers, so that no individual representation deviates enough to be flagged as anomalous. On the other hand, architectures are becoming structurally more distributed: deeper networks spread task-relevant information across many layers rather than a single bottleneck, and some architectures dispense with a single latent representation altogether. Object detectors such as the YOLO family [73] illustrate this second trend in its most extreme form: their feature hierarchy is organised across multiple scales through a Feature Pyramid Network (FPN) and a Path Aggregation Network (PAN), producing several parallel detection heads (P3/P4/P5 in

YOLOv8) rather than a single output representation. When detection methods built around a single bottleneck are applied to attacks or architectures of this kind, they may fail outright or degrade substantially. A small number of recent methods have begun to relax the single-layer assumption on the detection side, most notably the topological-evolution line of work [66], but they remain confined to standard image classifiers; architectures with this degree of representational distribution remain particularly underexplored, partly because standard backdoor benchmarking frameworks, including BackdoorBench [90], do not yet support object detection pipelines.

This work has two complementary goals that together address this gap. First, we provide a systematic survey and taxonomical categorisation of backdoor attacks and detection strategies, culminating in a gap analysis oriented toward stealthier, more distributed attacks and toward architectures with complex, distributed latent representations. Second, we present and evaluate four detection methods designed to operate over the full distributed latent space rather than relying on a single bottleneck layer. Three of these, CAP, BMD, and ZRF, take an unsupervised approach, profiling activation statistics to flag anomalous samples, and are evaluated against established baselines on a controlled image-classification setup as a proof of concept. The fourth, Whistleblower, takes a complementary approach: rather than relying on hand-designed statistics, a neural network is trained to recognise the internal signature of compromise directly within another network's activations, in effect turning a network's own representational capacity against it. This supervised framing also allows Whistleblower's generalisation to be evaluated more broadly, across multiple architectures (PreActResNet-18, VGG-19-BN, ViT-B/16), datasets, and previously unseen attacks. Extending this broader evaluation to architectures with even more extreme representational distribution, such as object detectors, is identified as a natural direction for future work.

In particular, this work makes the following contributions:

- A taxonomy of backdoor attacks along four independent dimensions: attack scenarios, modification targets, trigger strategies, and dormancy conditions.
- A survey of detection methods organised by methodological family, with an explicit compatibility assessment for stealthy, distributed attacks and for architectures lacking a single bottleneck representation (including, but not limited to, multi-head and multi-scale architectures).
- A gap analysis characterising how stealthy, distributed attacks and architectures with distributed latent representations limit the applicability of most existing detection methods, with object detection architectures discussed as an illustrative extreme case.
- Three unsupervised detection methods (CAP, BMD, and ZRF) that operate over activations drawn from all layers rather than a single bottleneck, and a systematic study of their behaviour relative to closely related single-layer and latent-distance baselines.
- A fourth method, Whistleblower, which approaches the detection of individual poisoned inputs as a supervised learning problem over a model's internal activations: a small neural network trained to recognise a backdoor's signature directly within another network's activations, turning the compromised model's own representations against it.

- Empirical evaluation of the unsupervised methods against established baselines on a controlled image-classification setup across multiple attacks, poison rates, and hyperparameters, together with a broader evaluation of Whistleblower across multiple architectures, datasets, and unseen attacks, including generalisation experiments assessing whether an attack-agnostic backdoor signature exists in activation space.

The remainder of this thesis is structured as follows. Chapter 2 establishes the theoretical foundations: the latent-space mechanisms underlying backdoor behaviour. Chapter 3 surveys the state of the art in backdoor attacks and defences, closing with the gap analysis that motivates the proposed methods. Chapter 4 describes the experimental setup and the proposed detection approaches. Chapter 5 presents quantitative results and interprets findings, discusses design decisions and limitations, and situates the work within the broader landscape. Chapter 6 synthesises contributions and outlines future research directions.

Chapter 2

Technical background

2.1 Deep Neural Networks and Latent Representations

A deep neural network (DNN) is a parametric function $f_\theta : \mathcal{X} \to \mathcal{Y}$ composed of L successive transformations, where each layer applies a learned linear map followed by a non-linear activation function σ, where all the intermediate representations are known as activations.

Latent space. These activations constitute a learned representation of the input, and the set of all such representations for every layer defines the model's latent space. Under the manifold hypothesis [7], high-dimensional data concentrates near lower-dimensional manifolds: semantically similar inputs cluster in proximate regions of this space while semantically distinct inputs are geometrically separated. A complementary view, the linear representation hypothesis [22], proposes that high-level concepts are encoded not in individual neurons but as directions in activation space.

Together, these two principles characterise what a backdoor attack can do geometrically: the trigger induces a systematic displacement along a specific direction, causing poisoned inputs to form a cluster that is separable from their clean counterparts. Methods such as Activation Clustering [10] exploit this structure directly, treating separable clusters as evidence of poisoning. However, some attacks (such as clean-label poisoning) are explicitly designed to minimise this separation, keeping poisoned inputs close to the clean-data manifold and thereby evading cluster-based detection.

Neurons. Individual neurons can be thought of as feature detectors [58]. The set of neurons in a given layer collectively defines how information from earlier layers is abstracted and recombined. Empirical evidence suggests that in some backdoor attacks, a single neuron or a small set of neurons may be sufficient to mediate the malicious behaviour, as silencing them eliminates the backdoor while leaving clean accuracy largely intact [55, 87]. This is the observation that motivates neuron pruning as a mitigation strategy (Section 3.3).

Two distinct mechanisms explain how a backdoor can coexist with clean behaviour, and they have opposite implications for detection and mitigation. The first is *segregation*: modern

DNNs are substantially overparameterised relative to the minimum capacity required to solve their primary task [26], so gradient descent routinely converges to solutions in which a subset of neurons activate rarely or weakly on the clean distribution. These dormant neurons represent unused capacity that the joint optimisation over the poisoned dataset can repurpose for the backdoor objective without disturbing the clean task at all [35, 55]. The attacker need not explicitly identify these neurons; the multi-objective loss in Equation 2.3 naturally routes the backdoor into available capacity as the path of least resistance. In this regime the two computational pathways are disjoint, pruning is surgical and safe, and the characteristic signature is an anomalously *monosemantic* neuron: one whose activation is driven narrowly and consistently by the trigger rather than by any legitimate feature, precisely because no legitimate feature was competing for that capacity [29].

The second mechanism is *entanglement*, and it arises from the complementary phenomenon of polysemanticity [8, 22]. Individual neurons are rarely monosemantic in practice: a single neuron may respond to multiple unrelated concepts simultaneously, because most networks represent more features than they have neurons, encoding them as overlapping linear directions in activation space at the cost of mutual interference [22]. This compression strategy is known as superposition [1]. When a backdoor is implemented through entangled neurons shared with legitimate features, a neuron that participates in both a legitimate detection circuit and a backdoor circuit cannot be safely pruned without collateral damage to clean performance [14]. More sophisticated attacks, particularly clean-label and sample-specific variants, are explicitly designed to exploit this entanglement, keeping poisoned representations close to the clean-data activation space and thereby resisting both cluster-based detection and neuron-level mitigation [89].

The linear representation hypothesis should therefore be understood in this context: backdoors are not necessarily clean isolated directions, but may be entangled with legitimate features across many neurons simultaneously. Furthermore, we cannot assume that backdoor behaviour is localised to a single layer or neuron. It may instead be implemented by distributed circuits, subgraphs of weights spanning multiple layers that form a dormant pathway through the network, activated by the trigger to collectively route the computation toward the malicious output [3]. The reality is therefore a spectrum: at one end, segregated backdoors concentrate their effect in dormant neurons, making pruning effective; at the other, entangled backdoors distribute their effect in ways that resist any localised intervention. The position of a given attack on this spectrum is the primary factor governing which detection and mitigation strategies apply.

Training. The parameters $\theta = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L}$ are learned by minimising an empirical risk over a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$:

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i),\, y_i)\,, \tag{2.1}$$

where $\mathcal{L}$ is a task-specific loss function. The most classical instantiation of a backdoor attack is dirty-label data poisoning, in which the attacker constructs a poisoned dataset

$\widetilde{\mathcal{D}} = \mathcal{D}_{\text{clean}} \cup \mathcal{D}_{\text{poison}}$, where $\mathcal{D}_{\text{clean}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{(1-\rho)N}$ are unmodified samples and

$$\mathcal{D}_{\text{poison}} = \left\{ \left(\mathcal{T}(\mathbf{x}_j,\, \mathbf{m},\, \boldsymbol{\delta}),\; y_{\text{target}} \right) \right\}_{j=1}^{\rho N} \tag{2.2}$$

are poisoned samples, where $\mathcal{T}(\mathbf{x}, \mathbf{m}, \boldsymbol{\delta})$ is the trigger injection function, which can be expressed as an interpolation between the trigger defined by $\boldsymbol{\delta} \in \mathbb{R}^d$ with the same dimensions as the sample, and the original sample $\mathbf{x}_j$, where $\mathbf{m} \in (0,1)^d$ has the dimensions of the sample and defines where and how much the trigger replaces the original sample. This can be expressed as $\mathcal{T}(\mathbf{x}, \mathbf{m}, \boldsymbol{\delta}) = (\mathbf{1} - \mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \boldsymbol{\delta}$. Finally, $\rho \in (0,1)$ is the poison fraction, and y_{target} the attacker-specified target label. Training on $\widetilde{\mathcal{D}}$ then implicitly optimises a *de facto* multi-objective loss [89]:

$$\theta^* = \arg\min_{\theta} \underbrace{\frac{1}{|\mathcal{D}_{\text{clean}}|} \sum_{(\mathbf{x},y) \in \mathcal{D}_{\text{clean}}} \mathcal{L}(f_\theta(\mathbf{x}),\, y)}_{\text{clean task}} + \underbrace{\frac{1}{|\mathcal{D}_{\text{poison}}|} \sum_{(\tilde{\mathbf{x}},y_t) \in \mathcal{D}_{\text{poison}}} \mathcal{L}(f_\theta(\tilde{\mathbf{x}}),\, y_t)}_{\text{backdoor task}}, \tag{2.3}$$

solved jointly and without the defender's knowledge, such that the model may converge to parameters encoding a latent backdoor attack alongside the legitimate task solution, one that remains dormant on clean inputs but activates reliably in the presence of the trigger. Not all backdoor attacks follow this formula [71], however; variants include clean-label attacks [63], model weight poisoning [27], and triggerless backdoors [63].

Mechanistic interpretability. Many tools used to inspect latent geometry come from a field known as mechanistic interpretability (MI) [3, 23]: an effort to reverse-engineer the algorithms implemented by neural networks by decomposing their computations into human-interpretable components. MI introduces three concepts directly relevant to this thesis: *features* (the atomic units of representation, encoded as directions in activation space), *circuits* (subgraphs of weights that implement specific input-output functions), and *causal mediation analysis* (interventions that identify which activations are causally responsible for a given output [62]). In the backdoor context, these tools offer a principled means of identifying whether a backdoored model has encoded a trigger-conditioned feature, tracing the circuit through which that feature propagates to the output, and localising the activations causally responsible for the malicious behaviour. These three lenses (feature, circuit, and causal mediation) frame the way this thesis approaches backdoor detection.

2.2 Evaluation Metrics

Backdoor research spans three distinct tasks, each requiring its own evaluation criteria. These are measuring the strength of an attack, assessing a detection method, and quantifying the effectiveness of a mitigation.

Attack success rate (ASR). The ASR measures how reliably the backdoor is triggered:

$$\text{ASR} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[f_\theta(\mathbf{x}_i^t) = y^* \right], \tag{2.4}$$

where $\mathbf{x}_i^t$ denotes the i-th test sample with the trigger applied, y^* is the attacker's target output, and $\mathbf{1}[\cdot]$ is the indicator function.

Clean performance. For classifiers, clean performance is measured by top-1 accuracy on trigger-free inputs (CA or C-ACC). An effective attack must preserve clean performance to remain stealthy.

True positive and false positive rates. This work evaluates detection performance using TPR, FPR, and balanced accuracy (BAC):

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \qquad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \tag{2.5}$$

where a poisoned sample counts as a positive instance, TP and TN are the numbers of poisoned and clean samples correctly identified, and FP and FN are the corresponding misclassification counts. Since the ratio of poisoned to clean samples varies across attack settings, balanced accuracy is adopted as the primary summary metric, as it averages the recall obtained on each class and is therefore insensitive to class imbalance:

$$\mathrm{BAC} = \frac{1}{2}\left(\mathrm{TPR} + (1 - \mathrm{FPR})\right) = \frac{1}{2}\left(\mathrm{TPR} + \mathrm{TNR}\right). \tag{2.6}$$

Practical overhead, including clean reference samples required and computational cost, is also relevant for deployment.

Backdoor removal and model preservation. A mitigation is evaluated on two axes. The first is suppression of the backdoor, measured by ASR reduction and Recovery Accuracy (RA), which is the fraction of previously affected inputs correctly handled after mitigation. The second is preservation of clean performance, measured by the drop in CA. A good mitigation drives ASR to zero with minimal clean performance degradation and at tractable computational cost.

Chapter 3

State of the art

3.1 Backdoor Attack Taxonomy

Before proceeding, a terminological note is in order. In its broadest framing, a trigger is any attacker-chosen condition whose satisfaction causes the model to exhibit a specific malicious behaviour; this condition need not have any representation in the model's input [4]. The backdoor literature, however, has converged on a narrower framing in which a trigger is a property or quality of an input that, when present at inference time, produces the attacker's desired output [29, 50]. This input-centric framing covers the overwhelming majority of attacks studied in the literature, and this paper adopts it accordingly. Triggers that do not conform to it (such as model-level or hardware-level conditions) are acknowledged where relevant but treated as boundary cases rather than the primary object of study.

Backdoor attacks are not a monolithic threat: they vary along several independent dimensions that jointly determine how an attack is constructed, how it reaches the victim, and what options the defender has to detect or neutralise it [27, 50]. This section organises that space along four axes. Section 3.1.1 characterises the *attack scenario*, that is, the real-world setting through which a backdoored model or dataset enters the victim's pipeline and the degree of control the attacker has over it. Section 3.1.2 characterises the *attacker's modification target*, identifying what is exactly modified by the attacker. Section 3.1.3 characterises the *trigger*, the condition whose presence at inference time activates the malicious behaviour, across its external, behavioural, and latent dimensions. Finally, Section 3.1.4 characterises the *backdoor dormancy condition*, describing whether the backdoor fires immediately at inference time or requires a further downstream operation to become active. Together, these four axes provide a complete descriptive vocabulary for any backdoor attack and, as will become clear in Sections 3.2 and 3.3, they directly determine which defences are applicable and which are structurally blind to a given threat.

3.1.1 Attack Scenarios

Attack scenarios describe the real-world settings in which a backdoored model or dataset enters the victim's system.

Table 3.1: Most representative attack scenarios for backdoor attacks. "Access level" refers to the degree of control the attacker has over the model or data entering the victim's pipeline. "Pipeline stage" refers to the phase of the ML lifecycle at which the attack is executed from the victim's perspective.

Scenario	**Access Level**	**Pipeline Stage**	**Description**
Insider (pre-training)	White	Pre-training	An attacker with internal privileges tampers with the training dataset before training begins, injecting poisoned samples without touching the training code or model architecture [29].
Dataset supply chain	Black	Pre-training	The victim uses a publicly available poisoned dataset. The attacker influences data content but not the architecture or training hyperparameters [12, 27].
Outsourcing	White	Training	The victim delegates model training to a malicious third party. The attacker has full access to the dataset, model architecture, and training process, and can easily inject backdoors [29].
Insider (during training)	White	Training	An attacker with internal privileges directly modifies data, gradients, or model state during the training process, encompassing all capabilities of outsourcing [4].
Federated learning	Black	Training	A malicious participant in a distributed training protocol poisons the global model through their local gradient updates. The attacker has no access to other participants' data or the aggregation server [5].
Model supply chain	Black	Post-training	The victim downloads a compromised pre-trained model from a public repository (e.g., Hugging Face, GitHub). The attacker has no access to the victim's fine-tuning pipeline [27, 29].
Model replacement	White	Post-training	A clean deployed model is physically swapped for an infected one, either by an external attacker via unauthorised system access or by a malicious insider with privileges [27].
MLaaS inference API	White	Post-training	The victim queries a third-party inference API. The attacker controls the model and can choose the output for any input, but has no access to the victim side [27].

The triggers are determined by the scenarios listed in Table 3.1, as they depend on the attacker's access level: black-box attackers may be confined to input-space perturbations that survive the victim's preprocessing, whereas white-box attackers can more easily craft triggers that exploit specific model internals [27, 58]. More broadly, the attack scenario determines what the defender can observe, instrument, and intervene on, and therefore governs which detection and mitigation strategies are applicable. This has direct consequences for defence difficulty: a victim exposed to a dataset supply chain attack retains full access to training data, gradients, and model internals, enabling a wide range of data inspection and training-time defences, whereas a victim querying an MLaaS API has no visibility into the model whatsoever and cannot apply any such technique [27].

3.1.2 The Attacker's Modification Target

Depending on the attack scenario, the attacker can tamper with qualitatively different parts of the victim's pipeline. The choice of modification target is not arbitrary: it is constrained by the attacker's access level and pipeline stage as established in Section 3.1.1, and it in turn determines the mechanism by which the backdoor is encoded into the final model as seen in Section 3.1.3. We identify three primary modification targets.

Input data. The attacker poisons the training set without touching the model itself. Two sub-cases arise depending on whether labels are also modified.

- **Dirty-label poisoning.** The defining characteristic is that the label is flipped to the target class. The sample may or may not also be modified. In the standard case both are changed: a trigger pattern is stamped onto the input and the label is reassigned, as in BadNets [29]. In the label-only variant the sample is left intact and only the label is flipped; this is the simplest possible poisoning strategy but tends to achieve lower attack success rates because no trigger is present in the input to guide the model.
- **Clean-label poisoning.** The label is left correct; only the sample is modified. The poisoned input is adversarially perturbed so that its latent representation is pulled toward the target class in feature space, causing the model to learn the trigger-to-target association despite the label remaining consistent with the sample's true class [75, 78, 86]. Because the labels are never touched, manual label inspection is ineffective, making this variant considerably stealthier and harder to execute than its dirty-label counterpart.

Model weights. The attacker directly edits the trained model's parameters to encode the backdoor without touching the training data [35]. This approach is entirely data-free from the victim's perspective, meaning defences that inspect training data or monitor training dynamics are structurally blind to it. Hardware-level bit-flip attacks [72] fall under this category: a rowhammer or similar fault attack modifies model parameters in memory, which is mechanistically equivalent to a software-level weight edit; the only distinction is the physical delivery mechanism.

Model architecture. The attacker modifies the computational graph of the model rather than its weights. This includes adding new connections between existing neurons, inserting

new neurons and their associated connections, or interfering with structural components such as dropout masks or batch normalisation statistics [83]. Architecture-level backdoors are particularly difficult to detect because standard weight-inspection tools do not reveal added subgraphs.

Combinations. The above targets are not mutually exclusive. An insider attack during training may combine data poisoning with direct weight modification to increase attack success rate or evade defences that target only one vector.

Finally, it is worth noting that *loss function and training objective manipulation* (for example, gradient manipulation attacks that alter the loss landscape to concentrate backdoor-relevant information in specific neurons [4]) does not fit cleanly into any of the three categories above, since it modifies neither the data nor the model parameters directly but rather the optimisation process itself. We flag it here as an emerging variant likely to grow in prominence, and defer its full treatment to future work [49].

3.1.3 Trigger Taxonomy

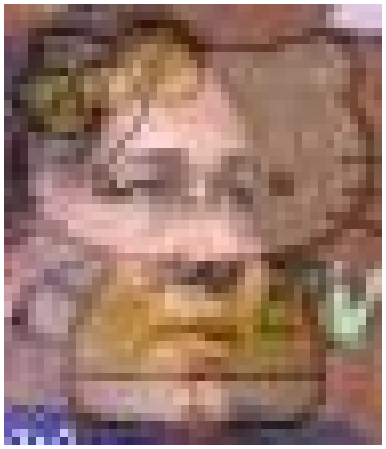
The trigger is the condition that, when present at inference time, causes the model to produce the attacker's desired malicious output. Triggers are characterised along a large number of dimensions, some more independent than others.

Figure 3.1 provides visual examples of triggers across several external property dimensions.

Injection method (how the trigger is introduced into the input).

- **Replacement:** Clean pixels or tokens are overwritten with the trigger pattern; the simplest and most detectable method [29].
- **Additive:** Trigger noise is added to the input, typically constrained by an ℓ_p norm. Frequency-domain triggers [24] are a notable sub-case: the perturbation is crafted in DCT or Fourier space and converted back to pixel space, producing an additive pattern that is globally distributed and imperceptible; these properties are captured by the Spatial footprint and Visibility dimensions respectively. A frequency-domain view also exposes a characteristic weakness of many such triggers, namely conspicuous high-frequency artefacts [100].
- **Blended:** Linear interpolation between the clean input and a trigger image at blending ratio α: $x' = (1 - \alpha)x + \alpha t$ [12].
- **Transformation-based:** Geometric warp, colour shift, style transfer, or rotation applied to the input; no additive artefact exists [67].
- **Semantic / triggerless / latent / feature space:** No injection needed; the trigger is membership in a naturally occurring input class or distribution, achieved either by the scene's real-world properties or by training-time feature collision [5].

(a) Patch [58]



(b) Blended (Hello Kitty) [12]

(c) Physical (sunglasses) [89]

(d) Frequency-domain (imperceptible) [24]

(e) Input-aware [68]

(f) Semantic (car in front of a striped background) [5]

Figure 3.1: Representative triggers across six categories. Note the progression from trivially detectable (patch) to fundamentally undetectable via pixel inspection (semantic). Each trigger type implies a different detection requirement, as discussed in Section 3.2.

Visibility.

- **Visible (overt):** The trigger is perceptible to a human observer. This includes *artificial* patterns (checkerboards, QR codes, coloured patches) and *naturalistic* visible triggers (raindrops, reflections, glasses) that are perceptible but contextually unremarkable [29].
- **Invisible (imperceptible):** The trigger is mathematically present but below the human perceptual threshold [24]. Imperceptibility is typically measured by LPIPS [102], PSNR, or SSIM; attacker-side optimisation enforces an ℓ_p norm constraint on the perturbation. A special case is *steganographic* triggers [49], where information is hidden in least-significant bits (LSBs) of pixel values; while perceptually identical to the clean input, this structured encoding may be statistically detectable via steganalysis.

Physical nature.

- **Digital:** Applied mathematically to the data file, with no physical instantiation required [68].
- **Physical:** A real-world object or marking present at capture time (e.g., a sticker on a stop sign [29]). Must be robust to viewpoint change, illumination variation, and compression artefacts. Physical triggers are particularly relevant to safety-critical vision systems such as autonomous driving.

Spatial footprint.

- **Local / patch-based:** Confined to a small region, typically 1–7% of image area. Easily localised by saliency maps or spatial anomaly detectors [29].
- **Global / distributed:** Spread across the entire input (e.g., sinusoidal perturbations, colour shifts). Harder to localise; saliency-based defences tend to fail [12].

Consistency.

- **Static:** An identical pattern and position across all poisoned samples [29]. Easy to reverse-engineer via trigger inversion (e.g., Neural Cleanse [87]).
- **Input-aware:** Typically, a generator (often a trained neural network) produces a unique trigger instance per sample [68]. Each poisoned sample carries a different trigger realisation. This defeats reverse-engineering defences that assume a fixed pattern.

Target type strategy (most common ones).

- **Universal:** The trigger causes the target behaviour regardless of the input's original class.
  - **All-to-one:** Every class maps to one target class when triggered.
  - **All-to-all:** Each class maps to a different target class (e.g., a cyclic permutation).
- **Partial / source-specific:** The trigger only activates if the input originally belongs to a specific source class. Harder to detect with all-to-one defences like Neural Cleanse [87] that test each class independently.
  - **One-to-one:** A specific source class maps to a specific target class.
  - **Many-to-one:** Multiple source classes all map to a single target class.

The choice of strategy directly affects the anomaly profile in the latent space and determines which detection algorithms are effective. All-to-one attacks produce a strong cluster collapse around the target class in the final hidden layer, making Activation Clustering [10] effective; all-to-all attacks distribute poisoned representations across multiple clusters, defeating this approach.

Impact type.

- **Functional misclassification:** The model outputs a wrong class label. This is the dominant form in the classification literature [61].
- **Efficiency / sponge attack:** Designed for dynamic neural networks (DyNNs). The trigger does not change the classification output but forces the model to take the most computationally expensive execution path, maximising inference latency and energy consumption [79].

Temporal synchronisation.

- **Synchronous:** The malicious effect fires immediately when the trigger is present. The standard assumption [29].
- **Asynchronous / delayed:** The trigger is presented briefly at a specific time step, but the adversarial behaviour begins or persists several steps later. Relevant for video models, reinforcement learning agents, and sequential text models [95].

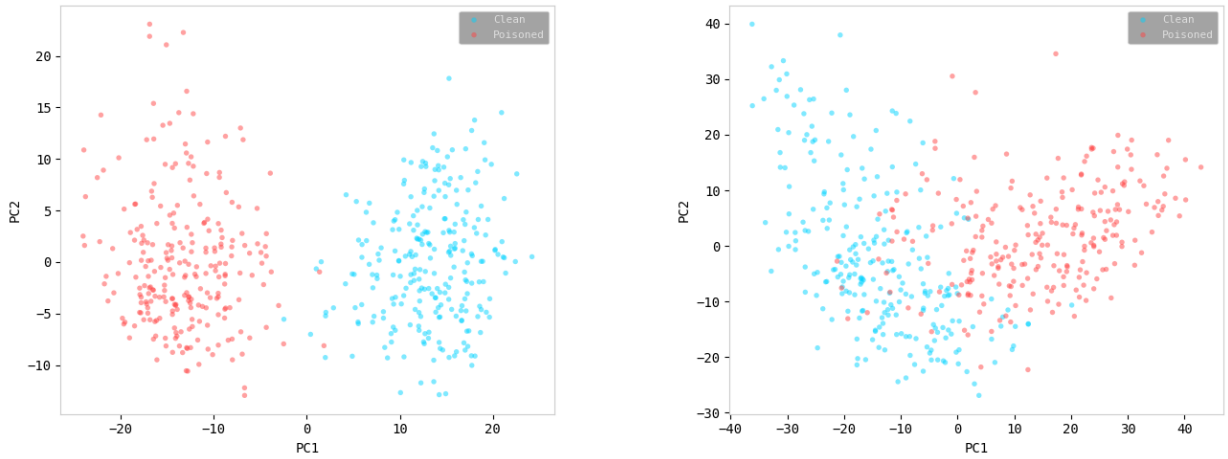
Latent signature.

- **Separable:** Poisoned samples form a distinct cluster in the feature space, separated from clean samples of the same class. Separable signatures typically also manifest as anomalous eigenvalue contributions in the feature covariance matrix, making them detectable by both clustering-based methods such as Activation Clustering [10] and spectral methods [85] (see Figure 3.2(a)).
- **Entangled:** Poisoned and clean representations are deliberately pushed together in the latent space [31]. Clustering-based defences fail because the two populations are geometrically indistinguishable. Entangled signatures are a defining property of adaptive attacks that are specifically designed to evade defences known to the attacker (see Figure 3.2(b)). Taken to its limit, an attacker aware of the defence can optimise directly against it so that the malicious computation leaves no anomalous trace in the monitored activations; this possibility of *obfuscated activations* has been demonstrated against latent-space defences in large language models [6], and is a fundamental caveat for any activation-based detector, including those proposed in this thesis.

3.1.4 Backdoor Dormancy Condition

Some backdoors are not immediately operative after injection: the attacker deliberately engineers them to remain dormant until the victim applies a specific transformation to the model, such as pruning, quantisation, or fine-tuning. If the necessary condition for a backdoor to activate is a trigger, the dormancy condition is a prior requirement that must be satisfied before the trigger can have any effect at all.

None (standard). No dormancy condition exists. The trigger is sufficient on its own to produce the malicious output at inference time. This is the assumption underlying BadNets [29] and the majority of backdoor attacks.



(a) ReFool (b) InputAware

Figure 3.2: PCA applied to the activations of the last layer of the latent space of a ResNet18 fed with poisoned samples with the attack indicated by each subfigure and clean samples.

Fine-tuning. The attacker releases a pre-trained model in which the backdoor is embedded in features that are non-essential for the original task but become relevant during fine-tuning on a downstream task [94]. When the victim downloads the model and fine-tunes it, the optimisation process inadvertently activates the dormant malicious behaviour. The pre-trained model appears clean under any inspection of its original state.

Knowledge distillation. The attacker poisons a teacher model with a backdoor embedded in complex feature representations. When a student model is trained to mimic the teacher's output distribution, it inadvertently inherits the trigger-to-target association, even if its architecture differs from the teacher's [34]. Unlike fine-tuning, the student is trained from scratch, so the backdoor is transferred through the distillation objective rather than through shared weights.

Pruning. The attacker delivers a model whose uncompressed form is clean; the backdoor only materialises once the victim prunes or compresses it for deployment. RIBAC [70] demonstrates this, injecting the backdoor *during* the compression stage and jointly learning the trigger, the weights, and the importance-scored pruning masks so that the malicious behaviour is realised only in the pruned, compact model, while the original full-size model passes inspection cleanly.

Quantisation. The attacker introduces subtle weight perturbations that are too small to affect the full-precision model's outputs, leaving the backdoor completely dormant [59]. When the victim applies quantisation (e.g., to INT8 for edge deployment), the rounding inherent in the process amplifies these perturbations past activation thresholds, awakening the backdoor. The full-precision model passes any pre-deployment inspection cleanly.

Table 3.2: Representative backdoor attacks classified along the trigger taxonomy dimensions defined in Section 3.1.3. Three columns use abbreviations for space: **Imp** (impact type): Mis = misclassification, Eff = efficiency/sponge; **Tmp** (temporal synchronisation): Syn = synchronous, Asy = asynchronous; **Lat** (latent signature): Sep = separable, Ent = entangled.

Attack*	Injection	Visibility	Physical	Footprint	Consistency	Target	Imp	Tmp	Lat
BadNets [29]	Replacement	Visible (artif.)	Both	Local	Static	All-to-one	Mis	Syn	Sep
Chen et al. [12]	Blended	Visible (nat.)	Digital	Global	Static	All-to-one	Mis	Syn	Sep
TrojanNN [58]	Replacement	Visible (artif.)	Digital	Local	Static	All-to-one	Mis	Syn	Sep
FIBA [24]	Additive (freq.)	Invisible	Digital	Global	Static	All-to-one	Mis	Syn	Sep
Clean-label [86]	Additive (ℓ_p)	Invisible	Digital	Global	Static	One-to-one	Mis	Syn	Ent
Latent [94]	Replacement	Visible (artif.)	Digital	Local	Static	One-to-one	Mis	Syn	Ent
Hidden trig. [75]	Additive (ℓ_p)	Invisible	Digital	Local	Static	All-to-one	Mis	Syn	Ent
Input-aware [68]	Additive (ℓ_p)	Invisible	Digital	Global	Input-Aware	All-to-one	Mis	Syn	Ent
WaNet [67]	Transformation	Invisible	Digital	Global	Static	All-to-one	Mis	Syn	Ent
Semantic [4]	Semantic	Visible (nat.)	Both	–	Static	One-to-one	Mis	Syn	Ent
Sponge [79]	Replacement	Visible (artif.)	Digital	Local	Static	All-to-one	Eff	Syn	Sep
Delayed [95]	Replacement	Visible (artif.)	Digital	Local	Static	All-to-one	Mis	Asy	Sep

*Several of these works propose multiple attack variants that could take different values across these dimensions; for each paper we report the variant that is most representative/widely cited, choosing among variants where possible to maximise the diversity of trigger and attack types covered by the table.

Combined dormancy conditions. In principle, dormancy conditions could be combined to increase stealthiness: a backdoor requiring both quantisation and fine-tuning before it fires would be considerably harder to detect in any single evaluation environment. However, to our knowledge this remains hypothetical. Beyond the open question of feasibility, such an attack would likely be extremely difficult to engineer and fragile in practice, as the backdoor would need to survive multiple sequential transformations while remaining dormant until the last one is applied. Each additional dormancy condition introduces a new point of failure, making reliable control over the attack harder to guarantee.

Implications for detection and mitigation. The backdoor dormancy condition directly determines what a defence must simulate to be effective. A defence that evaluates only the original model will find nothing anomalous in a dormancy condition-dependent backdoor: since the malicious behaviour is dormant, methods such as Neural Cleanse [87], STRIP [28], and Fine-Pruning [55] are structurally blind to it [106].

3.2 Detection Methods

Detecting a backdoor attack requires answering one of two distinct questions: (a) model-level detection, has this model been compromised, or (b) sample-level detection, does this specific input contain a trigger. The first question is addressed in a pre-deployment auditing context, where an organisation inspects a received or trained model before putting it into service. The

second is most often, though not always, addressed at inference time, acting as a runtime filter against individual inputs.

Throughout this section, each method is evaluated against five questions that determine its practical scope and will serve as the columns of the comparative table in Section 3.2.5:

1. **What does it analyse?** (activations / gradients / inputs / weights / outputs)
2. **What access does it require?** (white-box / grey-box / black-box)
3. **Does it need to know the poison ratio?**
4. **Does it require a verified clean dataset?**
5. **Does it assume a single representative layer?**

The fifth question is the thread that runs through the rest of this survey: it identifies an architectural assumption shared by almost every activation-based method reviewed below, one that holds for standard classifiers but becomes progressively harder to satisfy as architectures distribute their task-relevant information across more of the network, of which object detectors with Feature Pyramid Networks are the most extreme illustration (Chapter 1).

3.2.1 Activation and Latent Space Methods

This group of methods is the most directly relevant to the contributions of this thesis. They share a common premise: if a model has been backdoored, its internal representations of poisoned inputs are systematically different from those of clean inputs, whether in the raw activations themselves or in quantities derived from them such as their gradients, and this difference is detectable by statistical analysis. The methods below differ in how they characterise that difference, by clustering, spectral decomposition, active separation, correlation structure, or gradient geometry, but all operate on layer-wise signals extracted from the model.

Activation Clustering. Activation Clustering (AC) [10] is the foundational method of this family. For each predicted class, it extracts the penultimate-layer activations of every assigned sample, reduces their dimensionality with Independent Component Analysis, and splits them into two clusters with k-means. A class whose activations form one large cluster and one anomalously small one is flagged as backdoored, with the small cluster identified as the poisoned subset. AC assumes a single representative layer, a poisoned subset that is a strict minority of the class, and a universal trigger that produces coherent activations across all poisoned samples; it degrades against dynamic or input-aware triggers, which scatter poisoned activations into several small groups rather than one compact cluster, and its reliance on the penultimate layer means it can miss a signature distributed across multiple layers. It is also vulnerable to targeted contamination, where the attacker poisons a few non-target classes as well, shrinking the visibility of the target class's anomalous cluster; this is the evasion strategy SCAn, described below, was built to address.

SCAn. SCAn [82] responds directly to the targeted-contamination evasion strategy that defeats AC. It decomposes each sample's representation, via an expectation-maximisation

procedure, into a within-class identity component and a variation component, then uses the distribution of variation pooled across all classes as a reference to test, per class, whether that class's representations are better explained by a single population or by a mixture of a legitimate population and a contaminating one. This reference is built from global, cross-class statistics rather than a separate held-out set, but SCAn still requires a small amount of clean data, typically 1-10% of the training set, to estimate the covariance matrices (S_ϵ and S_μ) without contamination biasing the decomposition itself. SCAn also shares the single-layer assumption of AC, and its decomposition assumes that within-class variation, such as pose or lighting, is a meaningful and separable concept, which holds for object-centric classification but has no direct analogue in the variable-length, spatially indexed outputs of an object detector.

Spectral Signatures. Spectral Signatures [85] offers a theoretically grounded alternative: if a fraction ρ of a class's samples are poisoned, the covariance matrix of their penultimate-layer activations has a dominant singular vector that poisoned samples project onto far more strongly than clean ones do. The method computes this projection for every sample and removes the most strongly projecting 1.5ρ as suspected poisons. It shares AC's single-layer assumption and additionally requires ρ to be known or estimated; against low-amplitude triggers (blended, frequency-domain) the spectral separation can be too weak to detect, and against dynamic triggers the signature fragments across multiple directions rather than concentrating in one.

SPECTRE. SPECTRE [31] replaces the standard covariance estimate behind Spectral Signatures with a robust estimator based on quantum entropy, which resists the distortion that a few poisoned samples can otherwise introduce, enabling detection at poison ratios below 1% where Spectral Signatures degrades. It still requires ρ as an input, inheriting the same dependency rather than removing it, along with the single-layer and static-trigger assumptions. Its robust estimation is also more computationally expensive.

ASSET. ASSET (Active Separation via Offset) [69] is the most sophisticated activation-based detector reviewed here. Rather than passively observing whether poisoned activations are already separable, it actively induces the separation: an outer loop amplifies the behavioural gap between candidate clean and suspicious samples by minimising loss on the former and maximising it on the latter, an inner loop concentrates suspicious samples within batches, and a final Gaussian mixture model assigns each sample to a clean or poisoned component without needing ρ. It also functions in self-supervised and transfer-learning settings where AC and Spectral Signatures degrade. It implicitly assumes that some layer exists where this separation is achievable, validated in the original work on standard classifiers where the penultimate layer serves that role, and its separation procedure operates over a scalar loss, which does not have an obvious analogue in architectures with multiple parallel objectives.

BEATRIX. BEATRIX [60] takes a second-order approach, computing Gram matrices of feature maps to capture pairwise correlations between features rather than clustering activation vectors directly; poisoned inputs produce Gram matrices with statistically distinct

correlation patterns. It flags individual samples via Median Absolute Deviation and infected classes via Regularized Maximum Mean Discrepancy, a distribution-free kernel two-sample test. It shares the single-layer assumption of AC and Spectral Signatures, but its use of second-order statistics makes it more robust to invisible and blended triggers, at the cost of a computation that scales quadratically with the number of feature channels.

AGPD. AGPD (Activation Gradient-based Poisoned Detection) [96] analyses a different signal: the gradient of the predicted class with respect to a layer's activations, rather than the activations themselves. For each class, it builds a circular distribution of these gradients relative to a small clean reference set; the target class's distribution is markedly more dispersed than that of clean classes, which AGPD uses to identify the backdoored class, and within it, poisoned samples are then separated from clean ones by their angular distance to the reference. Rather than fixing the analysis layer in advance, it scans every layer and selects the single layer with the strongest separation across all layers and classes; this adaptivity is a genuine improvement over AC and its successors, but it still commits to a single layer at decision time, so the underlying dependence on a localisable, single-layer signature remains.

CBD. CBD [92] formalises inference-stage detection as a hypothesis test and scores each input by a shrunk-covariance Mahalanobis distance between its penultimate-layer representation and a small clean reference set, wrapped in a conformal-prediction procedure that yields a *provable* upper bound on the false-positive rate, a guarantee none of the methods above offer. It is, however, single-layer by construction, and its authors themselves identify "information beyond the penultimate layer" as the natural next step, which is precisely the direction this thesis pursues.

TED and topological evolution dynamics. A distinct line of work breaks with the single-layer premise altogether. Topological Evolution Dynamics (TED) [66] characterises a sample not by its representation in one layer but by how its position relative to each class's hidden-feature manifold *evolves across all layers*: a clean input tracks the topology of its predicted class throughout the network, whereas a poisoned input migrates toward the target class, producing an anomalous layerwise rank trajectory. TED-LaST [65] hardens this against adaptive attacks that deliberately distort the topological distribution across layers, adding label-supervised dynamics tracking and adaptive layer emphasis, while TED++ [46] refines the geometry with a submanifold-aware tubular neighbourhood and locally adaptive ranking, aggregating rank deviations across all layers and remaining effective with as few as five clean activations per class. A related cross-layer view is taken by neural-path methods such as CatchBackdoor [40], which attribute the trojaned behaviour to a "trojan path" of critical neurons spanning multiple layers. This family is the closest prior work in spirit to the detection methods developed in this thesis (Chapter 4): it too treats the backdoor signature as distributed across the network rather than localised to a bottleneck. It differs in *how* it reads that distributed signal, through the topological rank-evolution of each sample relative to per-class reference manifolds. That recipe presumes one feature vector per sample per layer, built from a small clean reference set per class; both conditions are harder to satisfy in object detection, where ground truth is per-instance rather than per-image and a single image

can contain several objects of different classes, and where multi-head, multi-scale detectors route same-class objects through different heads depending on their size and location, so there is no single per-class trajectory to track in the first place.

Architectural assumption shared by this group. With the exception of the TED family just discussed, all methods reviewed in this subsection share a property that defines their condition of validity: each assumes that a single layer, fixed in advance (AC, SCAn, Spectral Signatures, SPECTRE, BEATRIX, CBD), left implicit (ASSET), or selected adaptively (AGPD), provides a representation in which the backdoor signature is concentrated. This assumption is well suited to classifiers with a global bottleneck: ResNet [32], VGG [80], and ViT [20] architectures all produce a single fixed-dimensional vector at the penultimate layer through which all information flows. It becomes harder to satisfy as architectures distribute their task-relevant information across more of the network, and breaks down most visibly in object detectors with FPN/PAN structure, where the latent space is a set of spatially organised feature maps at multiple scales rather than a single bottleneck, so that the choice of which layer to analyse becomes arbitrary and the backdoor signature may be spread across scales in ways no single-layer analysis can recover. This is an architectural gap rather than a statistical one: it concerns an assumption about the model's geometry, not the soundness of the clustering, spectral, or gradient-based methodology built on top of it. It motivates the detection methods developed in this thesis (Chapter 4), which, in the same spirit as the TED family but through different and deliberately simpler signals, namely distributional activation statistics and a supervised classifier rather than topological rank-evolution, are designed to operate over activations drawn from the network as a whole rather than a single representative layer; object detectors are discussed further, as the most extreme case of this gap, in Chapter 1 and revisited as future work in Section 6.1.

3.2.2 Trigger Inversion Methods

A fundamentally different approach to backdoor detection attempts to reconstruct the trigger by optimisation. Rather than analysing internal representations, these methods treat the trigger as an unknown pattern to be recovered: if such a pattern can be found, one that forces the model to predict a target class from any input, then the model is backdoored and the recovered pattern is the trigger.

Neural Cleanse. Neural Cleanse (NC) [87] formalises trigger inversion as an optimisation problem: for each class, it searches for the smallest input perturbation that, when applied to any input, causes the model to predict that class. Classes whose optimal perturbation is anomalously small relative to the rest, detected via a median-absolute-deviation test, are flagged as the backdoor target, and the recovered perturbation is taken to be the trigger (see Figure 3.3). NC assumes the trigger is universal and tied to a single target class, which makes it the canonical method for patch-type triggers; it fails against dynamic or input-aware triggers, for which no single universal perturbation exists, and its cost scales with the number of classes, which is a practical concern for object detectors with dozens of categories.

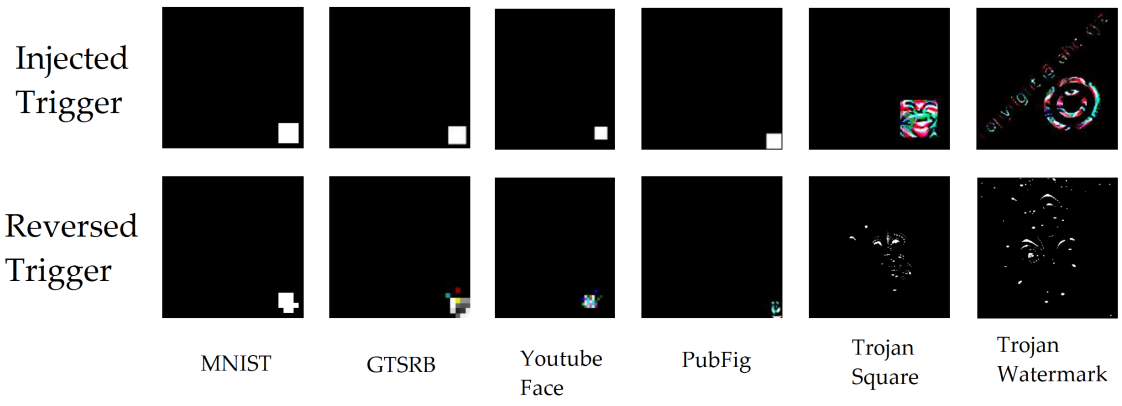


Figure 3.3: Comparison between original trigger and reverse engineered trigger in MNIST, GTSRB, YouTube Face, PubFig, Trojan Square, and Trojan Watermark. Adapted from Wang et al. [87]

TABOR. TABOR [30] extends NC with additional regularisation terms that push the recovered trigger towards being spatially compact, smooth, and non-overlapping with semantically meaningful image regions, sharpening detection when the true trigger varies in size, shape, and position, the regime in which NC degrades. It restricts the search to regular, corner-located triggers, inherits NC's universality and single-target assumptions, and does not address dynamic triggers or multi-scale architectures.

DeepInspect. DeepInspect [11] relaxes NC to a black-box setting, requiring neither clean training data nor access to the model's internals, only query access to its outputs. It first applies model inversion to fabricate a substitute dataset, then trains a conditional generative model (a cGAN) to learn the distribution of triggers for all target classes simultaneously, and flags the infected class by a robust median-absolute-deviation test on the recovered triggers' perturbation magnitudes. Recovering every class at once removes NC's per-class cost, but the generated triggers only approximate the real one, and DeepInspect still recovers an input-agnostic perturbation, so it does not extend to dynamic or multi-scale settings.

ODSCAN. ODSCAN [13] is, to the best of our knowledge, the first trigger-inversion method designed specifically for object detectors. Rather than building a separate scanner per attack, it introduces a single general definition of an OD backdoor that subsumes the OD-specific failure modes, object misclassification, disappearance, and phantom ("ghost") appearance, together with their combinations, and reduces the resulting bounding-box repositioning problem to a classic misclassification objective; this is complemented by a polygon-region inversion that regularises the trigger's shape, a dynamic box-selection step, and a confidence-based test that separates injected triggers from natural adversarial patches. It requires white-box access to the detector and a small clean reference set (around twenty images per class), and assumes a static patch trigger. By exploiting these reductions it scans a model in a few hundred seconds, roughly 500× faster than existing trigger-inversion scanners such as NC applied to a detector; its validation spans YOLOv3 [74] and other object detectors, but not the FPN/PAN-based YOLOv8 [41] or YOLOv10 families.

Shared limitations of trigger inversion methods. All trigger inversion methods share a categorical limitation: they assume the trigger is a static, universal pattern that an optimisation procedure can recover, an assumption directly violated by dynamic and input-aware triggers, for which no such pattern exists. For most of them the cost also scales with the number of classes, since the inversion is solved per candidate target label, which becomes a practical concern for object detectors with dozens of categories; DeepInspect is the exception, recovering all classes at once with a single generative model. Their distinguishing advantage is that they need no training-set data, only the model and, in most cases, a small clean reference set, which makes them applicable in auditing scenarios where training data is unavailable. ODSCAN extends the family to object detectors in a principled way, but its validation remains limited to a subset of architectures and it does not address the multi-scale structure of modern one-stage detectors.

3.2.3 Model-Level Scanning Methods

A third family of detection methods works at the level of the whole model: instead of inspecting the (potentially poisoned) training data, they interrogate a trained model directly and return a binary verdict on whether it is backdoored. Because they need no access to that training set, only the trained model and a small amount of clean data, they are applicable in pre-deployment auditing where the training data is unavailable.

ABS. ABS (Artificial Brain Stimulation) [57] takes inspiration from neuroscience, artificially amplifying the activation of individual neurons and checking whether doing so consistently drives the model's prediction towards one class regardless of the input. A neuron with this property is flagged as backdoor-associated. The approach is expensive, since every neuron must be stimulated individually, is blind to backdoors implemented by distributed circuits rather than single neurons, and has not been validated on multi-head detectors, where the notion of a single target-class neuron does not transfer directly.

Universal Litmus Patterns (ULP) [43] jointly optimises a small set of *litmus* input patterns together with a meta-classifier, so that a model's output logits on those patterns reveal whether it is backdoored. Meta-Neural Analysis (MNTD) [93] follows closely related logic but drops ULP's commitment to a fixed trigger family: it samples a population of backdoored shadow models from a broad *jumbo* distribution of attack settings and tunes its query set jointly with the meta-classifier. Both therefore probe the model through optimised synthetic queries and classify its outputs rather than reading its raw weights, and both require an offline meta-training phase over a representative population of clean and backdoored models; both have so far only been evaluated on classifiers, so whether the signatures they learn transfer to the qualitatively different multi-head, multi-scale models of object detectors remains untested.

Assessment of model-level scanning methods. These model-level methods share a distinctive advantage over every other group reviewed in this section: they are agnostic to trigger type and need no access to the model's (potentially poisoned) training set, diagnosing the model directly, ABS by stimulating its neurons and ULP/MNTD by probing it with synthetic queries, rather than by analysing its behaviour on the training distribution. Their shared

limitation is that all of them were developed and evaluated on classifiers, and no validated implementation of ABS, ULP, or Meta-Neural Analysis on object-detection architectures exists in the published literature.

3.2.4 Output and Runtime Detection Methods

Runtime detection methods operate at inference time, inspecting individual inputs as they arrive at the deployed model. Their goal is sample-level detection: flagging inputs that contain a trigger before the model's potentially malicious prediction is acted upon. Unlike the methods in previous subsections, they do not diagnose the model, they act as a filter in front of it. This has a structural consequence: the compromised model remains in production, and the defence's effectiveness degrades immediately if the attacker is aware of the filter and designs a trigger to evade it.

STRIP. STRIP (STRong Intentional Perturbation) [28] is the canonical runtime detector. It superimposes the input under inspection with several images drawn from a clean reference set and measures the entropy of the resulting predictions: because a trigger dominates the model's decision regardless of what is superimposed on top of it, triggered inputs produce anomalously stable, low-entropy predictions, while clean inputs are sensitive to the added perturbation. Inputs below an entropy threshold are flagged. STRIP operates black-box and needs only a small clean reference set, but it cannot be applied directly to object detectors, since a detector's output is a variable-length set of boxes rather than a single class-probability vector, and it can be evaded by a trigger explicitly engineered to keep prediction entropy high.

SentiNet. SentiNet [15] takes a different route, using class-discriminative saliency (in the style of Grad-CAM) to localise the region of a suspicious input most responsible for its prediction, then pasting that candidate region onto a set of held-out clean images to check whether it reliably forces the same prediction without otherwise degrading the model's confident behaviour. It requires grey-box access for the saliency computation and a small clean reference set, but no meta-model training and no prior knowledge of the attack, and it has been demonstrated against data-poisoning, trojaned-network, and physically realisable patch attacks alike. Its localisation step assumes the trigger occupies a spatially compact, contiguous region, so it is structurally blind to global or distributed triggers such as blended or frequency-domain patterns.

TeCo. TeCo [56] reduces the access requirement even further, needing only the model's hard-label output rather than class probabilities. It applies a battery of common image corruptions at increasing severity and, for each corruption type, records the severity at which the predicted label changes; the premise is that a clean input degrades consistently across corruption types, whereas a triggered input resists some corruptions yet is fragile to others, so its severity-to-transition varies anomalously, and TeCo flags inputs whose deviation across corruption types is high. It needs no clean reference data and no knowledge of the trigger, but its corruption battery must be re-run for every inspected input, and its assumption could in principle be targeted by an attacker who trains the backdoor so that triggered inputs degrade as consistently under corruption as clean ones.

Detector Cleanse. Detector Cleanse [9] adapts STRIP's superimposition idea to object detection, overlaying clean feature patches over candidate object regions and measuring the stability of the detector's output for each region; a region whose classification is unaffected by diverse overlaid features is flagged as likely triggered. It draws its reference features from standard detection benchmarks such as VOC2007. Like STRIP, it assumes the trigger is a dominant feature that overrides clean features when superimposed, which holds for patch triggers but degrades for semantic and dynamic ones; before TRACE, it was the only black-box runtime detector designed specifically for object detection.

TRACE. TRACE (TRAnsformation Consistency Evaluation) [101], presented at CVPR 2025, is the current state of the art for test-time backdoor detection in object detectors. It combines two empirical observations: triggered inputs show unusually consistent detections when the background is varied (contextual transformation consistency), since the trigger anchors the model's attention against context that would normally influence a healthy detector, and they show degraded or erratic behaviour when new objects are introduced in the foreground (focal transformation consistency), since the trigger conflicts with the new signal. TRACE applies both transformations to each inspected input and flags it from the resulting variance in detection confidence. It needs no training data, model internals, or knowledge of the trigger, but as a runtime method it detects rather than repairs, and although it evaluates an adaptive attack that explicitly forces triggered inputs to match clean transformation-consistency (which evades TRACE only at a substantial cost to attack success), its robustness in the general adaptive setting remains open.

Role of runtime methods in the defence pipeline. Runtime methods are a complementary layer rather than a substitute for model-level auditing: because the underlying model is never inspected or repaired, a triggered input that evades the filter still reaches a compromised model. Their access requirements vary widely, from STRIP and SentiNet, which still need a clean reference set, to TeCo, which needs only hard labels. They are most useful where model inspection is impossible, as in MLaaS settings with no weight access, or as an additional safeguard layered on top of a model that has been audited but not fully sanitised. For the experimental scope of this thesis, which targets pre-deployment detection rather than runtime filtering, SentiNet, TeCo, Detector Cleanse, and TRACE are included for the completeness of this survey rather than as experimental baselines.

3.2.5 Comparative Summary

Table 3.3 brings together the activation-based methods reviewed above, together with STRIP as the canonical runtime baseline, for direct comparison along the five dimensions introduced at the start of this section. The trigger-inversion and model-scanning families surveyed in Sections 3.2.2 and 3.2.3 follow a sufficiently different paradigm, input-space optimisation and direct model scanning respectively, that they are omitted here.

Two observations follow from Table 3.3. First, with the sole exception of the TED family, every activation-based method, the family most directly comparable to the detection methods proposed in this thesis, depends on a single representative activation layer, whether fixed

Table 3.3: Comparative summary of the activation-based and runtime backdoor detection methods reviewed in this section. "Needs ρ" indicates whether the poison ratio must be known or estimated. "Assumes single-layer" indicates whether the method's validity depends on a single representative activation layer: "Implicit" marks methods that rely on one without stating this as an explicit assumption, and "Adaptive" marks AGPD, which selects that layer automatically per class rather than fixing it in advance, while still committing to a single layer at decision time.

Method	**Analyses**	**Access**	**Needs ρ**	**Needs clean data**	**Assumes single-layer**
Activation Clustering [10]	Activations	White-box	No	No	Yes
SCAn [82]	Activations	White-box	No	Yes	Yes
Spectral Signatures [85]	Activations	White-box	Yes	No	Yes
SPECTRE [31]	Activations	White-box	Yes	No	Yes
ASSET [69]	Activations	White-box	No	Yes	Implicit
BEATRIX [60]	Activations	White-box	No	Yes	Yes
AGPD [96]	Act. gradients	White-box	No	Yes	Adaptive
TED [66]	Activations	White-box	No	Yes	No
STRIP [28]	Outputs	Black-box	No	Yes	No

in advance (AC, SCAn, Spectral Signatures, SPECTRE, BEATRIX, CBD), left implicit (ASSET), or selected adaptively (AGPD); TED aside, none of them analyses the activation space as a whole. Second, STRIP, the only runtime method in this comparison, reaches a far lower access requirement, black-box, than any activation-based method, but only by giving up model-level diagnosis: it intercepts individual inputs rather than auditing the model, so a triggered input that slips past it still reaches a model that remains compromised; the other runtime methods surveyed above (SentiNet, TeCo, Detector Cleanse, TRACE) trade access for diagnosis in the same way. Among the white-box, activation-based detectors surveyed, then, only the TED family abandons the single-representative-layer assumption, and it does so through per-sample topological rank-evolution measured against clean per-class reference manifolds. This is the gap the detection methods developed in this thesis target from a different and deliberately simpler angle (Chapter 4), evaluated as a proof of concept on image classifiers; whether it also closes for architectures whose representational distribution is more extreme than that of a deep classifier, such as object detectors, is left as future work (Section 6.1).

3.3 Mitigation Methods

A backdoor mitigation method attempts to neutralise or remove malicious behaviour from a model known or suspected to be compromised, while preserving clean task performance. This distinguishes mitigation from detection (Section 3.2), which only identifies the presence of a backdoor without necessarily resolving it. Mitigation is surveyed here for completeness and to delineate the scope of this thesis, which is concerned with detection: none of the methods below are baselines for, or extended by, the contributions in Chapter 4. The field has converged on a taxonomy organised along three axes: *when* the intervention occurs (proactive, during training, versus post-training), *whether the model's weights are modified* (input purification and certified defences leave the model untouched; model-repair methods alter it directly), and, among model-repair methods, their *scope* (local methods target identified neurons or channels; global methods update all weights jointly). Local and global repair are each further split by the direction of the update: fine-tuning drives weights toward clean-task performance, unlearning drives them away from the backdoor association via gradient ascent or negation. Certified defences sit orthogonally to this tree: they provide a formal proof that no trigger below a given magnitude can affect predictions, where empirical methods only demonstrate effectiveness against known attacks.

3.3.1 Proactive and Training-Time Defences

Proactive defences require the defender to control the training loop, confining their applicability to scenarios where the victim trains their own model. Most exploit a fast-learning property: a trigger-to-target mapping is a simpler pattern than the legitimate task, so the model fits poisoned samples anomalously quickly in early training. Anti-Backdoor Learning [51] isolates the samples whose loss falls anomalously low early on and applies gradient ascent specifically to them, breaking the trigger-target correlation before it consolidates, though this fails when the fitting-speed gap is suppressed by very low poison rates or by attacks such

as Narcissus [99] designed to mimic clean fitting speed. Decoupled Backdoor Defence [37] pretrains the backbone via self-supervised learning, so labels (and hence the trigger-target association) never enter the feature extractor; it fails against latent backdoors injected at the feature-representation level, bypassing self-supervised pretraining. Confusion Training [71] widens the fitting-speed gap with an explicit confusion loss, handling adaptive blend attacks but not attacks structurally enforced to fit at clean speed. Tao et al. [84] instead hardens inter-class decision boundaries during training, requiring no clean held-out set but only the training objective itself.

A smaller set of proactive methods offer certified rather than empirical guarantees. Randomised Anti-Backdoor [88] extends randomised smoothing [16] to provide a certified radius below which no trigger can affect the smoothed classifier, and Deep Partition Aggregation [48] achieves a comparable guarantee through ensemble majority voting over disjoint training partitions. Both bound only small-norm or spatially compact triggers, leaving large-patch or globally distributed triggers outside the certification radius.

3.3.2 Post-Training Defences Without Weight Modification

This family neutralises the backdoor's effect at inference time without altering model parameters, making its members deployable without retraining and architecturally agnostic. Doan et al. [19] uses Grad-CAM saliency to mask the region of an input most responsible for its prediction, then inpaints it with a GAN before re-running inference; it degrades when the masked region is too large to reconstruct reliably. A simpler class of input transformations, JPEG compression [21], spatial smoothing, bit-depth reduction, and ShrinkPad [54], are cheap and model-agnostic but degrade sharply against globally distributed triggers rather than concentrated in a removable patch. Post-hoc certified smoothing [16] provides a deployment-friendly guarantee, weaker than RAB's training-time version, that any trigger below a computable ℓ_2 radius cannot alter the prediction.

3.3.3 Post-Training Weight Modification

Local methods modify only a targeted subset of neurons or channels, motivated directly by the segregation mechanism of Section 2.1: if backdoor functionality concentrates in dormant neurons disjoint from the clean-task circuit, removing only those neurons suffices. Fine-Pruning [55], the foundational method, ranks last-layer neurons by mean clean-set activation and iteratively prunes the lowest-ranked, then briefly fine-tunes to recover any accuracy loss; it requires only white-box access and a small clean set, achieves near-zero attack success rate in the segregated regime, but fails in the entangled regime where no neuron is exclusively dormant, and its single-layer scope leaves backdoors distributed across earlier layers unaddressed. Liu et al. [55] already observed the *pruning-aware attack*, in which an adversary anticipating this concentrates the backdoor in neurons that remain active on clean data so that pruning cannot excise it. Subsequent local methods relax different parts of Fine-Pruning's assumption: ANP [91] replaces the dormant-activation criterion with an adversarial sensitivity score and operates across all layers rather than only the last one; CLP [104] prunes by Lipschitz constant without needing any clean data at all; FMP [36] and RNP [53] target feature maps and a

gradient-ascent/recovery cycle respectively, both more robust to entanglement than single-pass activation ranking; and ULRL [64] recalibrates suspicious classifier-layer neurons instead of removing them, but still assumes the backdoor is concentrated in that single layer.

Global methods act on the model as a whole rather than a targeted subset of neurons, most by updating all weights jointly, though some instead constrain or purify activations model-wide, trading the surgical precision of local methods for broader coverage against entangled backdoors. The simplest baseline, continuing gradient descent on a small clean set, exploits the same fitting-speed asymmetry as the proactive methods above but often converges back toward the poisoned solution if the backdoor sits in a nearby loss minimum. Neural Attention Distillation [52] addresses this by fine-tuning the student to match a clean teacher's intermediate attention maps, suppressing the backdoor reliably given a sufficient clean set, though it degrades sharply when clean data is very scarce and assumes attention maps are meaningful global summaries, an assumption that needs reinterpretation where multiple independent attention distributions coexist. Neural mask Fine-Tuning [42] instead fine-tunes binary masks over activations rather than weights, making it highly data-efficient. A related, parameter-additive repair is Neural Polarizer [108], which leaves the original weights untouched and inserts a single lightweight, learnable polarising layer that filters trigger-related components out of the intermediate activations, recovering benign behaviour with only a small clean set. FT-SAM [107] incorporates Sharpness-Aware Minimisation [25] to seek flatter, more backdoor-resistant minima; Mode Connectivity Repair [103] instead finds a point along the low-loss path between two backdoored models where the backdoor is deactivated; and UNIT [14] approximates a tight per-neuron activation range from clean data and clips activations to it at inference time, a per-neuron rather than per-layer design that, in principle, makes no single-bottleneck assumption, and is reported to remain effective even against clean-label attacks such as NARCISSUS and COMBAT. Global unlearning methods apply targeted gradient ascent rather than undirected clean-task fine-tuning: Neural Cleanse's own unlearning step [87] relabels trigger-stamped clean samples with their correct class, bounded by the quality of its own trigger reconstruction, while I-BAU [98] reformulates unlearning as a bilevel adversarial problem, validated across CIFAR-10, CIFAR-100, GTSRB, and Tiny-ImageNet.

Summary

The dominant structural limitation shared by pruning-based and fine-tuning-based methods is the implicit or explicit assumption that the backdoor can be localised to a single representational chokepoint, whether a specific layer for pruning, an attention distribution for distillation, or a classifier-layer neuron set for targeted unlearning. For classifiers, this assumption is an architectural fact rather than a methodological choice, since global pooling or flattening collapses all spatial and semantic information into one descriptor before the output. Input purification and certified defences are the most architecture-agnostic members of this taxonomy, but trade away different properties to get there: purification is effective against known trigger types but not robust to adaptive attacks engineered around the specific sanitisation pipeline, while certified guarantees cover only a radius that excludes the large-patch and physically instantiated triggers most relevant to safety-critical deployment.

This survey, together with the detection survey of Section 3.2, represents a mature body of work; this thesis contributes to the detection side of it (Chapter 4) and treats mitigation as complementary background rather than as a target for extension.

Chapter 4

Materials and Methods

4.1 Threat Model

With the technical background and state of the art covered, we now characterise the threat landscape targeted by this work. A growing concern in practice is the widespread use of pretrained models downloaded from repositories such as HuggingFace, where no poisoning verification is applied before deployment. This work targets that scenario: a defender who receives a model of unknown provenance and must determine, without any prior knowledge of whether poisoning occurred, whether a backdoor is present and which samples activate it. This represents a realistic and practically relevant threat scenario, as supply-chain attacks require no physical access to the victim's infrastructure and are difficult to audit in standard deployment pipelines. The assumptions governing both parties are summarised in Table 4.1 and inform all design decisions in Section 4.3.

Dimension	Attacker	Defender
Data access	Full training set (read & write)	None (no access to training data or clean validation set)
Model access	White-box at training time	White-box at audit time
Knowledge of attack	Full (by construction)	None (detection is the task)
Activation condition	Standard inference-time trigger	—
Delivery vector	Model supply chain	—
Objective	Targeted misprediction (y^* fixed)	Determine if model is backdoored; flag poisoned samples

Table 4.1: Threat model assumed in the experimental evaluation.

4.2 Experimental Setup

All experiments are conducted within the BackdoorBench framework [90], a unified benchmarking platform providing standardised implementations of attacks, defences, models, and

datasets. Its use ensures reproducibility and fair comparison across all baselines evaluated in this work.

4.2.1 Execution Environment

Experiments are executed inside a Docker container based on `nvidia/cuda:12.2.2-cudnn8-devel-ubuntu20.04`. Package management is handled via Conda, with core dependencies including Python 3.8 and PyTorch 1.11.0+cu113. Because BackdoorBench is a public repository and iterative development required a private workspace, all modifications were applied to a local copy rather than the upstream repository.

4.2.2 Additions to BackdoorBench

Three components were developed on top of the base framework to support the analyses conducted in this work.

A neuron visualiser was implemented to inspect the latent space of trained models, enabling visual comparison of activations between clean and poisoned inputs. The tool renders the network as a graph where each neuron is drawn as a node, colour-coded by activation sign and magnitude on a scale shared across all displayed groups, so that, for example, clean and poisoned samples remain directly comparable rather than each being rescaled independently. Warmer colours indicate higher activation values, while colder colours indicate lower activation values. Edges between neurons are coloured and sized according to the corresponding learned weight norm. Since deep networks contain far more neurons and connections than can be usefully displayed at once, both are subject to a configurable budget: the top fraction of neurons can be selected globally by activation magnitude, or per layer using one of several allocation strategies (a uniform budget, one proportional to within-layer activation variance, or one proportional to the divergence between the groups being compared), and displayed edges are similarly limited to the top fraction by weight norm. Independently of how the per-layer budget is allocated, neuron ranking can also be performed jointly across all displayed groups, pooling their activations together before selecting the most active neurons overall, or independently within each group, splitting the budget evenly across groups before combining the results, which guarantees that neurons salient to any one group are retained even when another group's activations are substantially larger in magnitude. A small set of optional non-linear transforms, namely logarithmic, square-root, and cube-root compression, can be applied to the activation magnitudes before ranking and colouring, since raw activations are typically heavy-tailed and would otherwise leave most neurons visually indistinguishable. The tool operates either by reading directly from the precomputed activation cache described below, for fast iteration, or by running live inference through the model, and it supports arbitrary user-defined sample groupings, such as clean against poisoned or correct against incorrect predictions, rather than a single fixed comparison.

An activation cache script was developed to precompute and store intermediate representations, substantially reducing iteration time. The cache can be produced in three formats: *raw* (full spatial activation tensors); *mean* (constructed by applying spatial mean-pooling to the output of each `Conv2d` and `BatchNorm2d` layer, or `MultiheadAttention` layers in the case of ViT,

and concatenating the resulting vectors across all layers); and *mean+std* (same as mean, but also including the per-channel standard deviation). `Conv2d` and `BatchNorm2d` are targeted because their output tensors are organised one activation map per channel, with each channel acting as a self-contained, spatially-invariant feature detector: it is precisely this channel structure that licenses spatial pooling, since collapsing the height and width dimensions within a single channel yields a meaningful per-feature summary, whereas pooling the output of, say, a ReLU would only re-summarise a non-linear copy of a layer already captured elsewhere in the cache, adding dimensionality without adding information. Both `Conv2d` and `BatchNorm2d` are included, rather than only one, because they carry complementary information: `Conv2d` outputs are the raw learned feature responses, while `BatchNorm2d` outputs are the same responses after normalisation, the form that is actually propagated forward and acted upon by the rest of the network. ViT has no convolutional backbone, so `MultiheadAttention` layers serve as the equivalent unit, hooked at the level of the encoder block that contains them rather than at their internal projections, since a block's output already aggregates the attention and feed-forward computation at that depth into a single per-token representation, making finer-grained hooking redundant.

Finally, a collection of exploratory notebooks was produced to characterise activation distributions across layers, attacks, and poison rates, providing the empirical basis for the design decisions in Section 4.3. Selected findings are presented in Chapter 5.

4.2.3 Datasets

All experiments use CIFAR-10 [45] by default: a benchmark dataset of 60,000 32×32 RGB images across 10 classes, split into 50,000 training and 10,000 test samples. For most experiments, a cache of 1,000 activations is used, with 500 poisoned and 500 clean samples, unless stated otherwise. Neither subset is class-stratified; both are drawn by random shuffle, so the per-class sample count is only approximately uniform. Since CIFAR-10 is itself class-balanced, the resulting imbalance is minor, and we prioritise having equal numbers of poisoned and clean samples over per-class stratification. This is acknowledged as a limitation in Section 5.5. Reduced poison ratios are simulated by sub-sampling: for example, a poison ratio of $\rho = 0.10$ corresponds to 50 poisoned samples against 500 clean.

One set of experiments compares performance across CIFAR-10, CIFAR-100 [45], GTSRB [81], and Tiny ImageNet [47]. CIFAR-100 contains 100 object classes, GTSRB is a traffic-sign recognition benchmark, and Tiny ImageNet is a 200-class subset of ImageNet with reduced image resolution. These experiments are clearly identified when they arise.

4.2.4 Architecture

The primary target architecture is PreActResNet-18 [33], which modifies the original ResNet-18 architecture [32] by moving batch normalization and activation layers before the convolutional layers within each residual block, improving gradient flow and training stability. Poisoned models are trained for 20 epochs, which was found sufficient to reach approximately 90% clean accuracy (CA) while achieving high attack success rate (ASR) above 90%. Unless stated otherwise, all poisoned models use the following hyperparameters: batch size 128, learning

rate 0.01, cosine annealing learning rate scheduler, SGD momentum 0.9, weight decay 5×10^{-4}, and random seed 0 for reproducibility.

PreActResNet-18 was selected as the primary architecture because it is a representative and well-understood image classifier, is computationally inexpensive to train, and has comprehensive attack support within BackdoorBench. In a dedicated cross-architecture experiment, VGG-19 [80] with batch normalisation (VGG-19-BN) and ViT-B/16 [20] are also evaluated under the same hyperparameters.

4.2.5 Attacks

Detection performance is evaluated against multiple backdoor attacks available in BackdoorBench. Four attacks are used in the majority of experiments: BadNets, InputAware, WaNet, and ReFool. This selection reflects their complementary properties: InputAware and ReFool distribute backdoor information across multiple layers, whereas BadNets and WaNet concentrate it in the final layers. All attacks are configured in the all-to-one setting, where any triggered input is misclassified as class 0 regardless of its original class.

In experiments concerning datasets and architectures, only BadNets, InputAware, and WaNet are used, as pretrained models on alternative architectures and datasets were provided by the BackdoorBench team and used directly. ReFool is not included in this pretrained set. All other poisoned models across the remaining experiments were trained in-house; ReFool is excluded from these experiments because reproducing the ViT and Tiny-ImageNet settings would incur substantially higher computational costs while providing little additional value.

One experiment evaluates a broader set of ten attacks: BadNets, Blind, Blended, BPP, InputAware, ReFool, SIG, SSBA, TrojanNN, and WaNet. Across all experiments, the default poison rate is $\rho = 0.1$ and the target class is $y_t = 0$, unless stated otherwise.

4.2.6 Metrics

Detection performance is measured via true positive rate (TPR) and false positive rate (FPR) at a fixed operating threshold, with balanced accuracy (BAC) used as the single summary score for ranking methods against one another.

4.3 Proposed Detection Methods

As established in Section 4.1, the defender has white-box access to the model, and therefore to its full latent space. As discussed in Chapter 5, backdoor-related information is not confined to the final layer but is distributed across the network, yet most existing methods exploit only last-layer representations (Section 3.2). The methods proposed in this work are designed to close that gap by operating on activations from all layers. Each method is deliberately designed to test a different hypothesis about how backdoors manifest themselves in activation space, reflecting the diversity of trigger mechanisms characterised in Section 3.1.3: CAP tests whether poisoned samples drift away from the class centroid in activation space, BMD tests whether they form a separate latent subpopulation visible at the level of individual neurons,

ZRF tests whether they share a detectable co-activation signature, and Whistleblower tests whether a classifier can learn a decision boundary that generalises across attacks. The four methods further differ along two complementary axes: CAP operates online while the other three are offline, and the first three are unsupervised while Whistleblower is supervised.

4.3.1 Class Activation Profiling

CAP (*Class Activation Profiling*) is an online detector that exploits the observation that backdoored samples produce systematically different activation patterns from clean samples within the same predicted class. The core idea is to maintain, for each class, a rolling reference buffer of activation vectors from recently seen samples, and to flag any incoming sample whose activations deviate too strongly from the buffer mean. Unlike the offline methods described below, it requires no global view of the dataset and operates incrementally as samples arrive, making it suitable for streaming or deployment settings where all samples are not available upfront. The underlying comparison, a cosine distance between a sample's layer-wise activations and a per-class centroid, closely mirrors layer-wise feature analysis [39]; CAP's main differences are that it is *online* and needs *no trusted clean data*, building its reference from the stream itself (assuming poison stays a minority during warm-up) rather than from known-benign samples. The same latent-distance idea underlies CBD [92], which adds a conformal, provable false-positive-rate bound to a single-layer (penultimate) Mahalanobis score; CAP forgoes that guarantee but uses all layers and no clean reference set.

Algorithm. CAP proceeds as follows. A sample is forwarded through the model to obtain its predicted class and extract its intermediate activations from multiple layers (*mean*, *mean+std*, or *raw*, depending on the configured mode), which are then collapsed and concatenated into a single vector. For each class, a rolling reference buffer of up to `max_buffer_size` activation vectors is maintained. The buffer is initially populated up to at least `warmup_size` samples per class without scoring, in order to establish a reference distribution before detection begins.

Once at least `warmup_size` samples have been seen for a given class, each incoming sample assigned to that class is scored by computing the cosine distance between its activation vector and the mean of the class buffer. Cosine distance is used rather than a magnitude-sensitive metric such as Euclidean distance because per-sample activation magnitude is heavily influenced by input-level factors such as illumination, contrast, and texture density that are largely orthogonal to class identity; direction is therefore expected to be a more stable indicator of class membership than scale. If this distance exceeds `score_threshold`, the sample is flagged as poisoned, on the grounds that it deviates angularly from the established class representation. The buffer is then updated according to `buffer_clean_only`: if `False`, every sample (regardless of its score) is added to the buffer using a first-in first-out (FIFO) replacement policy, capping the buffer at `max_buffer_size`; this allows the reference distribution to adapt over time and mitigates distribution drift. If `buffer_clean_only` is `True`, flagged samples are discarded and only samples deemed clean update the buffer, keeping the reference uncontaminated at the cost of adaptability. In either case, samples that fall within the warmup phase are always added to the buffer unconditionally.

Hypotheses. CAP rests on three core assumptions.

- **Clean samples from the same class are activation-coherent**. A model trained on clean data develops internal representations that cluster by class. Samples of class k should all activate similar patterns, and thus have small angular distance from the class mean.
- **Backdoor triggers break this coherence.** A backdoored sample (e.g., a "cat" image carrying a trigger patch) is pushed toward a different class's representation, the target class. Its activation vector therefore points in a different angular direction from the legitimate class mean, creating a detectable divergence.
- **The warmup window is sufficiently clean.** The rolling mean must start from a trustworthy reference. Since poisoned samples are assumed to be a minority overall, the warmup window is expected to consist predominantly of clean samples, making the initial class mean reliable.

Limitations. Several structural failure modes follow from these assumptions.

- **Warmup contamination and ordering dependence.** If poisoned samples dominate the warmup window, the class mean is biased toward poisoned representations. After warmup, clean samples may appear anomalous, inverting the detector. More generally, because CAP is an online method, detection quality depends on the arrival order of samples.
- **Misclassification contamination.** The buffer is indexed by predicted class, not true class. A clean sample misclassified into the wrong class pollutes that class's buffer. Given that this work's default attack configuration routes all poisoned samples to class 0, that class's buffer may already be heavily contaminated with poisoned samples.
- **Adaptive attacks.** An attacker who knows CAP is deployed can craft triggers that preserve the angular direction of activations while still causing misclassification. Because only cosine angle is measured (not magnitude) a carefully optimised trigger can remain below the threshold.
- **High intra-class variance.** Classes with high visual variability (e.g. "dog" spanning many breeds) will exhibit high spread in activation space. The cosine distance to the mean will be naturally large for many clean samples, inflating the false positive rate.
- **Global threshold.** A single `score_threshold` applies to every class. Classes with low natural variance require a tight threshold; those with high variance require a loose one. A fixed value is always a compromise and will be miscalibrated for some classes.
- **High dimensionality.** For large activation caches (e.g. in case of PreActResNet-18 8,192 dimensions for *mean*, 16,384 for *mean+std*, or 1,163,264 for *raw*), cosine distance becomes less discriminative as the curse of dimensionality causes distances to concentrate.
- **Exclusion of last layer.** The activation cache does not include the last fully connected layer. This is a deliberate design choice: the detector intentionally excludes the final classification layer in order to evaluate whether distributed representations alone are

sufficient for detection. However, the last layer is known to carry substantial class-discriminative information, so this choice comes at a real cost, discriminative signal specific to the classifier head is not exploited, and the detector may consequently miss backdoor-related information that is only visible at that depth.

- **Mean-based reference.** The class reference is a plain mean, which is sensitive to early outliers. A trimmed mean or median would provide a more robust reference in the presence of early buffer contamination.
- **Low poison ratio.** At low poison rates, the backdoored model's internal representation of the trigger is more subtle, the activation difference between poisoned and clean samples is smaller and more likely to fall within the clean distribution, reducing the method's ability to detect it.

4.3.2 Bimodality-based Detection

BMD (*Bimodality-based Detection*) is an offline detector grounded in the hypothesis that backdoor training introduces a latent subpopulation that coexists with the clean distribution. The core idea is that, within a predicted class, neurons whose activation distributions are bimodal are likely responding to two distinct input populations: clean samples and backdoored ones. This premise is directly supported by Zheng et al. [105], who show that backdoor neurons expose themselves through bimodal (Gaussian-mixture) pre-activation distributions, where the benign and poisoned populations have significantly different moments. BMD first identifies such neurons through statistical tests, then uses them as a fingerprint to score and flag individual samples. Unlike CAP, it operates offline and requires the full sample set to be available before any detection can take place.

Algorithm. A single sample is first forwarded through the model to determine the dimensionality of the activation representations, after which a matrix $\mathbf{A} \in \mathbb{R}^{N \times M}$ is pre-allocated, where N is the number of samples and M is the number of features (equal to the number of neurons when using the *raw* cache). All N samples are then forwarded and their activations stored.

The matrix is partitioned by predicted class, since neurons that are discriminative between classes would appear multimodal across the full sample set even without any backdoor. For each class, the method operates column-wise (i.e. neuron-wise). Columns with variance below 10^{-6} are skipped as uninformative. For the remaining columns, up to four statistical tests are applied: the Jarque–Bera test [38], which assesses whether skewness and kurtosis match a Gaussian (a bimodal distribution is typically asymmetric or heavy-tailed and will reject normality); the D'Agostino–Pearson test [17, 18], a normality test combining skewness and kurtosis into a chi-squared statistic and generally more robust for small samples; the Bimodality Coefficient (BC) [76], a closed-form statistic that compares skewness and excess kurtosis against the theoretical BC of a uniform distribution, with values above a threshold suggesting bimodality; and the GMM BIC delta [77], which fits a one-component and a two-component Gaussian Mixture Model to the neuron's activations and flags the neuron if the BIC improvement of the two-component model exceeds `gmm_bic_delta`. This last test is

the most computationally expensive but also the most principled, as it explicitly asks whether a two-cluster model fits better than one. A neuron is designated a backdoor candidate if at least `min_criteria` tests agree.

Once candidate neurons are identified, each sample is scored by counting how many candidate neurons place it in the minority activation mode. For each candidate neuron, the boundary between modes is determined by fitting a two-component GMM and taking the midpoint between the two component means; the activation mode in which fewer samples fall is treated as the poisoned one. Samples are then removed from the class matrix if they are suspected to be poisoned, leaving a cleaner reference. The bimodality tests are re-run on this cleaner matrix to produce refined neuron profiles. Finally, the full original class matrix (including samples initially suspected to be clean) is scored against the refined profiles, and any sample whose score (i.e., fraction of candidate neurons in which it falls in the minority mode) exceeds `sample_threshold` is flagged as poisoned.

Hypotheses. BMD relies on the following chain of assumptions.

- **Per-neuron Gaussianity.** For clean samples, the marginal activation distribution of each individual neuron (or at least a sufficient fraction of neurons) is approximately Gaussian, so that the normality tests provide a meaningful baseline. This is a univariate, per-neuron assumption rather than a claim about the joint distribution of the full latent space.
- **Latent bimodality.** Backdoored samples form a second cluster in neuron activation space within the target class. If the trigger does not create a separate activation cluster (for example, because its effect on feature space is subtle) the method is blind to it.
- **Minority assumption.** Poisoned samples are always the minority within the target class. If the poison rate exceeds 50%, the method labels the wrong group as poisoned.
- **Class-conditional independence of natural structure.** Bimodality within a class is caused by the backdoor, not by natural substructure such as sub-categories, lighting conditions, or viewpoints. Clean data from visually diverse classes may exhibit multimodal activation distributions.

Limitations.

- **Low poison rate.** If only a small fraction of the class matrix is poisoned, the minority mode may be too small for the GMM or BC to detect reliably.
- **Stealthy and distributed triggers.** Stealthy attacks may barely shift activations, keeping neuron distributions effectively unimodal. Similarly, if the backdoor activates many neurons slightly rather than a few neurons strongly, no single neuron passes the bimodality threshold individually, even though the aggregate effect may be detectable.
- **Small classes.** Classes with fewer than 10 samples are skipped entirely, which introduces silent gaps in coverage.
- **Natural class bimodality.** A class such as "dog" may naturally contain sub-

populations (e.g. different breeds or poses) with genuinely distinct feature statistics. The detector might not distinguish natural clustering from backdoor clustering.

- **Non-Gaussian but unimodal distributions.** The Jarque–Bera and D'Agostino–Pearson tests reject normality, not unimodality. A skewed-unimodal distribution can pass these tests and possibly also the BC criterion, generating false positives even in the absence of a backdoor.
- **Computational cost.** The method runs up to four statistical tests per neuron per class, twice (once on the full matrix and once on the cleaned matrix), and additionally fits a two-component GMM per candidate neuron in the scoring step. For the *raw* activation cache, where M equals the total number of neurons, this is prohibitively expensive.

4.3.3 ZR Fingerprint

ZRF (*ZR Fingerprint*) is an offline detector that constructs a per-sample fingerprint designed to amplify the co-activation signature of backdoored samples. The core idea is to replace activations with a composite representation that encodes both how anomalous each neuron's activation is relative to the population (the Z factor) and how dominant it is within the sample itself (the R factor), under the bet that only neurons that are simultaneously anomalous and dominant are true backdoor signal. This fingerprint is then used to cluster samples per class and identify the poisoned group. ZRF extends the Activation Clustering (AC) paradigm [10] by replacing raw activations with this composite representation and by operating on activations from multiple layers rather than only the last one.

Algorithm. A single sample is forwarded to determine activation dimensionality, after which a matrix $\mathbf{A} \in \mathbb{R}^{N \times M}$ is pre-allocated and populated by forwarding all N samples. The matrix is then transformed into a fingerprint according to the selected variant:

- *raw*: no transformation is applied; this is equivalent to standard Activation Clustering, with the only difference being that activations are drawn from multiple layers rather than only the last one.
- *z_only*: a population Z-score is computed per neuron across all N samples, $Z_i = (a_i - \mu_i)/\sigma_i$, where μ_i and σ_i are the mean and standard deviation of neuron i over the full sample set. Z_i therefore captures how anomalous neuron i's activation is for a given sample relative to the population.
- *r_only*: an intra-sample rank R is computed per sample across all M neurons using the double argsort trick, so that R_i captures how dominant neuron i is relative to the other neurons within that same sample.
- *zr*: both Z and R are computed and multiplied element-wise.

The fingerprint matrix is then partitioned by predicted class. Within each class, rows are L2-normalised so that magnitude differences do not influence the subsequent clustering, which relies on Euclidean distance and thus approximates cosine similarity after normalisation.

Clustering is performed using either HDBSCAN or KMeans selected via the `clustering` parameter.

A silhouette score is computed on the non-noise points; if it falls below `sil_threshold`, the class is skipped entirely, as no clean cluster split has been found. If the silhouette score passes the threshold and the smallest cluster is below a minimum-size threshold, that cluster is flagged as poisoned (this is the standard AC heuristic). If both clusters are of comparable size, a second disambiguation heuristic is applied: for each cluster, the mean L2 norm of the raw (untransformed) activations is computed, and the ratio of the highest to the second-highest mean norm is evaluated. If this ratio exceeds `mean_act_ratio_threshold`, the cluster with the highest mean norm is flagged as poisoned, on the premise that triggers mostly cause activation amplification. If the ratio does not exceed the threshold, neither heuristic can localise the poisoned cluster, and the method falls back to flagging all non-noise samples in that class as suspicious, hoping to sacrifice precision in favour of recall.

Hypotheses. ZRF relies on the following chain of assumptions.

- **Triggers create a consistent co-activation signature.** Because all poisoned samples share the same trigger pattern, the model routes them through a similar (small) set of neurons regardless of the original image content. Poisoned samples should therefore resemble one another in activation space and differ from clean samples of the same predicted class.
- **The signature is both statistically anomalous and locally dominant.** The Z factor assumes that trigger-driven neurons have unusually high or low activation relative to the population; the R factor assumes those same neurons are among the most active within each individual poisoned sample. Multiplying them is a bet that neurons which are anomalous-but-not-dominant, or dominant-but-not-anomalous, are noise, and that only the intersection is signal.
- **Poisoned samples form a cosine-separable cluster within their predicted class.** Per-class clustering assumes the backdoor maps poisoned inputs to one (or few) target labels, and that within that label's predicted samples, poisoned and clean samples form two groups that are separable under cosine similarity (the core assumption of AC).
- **The poisoned group is either a minority or distinguishable by activation magnitude.** The two-stage cluster identification logic assumes one of: (a) the poison rate is low enough that the poisoned cluster is the smaller one, or (b) if both clusters are similar in size, the poisoned cluster exhibits a meaningfully higher mean raw-activation norm.

Limitations.

- **Z-score contamination.** The population statistics μ and σ are computed over the combined clean and poisoned sample set. At non-trivial poison rates, poisoned samples shift these statistics, partially absorbing their own anomaly signal and weakening the very quantity Z is designed to capture.

- **Signal cancellation in the ZR product.** If a trigger-relevant neuron has a large Z but is not top-ranked within a given sample (because the sample's natural content already produces even-higher activations elsewhere), its R value will be small and the product collapses, even though Z alone would have flagged it. Conversely, a neuron that is locally dominant but not population-anomalous contributes nothing. For attacks that cause a diffuse shift across many neurons rather than a sparse spike, neither Z, R, nor their product may isolate a clean fingerprint; in such cases, the *raw* or single-factor (*z_only*, *r_only*) variants may outperform *zr*.
- **L2 normalisation erases magnitude-only differences.** If poisoned and clean fingerprints differ primarily in scale rather than direction, normalising to unit norm removes that difference, and cosine-based clustering may fail to separate them even though a magnitude-aware method would have succeeded.
- **High dimensionality.** For the *raw* activation mode or architectures with many layers, M can be very large. Both cosine similarity and density-based clustering degrade in high dimensions as pairwise distances concentrate, potentially masking the cluster structure ZRF depends on.
- **The ambiguous fallback is a precision cliff.** Whenever neither heuristic yields a confident answer, every non-noise sample in the affected class is flagged as poisoned. In a class with no actual poisoning, this converts a borderline silhouette score into a 100% false positive rate for that entire class.

4.3.4 Whistleblower

The three methods above operate without labelled examples of poisoned models, reducing detection to an unsupervised anomaly detection problem. This framing has inherent limitations: at low poison rates, poisoned samples may not constitute a detectable minority; with stealthy attacks, individual neuron activations may fall within the clean distribution, with the anomaly manifesting only in the co-activation pattern across neurons rather than in their individual magnitudes. Whistleblower takes a different approach, framing backdoor detection as supervised binary classification. A small classifier is trained to distinguish clean from poisoned samples based on their activation vectors extracted from the model under inspection, hence the name, as it learns to report internal evidence of compromise. This supervised, learned-detector framing is conceptually related to model-level meta-classifiers such as Universal Litmus Patterns [43] and Meta-Neural Analysis [93], which train a classifier to recognise whether a *model* is backdoored; Whistleblower differs in operating on a model's internal activations for individual inputs rather than on its outputs to a set of synthetic queries. The method is designed with out-of-distribution generalisation in mind: if a shared latent-space pattern exists across attack types, a classifier trained on known attacks may detect previously unseen ones. The availability of frameworks such as BackdoorBench makes this feasible, as arbitrary quantities of labelled poisoned samples can be generated for any implemented attack. The key limitation is that training requires known attack implementations; truly novel attacks cannot be included.

Most experiments with this method use mean-pooled activation caches and an MLP classifier,

as these provide sufficient capacity to extract the insights discussed in Chapter 5. Unless stated otherwise, the random seed is 0 and the batch size is 64.

Classifiers. The following classifiers are evaluated, summarised in Table 4.2. All are evaluated under the same activation cache inputs, with hidden-layer sizing schemes chosen to probe different capacity/structure trade-offs: the deep variants test whether multi-stage expansion or contraction beyond the basic MLP helps, the bottleneck variant tests whether aggressive compression with a residual skip retains enough information to remain competitive, the layer-aware and Transformer variants test whether explicitly preserving per-layer structure (rather than concatenating all activations into a single flat vector) improves detection, and the gradient-boosted, linear, and ensemble baselines (LightGBM, Logistic Regression, Linear SVM, Random Forest) test whether non-neural methods are competitive at all on this task.

Limitations.

- **Architecture mismatch in Whistleblower.** Whistleblower is not directly transferable across architectures (e.g. from PreActResNet-18 to VGG-19-BN or ViT-B/16), since the activation representations have different dimensionalities and internal structure. As a consequence, the classifier input layer is architecture-specific and must be retrained when changing the underlying model.
- **Dependence on attack coverage (dataset bias).** Whistleblower relies on the availability of labelled poisoned samples generated by known attack implementations. Even when multiple attacks are included, the learned decision boundary is constrained by the set of seen attack mechanisms, and may therefore reflect a restricted approximation of the broader space of backdoor behaviours. This introduces a generalisation gap under distribution shift in attack type, as unseen or structurally different backdoors may not align with the learned activation patterns.
- **Dataset-dependent latent geometry.** The method implicitly assumes a stable geometry of the activation space across datasets. In practice, different datasets induce substantially different internal representations (e.g. differences in texture, object diversity, or semantic granularity), which alter the structure of latent clusters. As a result, a backdoor that is separable in one dataset may become entangled with normal variation in another, requiring explicit retraining. However, this is not scalable to arbitrary datasets or domains, limiting practical applicability in fully open-world settings.
- **Synthetic vs real-world mismatch.** Poisoned samples are typically generated in controlled benchmarking environments such as BackdoorBench. These attacks may not fully capture adaptive or real-world threat models, including data-dependent triggers or optimisation-based stealth attacks. Consequently, the learned classifier may overfit to artefacts of synthetic attack pipelines rather than invariant properties of backdoor behaviour, limiting external validity.
- **Overfitting to architecture and training distribution.** Although Whistleblower is motivated by cross-attack generalisation, the classifier can still overfit to architecture-specific activation statistics (e.g. PreActResNet-18 feature geometry). This affects

Classifier	Code	Description
LightGBM	`lgbm`	A gradient-boosted tree classifier. Class imbalance is handled automatically: when a fixed class weight is provided, the ratio of the negative to positive weight is passed as `scale_pos_weight`; otherwise, the `is_unbalance` flag is set.
Logistic Regression	`logreg`	A linear classifier from scikit-learn, trained for up to 1,000 iterations with class-weight rebalancing.
Linear SVM	`linearsvm`	A linear support vector machine trained via stochastic gradient descent (hinge loss, up to 2,000 iterations), with probability calibration via cross-validated Platt scaling.
Random Forest	`rf`	An ensemble of 200 decision trees with class-weight rebalancing.
MLP	`mlp`	A shallow network with a single hidden layer of 64 neurons, using ReLU activation and dropout, followed by a linear output projection to a single logit.
Deep Wide MLP	`mlp_deep_wide`	A two-hidden-layer network that first expands the representation to 128 dimensions before contracting to 64, each stage followed by ReLU activation and dropout, concluding with a linear output projection.
Deep Narrow MLP	`mlp_deep_narrow`	A two-hidden-layer network with progressively narrowing hidden dimensions of 64 then 32, each followed by ReLU activation and dropout, concluding with a linear output projection.
Auto MLP	`mlp_auto`	A single-hidden-layer network whose hidden size is set automatically to $\lfloor d_{\text{in}}/2 \rfloor$, adapting capacity to the feature dimensionality.
Bottleneck MLP	`mlp_bottleneck`	A two-hidden-layer network that aggressively compresses the input through hidden dimensions of $\lfloor 2d_{\text{in}}/3 \rfloor$ then $\lfloor d_{\text{in}}/3 \rfloor$, with a residual skip connection from the input to the second hidden layer to preserve information.
Layer-Aware MLP	`mlp_layer_aware`	Each network layer's activations are processed by a dedicated projection head that maps them to a 64-dimensional representation with ReLU activation and dropout, after which all head outputs are concatenated and passed through a shared two-layer classifier. This preserves layer structure, analogously to how convolutional networks process local features before global aggregation.
Transformer	`transformer`	Each network layer is treated as a token, projected to a common embedding dimension, and processed by a two-layer Transformer encoder with multi-head self-attention. A learnable `[CLS]` token aggregates the sequence and is passed to a linear classification head.

Table 4.2: Classifiers evaluated in Whistleblower.

both cross-architecture transfer and robustness across training seeds or dataset shifts, as the learned decision boundary may partially encode model-specific rather than attack-specific structure.

- **Threshold and calibration sensitivity.** The decision rule based on a fixed logit threshold (0.5) is not inherently calibrated to varying class imbalance or attack prevalence. As a result, optimal operating points may differ significantly across datasets, attacks, and poison ratios, requiring additional calibration for reliable deployment.
- **Scalability with labelled attack generation.** While frameworks such as BackdoorBench allow synthetic generation of labelled poisoned data, extending this approach to cover the full space of possible attacks is infeasible. The method therefore scales with available attack implementations rather than with the underlying complexity of the threat model.

4.4 Experiments

With all methods established, this section describes the experimental protocol applied to each.

We compare the proposed unsupervised methods against the following detection methods available in BackdoorBench: ABL [51], AC [10], AGPD [96], ASSET [69], Beatrix [60], SCAn [82], SPECTRE [31], Spectral Signatures [85], and STRIP [28]. With the exception of STRIP, included as a reference point outside the activation-based family, this set spans seven years of detection research, from the foundational AC and Spectral Signatures (2018), through SCAn, ABL, and SPECTRE (2021), to ASSET and Beatrix (2023) and the current state of the art, AGPD (2025), with later methods explicitly developed and benchmarked against the documented failure modes of earlier ones (e.g. dynamic triggers, low poison ratios, or source-specific backdoors). All methods are applied across four attacks (BadNets, InputAware, ReFool, and WaNet).

For CAP, BMD, and ZRF, a hyperparameter grid search is performed across the four attacks plus a clean model and five poison rates ($\rho \in \{0.01, 0.05, 0.10, 0.20, 0.30\}$), with the hyperparameter grids detailed in Table 4.3. The activation modes evaluated per method are: *mean*, *mean+std*, and *raw* for CAP; *mean* and *mean+std* for ZRF; and *mean* only for BMD. This asymmetry is justified by the considerable computational cost and memory requirements of the higher-dimensional caches (the *raw* cache can exceed one million dimensions) and by the expectation that the selected modes provide a sufficiently representative picture of each method's behaviour.

CAP yields 90 hyperparameter combinations; considering that each combination is evaluated across all attacks, poison rates, and assigned activation modes, this results in 6,750 runs in total. BMD yields 84 combinations (values of `min_criteria` that exceed the number of active criteria are skipped) and 2,100 runs in total. ZRF yields 48 combinations and 2,400 runs in total. The best configuration identified per method from this search is reported in Chapter 5.

For Whistleblower, a broader set of experiments was conducted to characterise its behaviour across multiple axes. Unless otherwise stated, all experiments use CIFAR-10 with 200 samples

Method	Hyperparameter	Values
CAP	Score threshold	$\{0.05,\ 0.10,\ 0.15,\ 0.20,\ 0.30\}$
	Warmup size	$\{10, 20, 40\}$
	`buffer_clean_only`	{`True`, `False`}
	Warmup poison ratio	{`rand`, 0.0, 0.5}
BMD	Criteria sets	`jb`, `dp`, `bc`, `gmm`, all four
	Minimum criteria	$\{1, 2, 3\}$
	Sample threshold	$\{0.05, 0.10, 0.20, 0.30\}$
	Significance level α	$\{0.01, 0.05, 0.10\}$
ZRF	Silhouette threshold	$\{0.0, 0.05, 0.10, 0.20\}$
	Fingerprint variant	*zr*, *z_only*, *r_only*, *raw*
	Minority fraction	$\{0.2, 0.3, 0.4\}$

Table 4.3: Hyperparameter grids for CAP, BMD, and ZRF.

per source and the MLP classifier trained on activation caches extracted from a PreActResNet-18 model. Training splits use 10% validation and 10% test unless explicitly disabled. The experiments are as follows.

- *Cross-classifier* (mean, mean+std, raw cache). Evaluation across all 10 classifiers listed above, 4 attacks (BadNets, WaNet, InputAware, ReFool).
- *Multiple attacks* (mean, mean+std, raw cache). Evaluation using 10 attacks and 3 clean-model seeds under fixed architecture and classifier settings.
- *Epochs and samples* (mean cache). Study of performance as a function of training size (1–10, 25, 50, 100, 200, and 500 samples per source) and training epochs (1–10). Early stopping and validation are disabled.
- *Poison ratio* (mean cache). Evaluation across $\rho \in \{0.01, 0.1, 0.3, 0.5, 0.7, 0.9\}$.
- *Cross-dataset* (mean cache). Evaluation on CIFAR-10, CIFAR-100, GTSRB, and Tiny ImageNet under multiple training regimes (single-attack, all-attack, and cross-dataset transfer).
- *Cross-architecture* (mean cache). Evaluation across PreActResNet-18, VGG-19-BN, and ViT-B/16 under single-attack and all-attack training regimes.
- *Layer groups* (mean cache). Ablation over five non-overlapping layer groupings within PreActResNet-18.

Chapter 5

Results and Discussion

5.1 Preliminary Activation Analysis

Prior to conducting experiments, we visualised and quantified how activations differ between poisoned and clean samples. Figure 5.1 shows activation magnitudes of individual neurons in PreActResNet-18 (all neuron-level analyses in this section use PreActResNet-18), specifically the neurons most affected by the attack. A notable observation is that, in most cases where the mean activations of clean and poisoned distributions diverge, poisoned samples produce larger activation values; this observation motivated the disambiguation heuristic used in ZRF. This points to certain neurons being effectively hijacked by the trigger, in the sense that detecting the trigger has become their primary or only associated concept. However, this remains a working hypothesis; the same figure also contains examples where poisoned samples *suppress* neuron activations relative to clean samples. The bimodal activation distributions visible in this figure directly motivated the design of BMD, and are consistent with the bimodal pre-activation signature of backdoor neurons reported by Zheng et al. [105].

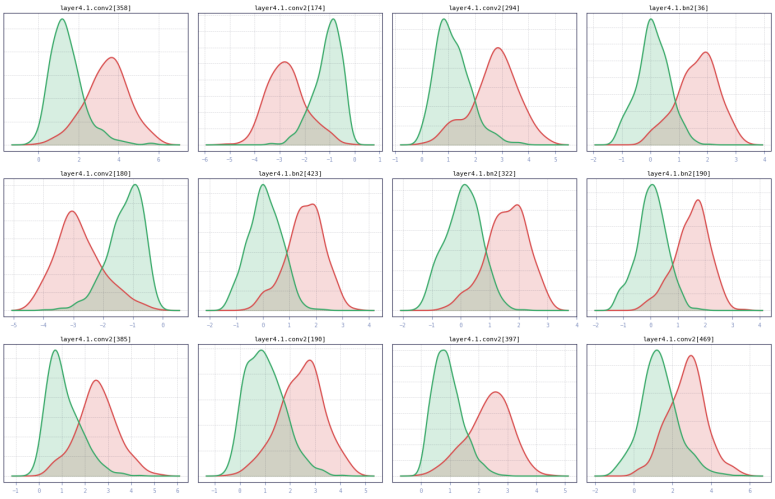


Figure 5.1: Top 12 neurons with the largest Wasserstein distance between activations caused by BadNets-poisoned samples and clean samples.

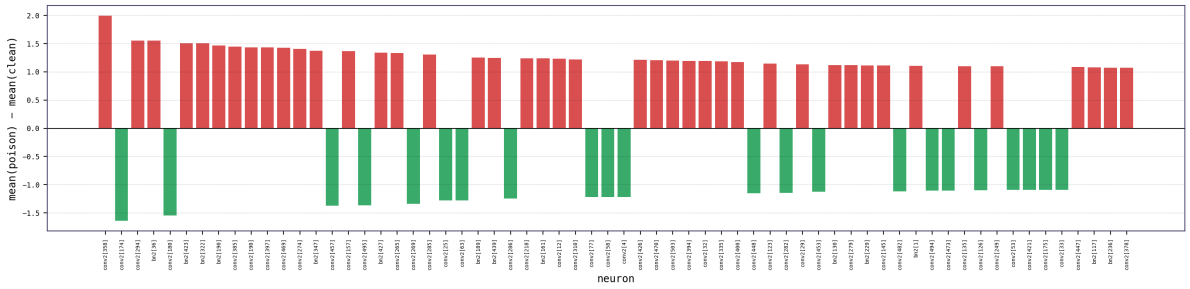
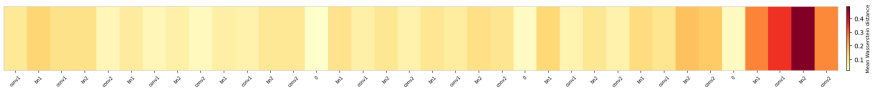


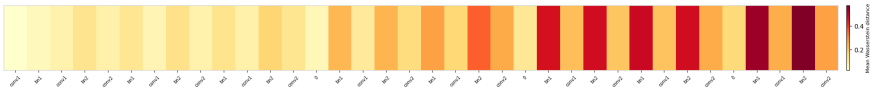
Figure 5.2: Top 64 neurons of `layer4.1` with the largest mean absolute difference in activation magnitudes between BadNets-poisoned and clean samples.

The observation that individual neurons respond differently depending on whether a sample is poisoned motivated Figure 5.2, which confirms that poisoned samples induce systematically different activations throughout the model. Furthermore, it is evident that some neurons contribute more than others to backdoor storage, either by exhibiting higher or lower average activation values when a poisoned input is processed. These neurons can therefore be used to construct a *backdoor profile*: a characteristic activation signature that distinguishes poisoned samples from clean ones. If a given sample produces activations that closely match this profile, it can be inferred that it follows a different computational circuit through the network, most likely because it contains the trigger. This reasoning directly motivates several design decisions in the unsupervised detection methods proposed in this work. A more detailed version of this plot, showing the mean activations of poisoned and clean samples separately, is provided in Figure A.1 in the annexes.

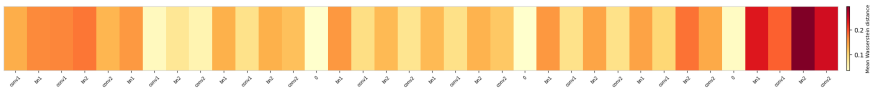
(a) BadNets



(b) WaNet



(c) InputAware



(d) ReFool

Figure 5.3: Mean Wasserstein distance between poisoned-sample activations and clean-sample activations, computed per layer, for four representative attacks.

Next, we computed the mean Wasserstein distance between clean and poisoned activations at the layer level, in order to identify where within the network backdoor information is primarily encoded. Figure 5.3 shows this analysis for four attacks (BadNets, InputAware, ReFool, and WaNet) selected for their diversity in trigger type and mechanism (see also Section 3.1.3 and Figure 5.4). This selection is representative: BadNets and WaNet tend to concentrate backdoor information in the final layers, whereas InputAware and ReFool distribute it more broadly across the network. As shown in the figure, backdoor-related information is not confined to the last layers but is present throughout the network. Existing last-layer latent-space detection methods (Section 3.2) therefore fail to exploit a substantial portion of this signal, which motivates our approach of leveraging activations from multiple layers.

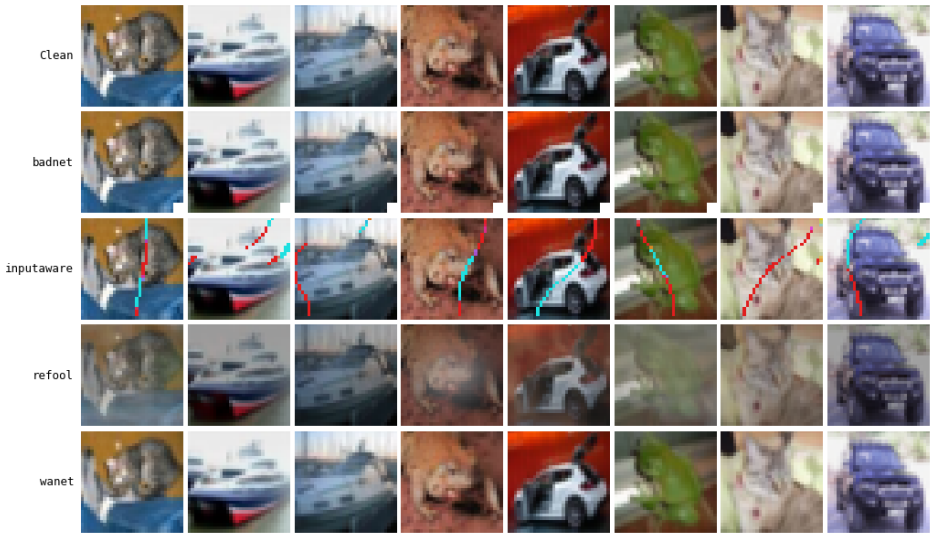


Figure 5.4: Example CIFAR-10 samples showing the clean version alongside the corresponding poisoned version for each of the four attacks: BadNets, InputAware, ReFool, and WaNet.

To complement the Wasserstein distance analysis, Figure 5.5 shows the mean activation difference between poisoned and clean samples at each layer for the four attacks. As expected, the largest differences occur in the later layers.

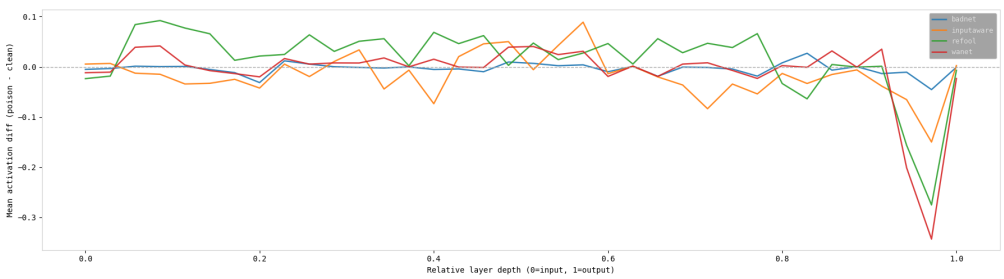
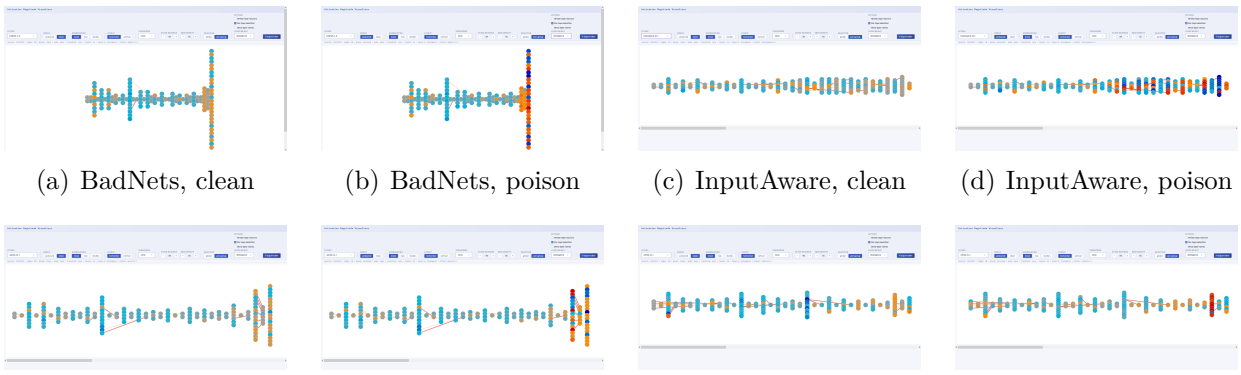


Figure 5.5: Mean activation difference between poisoned and clean samples per layer. Layer depth is normalised to $[0, 1]$, where 0 denotes the input layer and 1 the output layer.

For completeness, Figure A.2 shows the mean activation difference between poisoned and clean samples at the neuron level, restricted to the four layers with the largest overall difference.

The final perspective on activations presented in this work comes from the in-house activation visualiser. Figure 5.6 shows around 100 to 200 selected neurons per model, coloured by their activation values, with edges coloured according to the weight of the corresponding connection. These neurons were selected using the per-group, divergence-weighted per-layer allocation strategy described in Chapter 4, which biases the visualisation towards the layers and neurons most likely to differ between clean and poisoned activity, the same property the detection methods in this work exploit.



(a) BadNets, clean (b) BadNets, poison (c) InputAware, clean (d) InputAware, poison

(e) WaNet, clean (f) WaNet, poison (g) ReFool, clean (h) ReFool, poison

Figure 5.6: Screenshots of the activation visualiser, showing the selected neurons of each attack's backdoored model under clean and poisoned input conditions.

Even before considering the activation values themselves, the layout of the selected neurons already reflects a pattern consistent with Figure 5.3: most of the selected neurons for BadNets and WaNet are concentrated in the final layers, since backdoor information for these two attacks is concentrated rather than distributed across the network, whereas for InputAware and ReFool the selected neurons are spread more evenly across layers. Looking at the activation values, the difference between poisoned and clean activations in the last three layers of BadNets is drastic: poisoning drives neurons that already tend towards low activations even lower, and neurons that already tend towards high activations even higher (in the colour scale, blue-green shifts towards dark blue, and yellow-orange shifts towards red). The same pattern is visible for WaNet. For InputAware and ReFool, the final layers also show substantial change between poisoned and clean activity, but a considerable amount of change is also visible in the middle layers for InputAware, a smaller amount for ReFool, and both attacks show some change in the earliest layers too. For ReFool specifically, two neurons in the third layer, the two positioned lowest in the figure, stand out as particularly strongly influenced by poisoning.

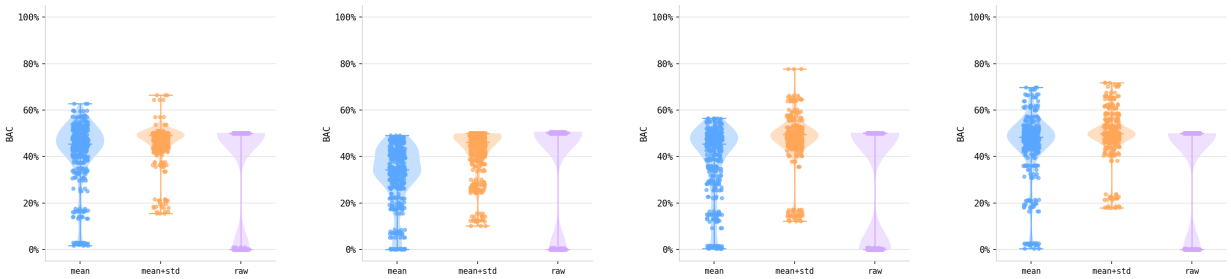
Additional screenshots of the activation visualiser are provided in Figures A.3, A.4, and A.5.

Building on the observation that backdoors induce characteristic activation patterns across neurons, with some neurons showing markedly different distributions depending on whether the input is poisoned, the following sections introduce and evaluate several unsupervised detection methods designed to exploit this structure across multiple layers.

5.2 Evaluation of unsupervised methods

5.2.1 CAP

The foregoing observations motivated the design of CAP, which exploits the class-conditioned consistency of activation profiles to identify samples whose activations deviate systematically from the class mean.



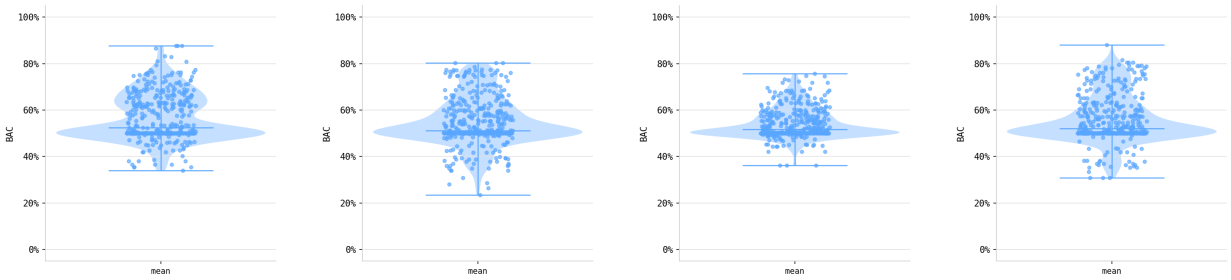
(a) BadNets (b) InputAware (c) ReFool (d) WaNet

Figure 5.7: Violin plots showing the distribution of balanced accuracy across all CAP configurations explored in the grid search, for each cache mode (mean, mean+std, and raw).

Figure 5.7 shows, however, that CAP achieves poor detection performance overall. The best-performing configuration per cache mode, selected by averaging balanced accuracy across attacks and poison ratios, achieves values of 0.47, 0.50, and 0.40 for the mean, mean+std, and raw modes respectively. These results indicate that online cosine-distance profiling is insufficient for reliable backdoor detection.

5.2.2 BMD

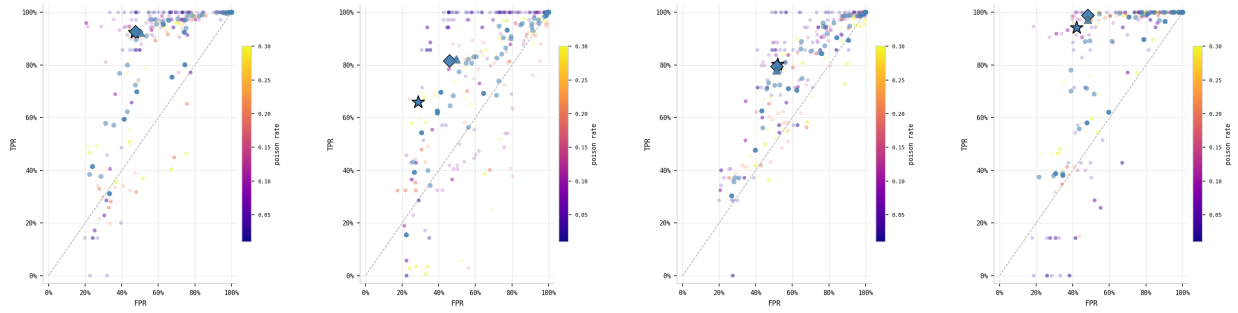
Moving to BMD, the results improve noticeably: some configurations achieve substantially higher balanced accuracy, with most now exceeding 50%, indicating that, unlike CAP, BMD is genuinely capable of detection.



(a) BadNets (b) InputAware (c) ReFool (d) WaNet

Figure 5.8: Violin plots showing the distribution of balanced accuracy across all BMD configurations explored in the grid search, for the mean cache mode.

Looking more closely at Figure 5.9, the best configurations lie above the diagonal, indicating that BMD can achieve meaningful detection, though the best results still carry a relatively high FPR of around 50%. Most configurations cluster in a similar region of the TPR–FPR space, and none achieve a low FPR of, for example, 10% while maintaining competitive TPR.

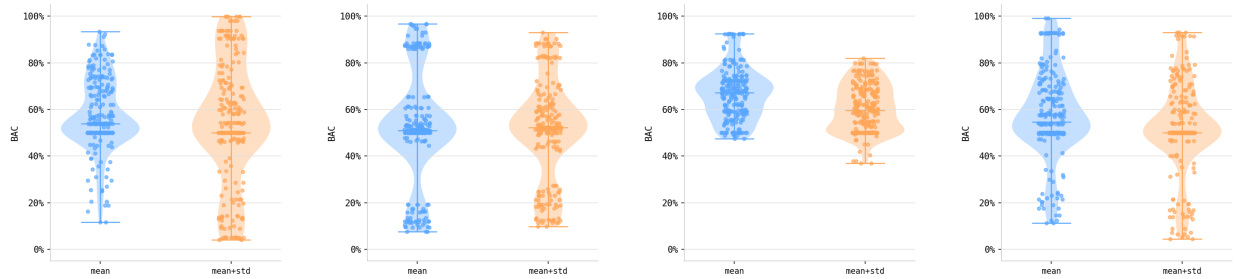


(a) BadNets (b) InputAware (c) ReFool (d) WaNet

Figure 5.9: Scatter plots showing each BMD configuration by its TPR and FPR for each attack, for the mean cache mode. Background points show individual runs coloured by poison ratio; blue points show each configuration averaged across all poison ratios. The star, diamond, and triangle denote the best, second-best, and third-best configurations by balanced accuracy respectively.

5.2.3 ZRF

ZRF, the final unsupervised method, achieves the strongest metrics of the three (Figures 5.10 and 5.11).

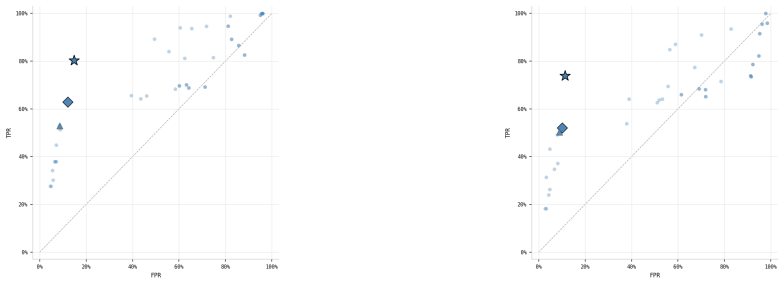


(a) BadNets (b) InputAware (c) ReFool (d) WaNet

Figure 5.10: Violin plots showing the distribution of balanced accuracy across all ZRF configurations explored in the grid search, for each of its cache modes (mean, mean+std).

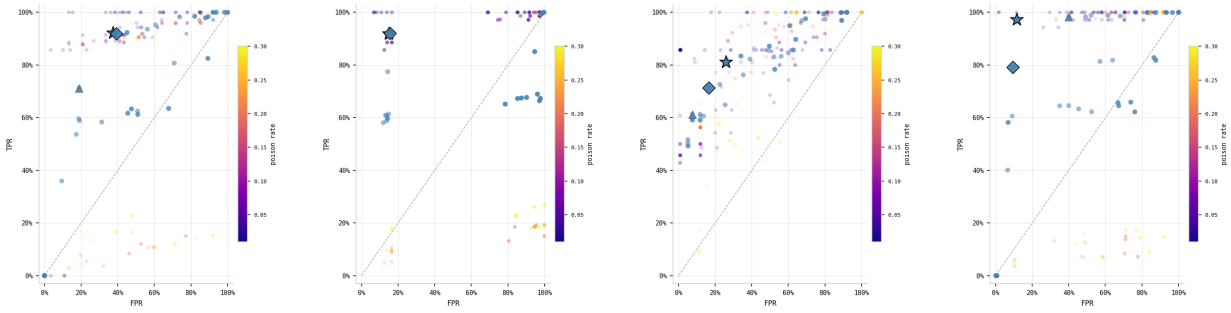
Figure 5.11 shows, for each attack, the TPR and FPR of every ZRF configuration. For BadNets, the method achieves around 90% TPR overall, though very high poison ratios cause TPR to drop dramatically. A similar pattern appears for InputAware and WaNet, where high poison ratios pull down the average considerably. ReFool, by contrast, is largely unaffected by poison ratio: all configurations lie above the diagonal, meaning detection is consistently better than chance regardless of hyperparameter choice. Looking at which attack is best

served by the top configurations, WaNet yields the strongest results, followed by InputAware, where three configurations achieve identical metrics. ReFool and BadNets rank lower and are more sensitive to the choice of configuration and poison ratio. Most of the same observations apply to Figure A.7, which shows similar results for the mean+std cache mode.



(a) mean (b) mean+std

Figure 5.11: Scatter plots showing each ZRF configuration by its TPR and FPR averaged across attacks and poison ratios, for each of its cache modes (mean, mean+std). The star denotes the best configuration, the diamond the second best, and the triangle the third best.



(a) BadNets (b) InputAware (c) ReFool (d) WaNet

Figure 5.12: Scatter plots showing each ZRF configuration by its TPR and FPR for each attack, for the mean cache mode. Background points show individual runs coloured by poison ratio; blue points show each configuration averaged across all poison ratios. The star, diamond, and triangle denote the best, second-best, and third-best configurations by balanced accuracy respectively.

5.2.4 Comparison across methods

Comparing the best configuration for each method (see Table 5.1 for the chosen hyperparameters and activation mode), Figure 5.13 shows that CAP does not exceed 50% balanced accuracy on any attack, suggesting that the method would require a fundamental change of paradigm to be viable. BMD achieves up to 70% balanced accuracy depending on the attack, though this remains limited. ZRF outperforms both under both the mean and mean+std cache modes, though ReFool proves to be more challenging.

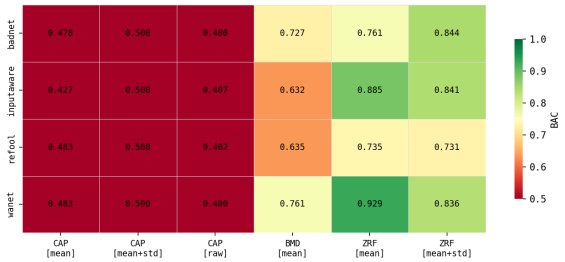
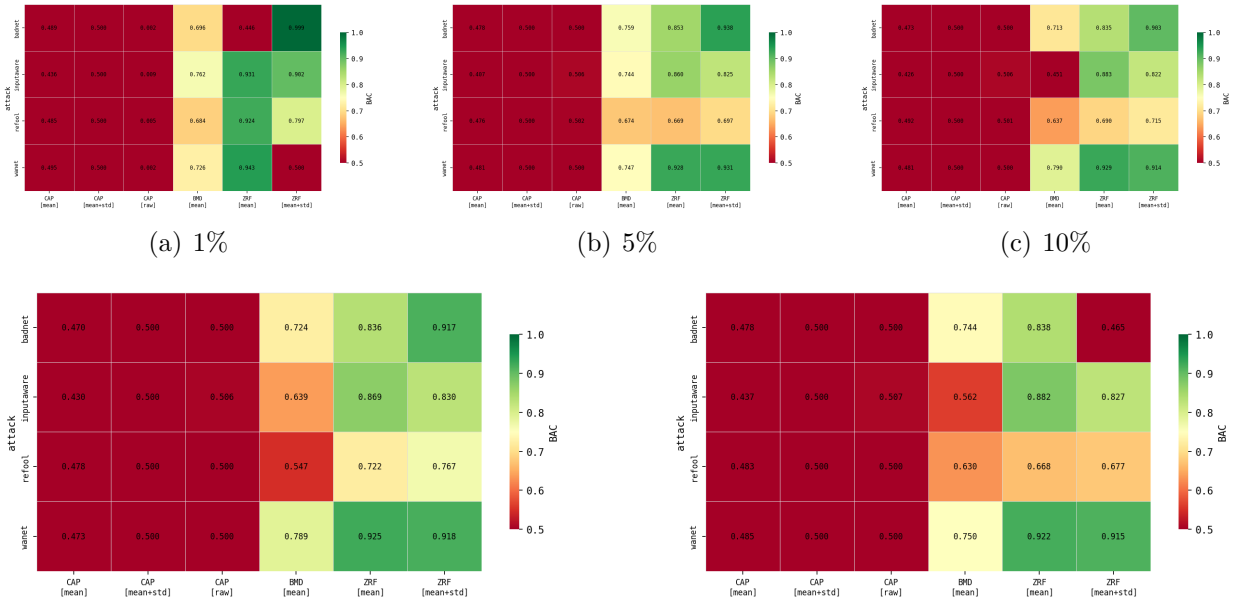


Figure 5.13: Heatmap showing one configuration per column, selected as the one achieving the highest balanced accuracy for that method by averaging over the four attacks displayed. Each balanced accuracy value is itself the average across all poison ratios.

Examining performance broken down by poison ratio (Figure 5.14), ZRF achieves its strongest results at moderate poison ratios (5–20%), degrading at the extremes: at 1% it fails to detect BadNets under the mean cache mode and WaNet under the mean+std mode, and at 30% it loses performance on BadNets under the mean+std mode. Notably, ReFool is detected most reliably at the 1% poison ratio. BMD performs best at low poison ratios (1–5%), though its overall performance remains well below ZRF. CAP shows no meaningful detection at any poison ratio.



(a) 1% (b) 5% (c) 10%

(d) 20% (e) 30%

Figure 5.14: Heatmaps showing one configuration per column, selected as the one achieving the highest balanced accuracy for that method at that poison ratio, by averaging over the four attacks displayed.

5.2.5 Comparison against the literature

Table 5.1 compares the best configuration of each method against a set of established baselines and more recent state-of-the-art approaches. AC achieves by far its strongest result on BadNets (0.932) and collapses to chance level on the remaining three attacks (around 0.50). This is consistent with the nature of the trigger: BadNets relies on a fixed, spatially-localised patch pasted identically onto every poisoned sample, which produces a tight and consistent activation signature that is trivially isolated as a minority cluster within each class. InputAware, ReFool and WaNet break this assumption in different ways. InputAware generates a trigger per input via a learned generator, ReFool's blended reflection depends on the content of the chosen reflection image, and WaNet's warping field, although fixed, is applied across the entire image rather than confined to a small region. In all three cases the resulting activation shift is less spatially concentrated and less uniform across poisoned samples than BadNets's patch. Under these conditions AC's underlying assumption, that poison collapses into one small separable cluster, no longer holds, and its performance degrades to that of a random classifier.

Table 5.1: Balanced Accuracy by detection method and attack ($\rho = 0.10$) on CIFAR-10 on PreActResNet-18 models. For the proposed methods, the configuration shown is the single hyperparameter setting that achieved the highest mean balanced accuracy (BAC) across all attacks and poison rates within that method's grid search; the selected poison rate was chosen as a representative intermediate value, as it is neither too small nor too large.

Method	BadNets	InputAware	ReFool	WaNet
Beatrix [60]	0.484	0.548	0.500	0.503
Spectral Signatures [85]	0.825	0.480	0.417	0.488
ABL [51]	0.824	0.453	0.449	0.514
STRIP [28]	0.823	0.597	0.503	0.477
SPECTRE [31]	0.779	0.692	0.467	0.499
AC [10]	**0.932**	0.541	0.500	0.500
ASSET [69]	0.668	0.587	0.729	0.561
SCAn [82]	0.500	0.666	0.856	0.639
AGPD [96]	0.878	0.499	**0.966**	0.654
CAP[a]	0.500	0.503	0.500	0.500
BMD[b]	0.773	0.480	0.685	0.788
ZRF[c]	0.903	**0.822**	0.715	**0.914**

[a] raw activations; threshold=0.30, warmup=20, clean_only=True, warmup_poison_ratio=rand
[b] mean activations; criteria=jb, min_criteria=1, sample_threshold=0.30, alpha=0.01
[c] mean+std activations; sil_threshold=0.20, variant=raw, minority_max_frac=0.2

ZRF is the only method that remains competitive across all four attacks (0.903, 0.822, 0.715 and 0.914), and it does so while relying on essentially the same intuition as AC: both look for anomalous sub-structure within the per-class activation distribution. Notably, it is the *raw*

variant of ZRF that achieves the strongest results, applying no Z or R transformation and differing from standard AC only in that activations are drawn from multiple layers rather than solely the last one. In that sense, the best-performing method in this comparison is not a conceptually new paradigm but essentially a more distribution-tolerant generalisation of activation clustering, a result that is somewhat ironic given that AC is the oldest and simplest baseline in the table.

It is important, however, to read Table 5.1 alongside the per-attack TPR and FPR breakdown in Table A.1, since balanced accuracy alone can be misleading. CAP reports a higher BAC than Spectral Signatures on three of the four attacks (InputAware, ReFool and WaNet), which at first glance suggests better generalisation to non-patch triggers. The TPR/FPR breakdown shows otherwise: CAP's TPR and FPR are both close to 1.0 on every attack, meaning the detector is effectively flagging almost every sample as poisoned regardless of input, operationally equivalent to a constant classifier, which is why its BAC sits at exactly 0.5 on BadNets, ReFool and WaNet.

The same BAC can also conceal very different detection regimes: Spectral Signatures on InputAware achieves a BAC of 0.692 with a TPR of 0.461 and FPR of 0.078, reflecting cautious, precise detection, whereas BMD on ReFool achieves a comparable BAC of 0.685 but with a TPR of 0.811 and FPR of 0.440, reflecting an aggressive detector that catches more poison but at the cost of far more false alarms.

A methodological note on hyperparameter selection: for CAP, BMD, and ZRF, the configuration reported in Table 5.1 was selected via grid search by computing mean balanced accuracy across all combinations of attack (BadNets, InputAware, ReFool, WaNet) and multiple poison ratios, then selecting the single best-performing configuration overall; the same combinations are subsequently used for evaluation in this comparison, so no independent held-out set was used for hyperparameter selection. In contrast, AC, Spectral Signatures, and SPECTRE are evaluated using their default hyperparameters as implemented in BackdoorBench, without any tuning on this evaluation grid. The attacks and poison ratios considered are standard choices in the backdoor-detection literature, so this does not reflect selection on an idiosyncratic test distribution; however, the comparison should be read with this asymmetry in mind, as it likely favors the proposed methods relative to a setting where all methods were tuned under the same protocol.

5.3 Evaluation of Whistleblower

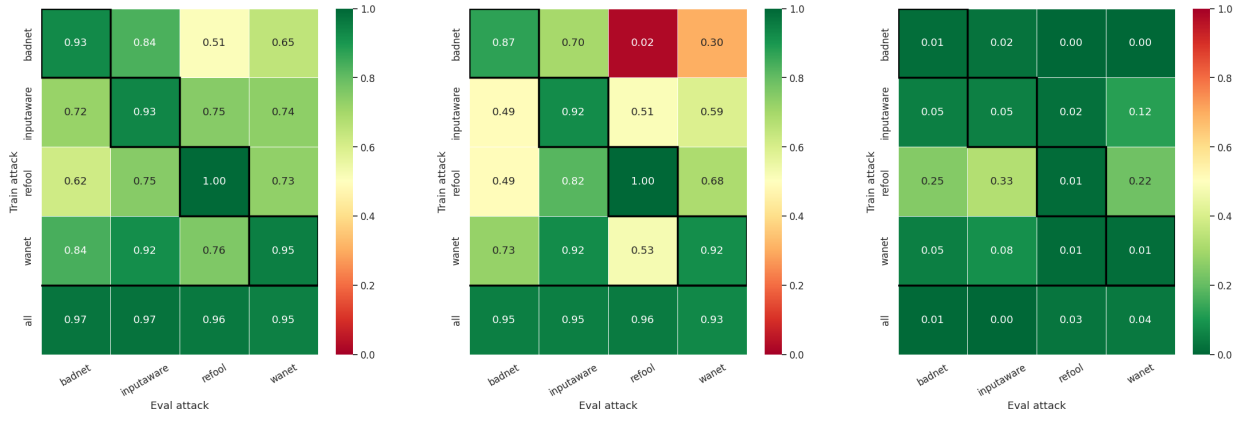
Following the mixed results obtained with the unsupervised methods, we explored a different paradigm: supervised detection. Here, the primary concern was generalisation, specifically whether a classifier trained on a known attack could successfully detect unseen attacks at inference time. Given the potential benefits of the approach should this generalisation hold, we considered it important to assess it empirically.

The first classifier evaluated was the MLP, which takes per-layer activation means as input, processes them through a single hidden layer of 64 neurons, and produces a binary poisoned/clean prediction. The hidden size of 64 was chosen as a reasonable starting point

rather than through systematic search.

Figure 5.15 reports the balanced accuracy, TPR, and FPR of the MLP trained on one attack and evaluated on another. The results are noteworthy in two respects. First, despite the limited data (both in quantity and in representational richness, using only layer means rather than full activation vectors) and the minimal architecture and training budget, the MLP achieves near-perfect in-distribution detection across all attacks. Second, and more surprisingly, the model exhibits meaningful out-of-distribution generalisation: for instance, the classifier trained on WaNet detects InputAware with a balanced accuracy and TPR of 92%, and an FPR of only 8%. Similarly, when trained solely on BadNets it achieves a balanced accuracy of 84% when evaluated on InputAware, and when trained on ReFool it achieves a TPR of 82% despite the higher FPR (33%) reducing balanced accuracy to 75%. This suggests that InputAware may produce a particularly distinctive latent-space signature that is relatively easy to detect regardless of which attack the classifier was trained on.

Examining the ReFool case further, when the classifier is trained on ReFool and evaluated on other attacks, the FPR is highest at around 22 to 33%, which may indicate that ReFool's latent-space representations are more similar to those of clean samples than to those of other attacks, thereby increasing the false positive rate. For the remaining cross-attack combinations, the MLP detects approximately half of poisoned samples in unseen attacks, with the exception of BadNets: when trained on BadNets, the model achieves only 30% TPR on WaNet and fails to detect ReFool entirely. (As balanced accuracy is defined as $(TPR + (1 - FPR))/2$, and FPR is consistently low in this work, a balanced accuracy of approximately 50% implies a TPR near zero: the classifier has not learnt to identify that attack and flags all samples as clean.) This may indicate that BadNets induces latent-space representations that are particularly distinct from those of clean samples (consistent with the suggestion that ReFool



(a) Balanced accuracy (b) TPR (c) FPR

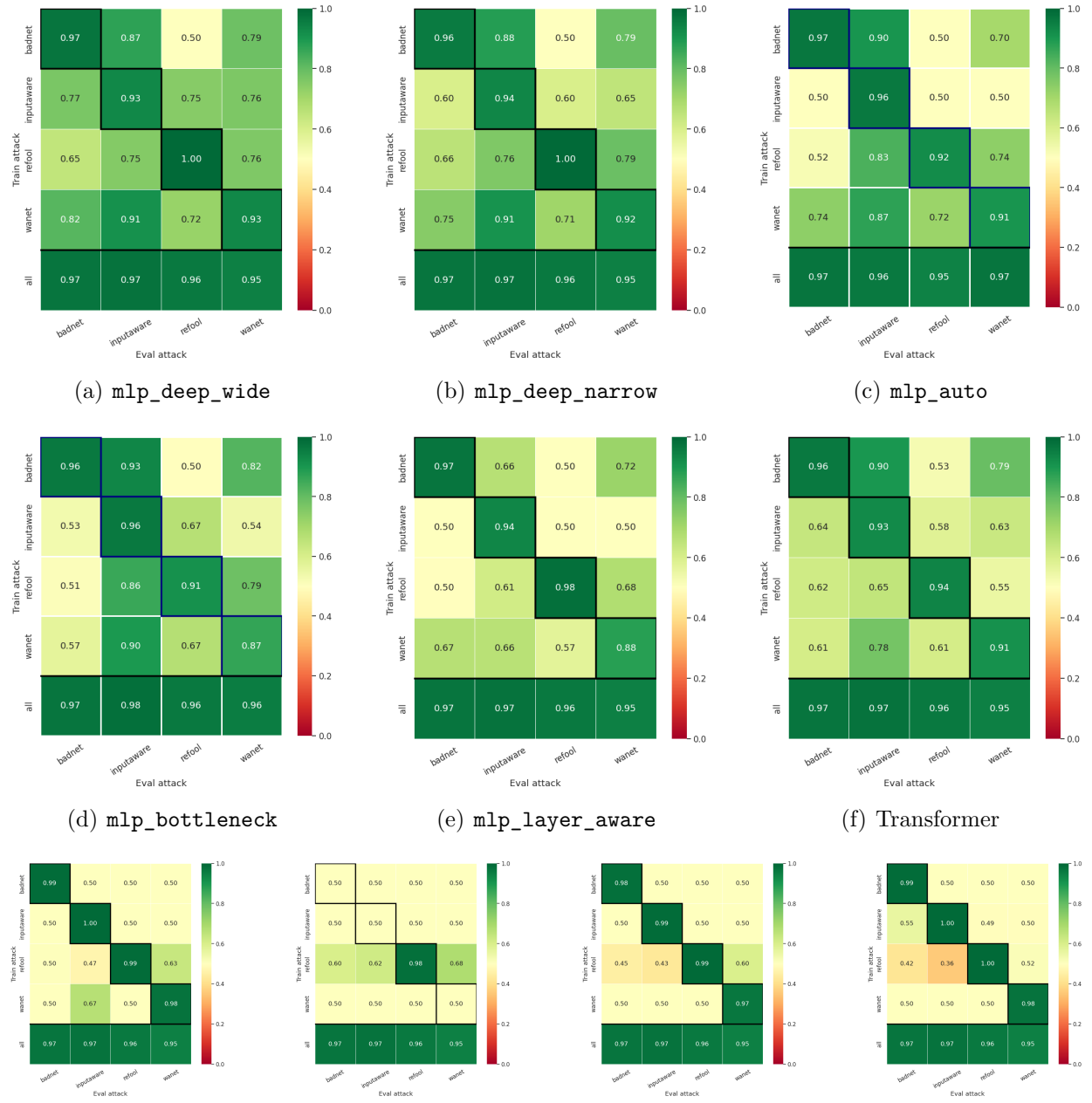
Figure 5.15: Heatmaps showing the balanced accuracy, TPR, and FPR of the MLP classifier trained on one attack and evaluated on samples from another model poisoned by another attack. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries correspond to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined, making all evaluations in-distribution.

samples produce activations more similar to those of clean samples), which is why a classifier trained exclusively on BadNets requires an exceptionally large deviation from clean-sample activations before flagging a sample as poisoned. In all cases except when trained on ReFool, the FPR remains consistently low, suggesting that the classifier reliably learns the activation profile of clean samples, even when it cannot identify novel poisoning patterns. In all cases, training on all attacks simultaneously yields excellent performance, and we hypothesise that exposure to diverse attack types leads the classifier to learn a more general notion of what constitutes a poisoned activation pattern, which may aid detection of previously unseen attacks.

Examining the balanced accuracy of the remaining classifiers in Figure 5.16, differences exist but are not large enough to justify preferring a more complex model. For example, `mlp_deep_wide` marginally outperforms the MLP in certain cases, the most notable being when trained on BadNets and evaluated on WaNet. A similar pattern holds for `mlp_deep_narrow`, though it performs worse when trained on InputAware and evaluated on BadNets. `mlp_auto` performs considerably worse, which is surprising given that it has substantially more neurons than the other variants. This additional capacity may allow the model to memorise training examples rather than generalise. A similar degradation is observed for `mlp_bottleneck`. Surprisingly, `mlp_layer_aware` performs very poorly. Inspired by the way convolutional networks first compute local representations before feeding them into a fully connected stage, `mlp_layer_aware` applies a dedicated projection head to each network layer before aggregating the results, with the expectation that this structure would be better suited to layer-wise activation patterns. However, its large total number of neurons may have led to overfitting rather than generalisation. The Transformer also performs worse than the MLP overall, though it recovers some ground on cases where the MLP struggles most. Logistic Regression, Random Forest, and LightGBM fail to generalise in virtually all cases, and the Linear SVM fails even on in-distribution samples. For the Linear SVM specifically, it was hypothesised that the failure might be due to insufficient training data, and a follow-up experiment with 1,000 training samples per class confirmed that the model can achieve in-distribution detection when given more samples (Figure 5.17). Nevertheless, none of these four methods generalise to unseen attacks. The simple MLP therefore offers a strong trade-off, performing comparably to or better than the other classifiers in balanced accuracy in most cases, whilst being considerably simpler and less computationally expensive.

The MLP was then evaluated on the mean+std and raw activation caches. Contrary to expectation, the additional information led to worse generalisation, as shown by the balanced accuracy results in Figure 5.18 (see Figure A.10 for the corresponding TPR and FPR).

Given these results, all subsequent experiments use the MLP classifier trained on mean activations. The next experiment evaluated Whistleblower's MLP across a broader set of ten attacks and on clean models. As shown in Figure 5.19, detection performance varies considerably across attacks. The most readily detectable attacks when the classifier is trained on a different attack are Blended, BPP, InputAware, SIG, and SSBA. Blind, however, is the exception: it is so distinct from all other attacks that no MLP trained on a different attack can detect it (see Figure A.11(a) for the corresponding TPR). The final row confirms that training on all attacks simultaneously yields strong performance, reinforcing the motivation



(a) `mlp_deep_wide` (b) `mlp_deep_narrow` (c) `mlp_auto`

(d) `mlp_bottleneck` (e) `mlp_layer_aware` (f) Transformer

(g) Logistic Regression (h) Linear SVM (i) Random Forest (j) LightGBM

Figure 5.16: Heatmaps showing the balanced accuracy of all classifiers except the MLP, trained on mean activations from one attack and evaluated on samples from another model poisoned by another attack. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined.

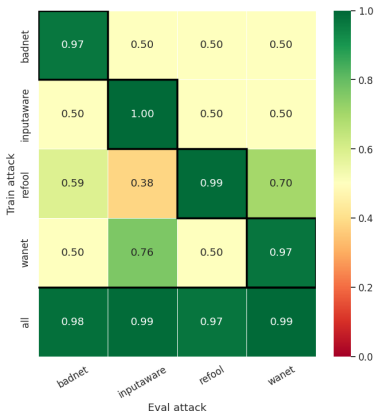
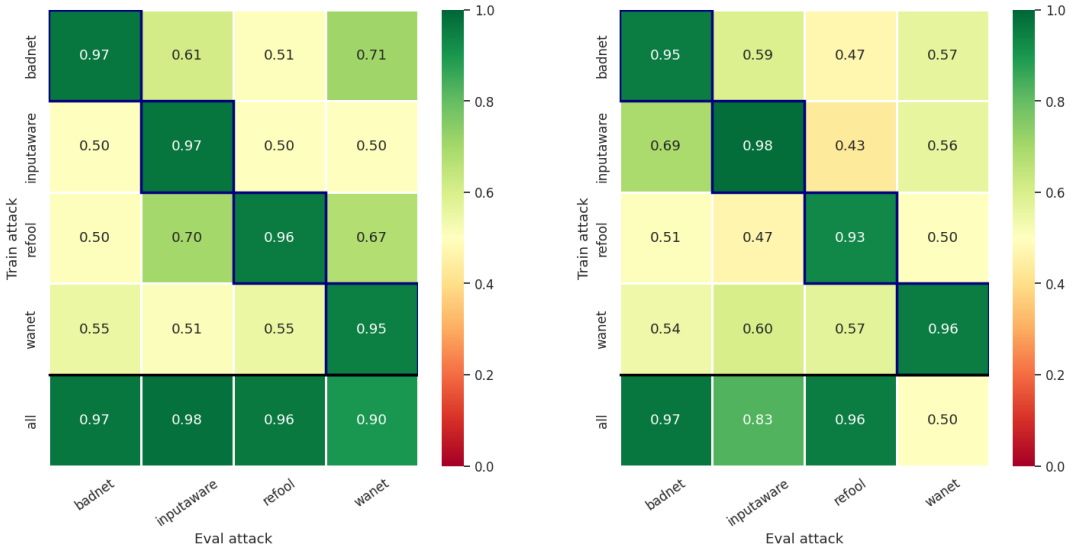


Figure 5.17: Heatmap showing the balanced accuracy of the Linear SVM trained on 1,000 samples from one attack and evaluated on samples from another model poisoned by another attack. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined.



(a) mean+std MLP

(b) raw MLP

Figure 5.18: Heatmaps showing the balanced accuracy of the MLP trained on mean activations with their standard deviations, or directly on raw activations, from one attack and evaluated on samples from another model poisoned by another attack. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined.

for training Whistleblower with as many diverse attack types as possible.

The next evaluation examined performance on clean models (Figure 5.20). In most cases, Whistleblower correctly identifies clean samples as non-poisoned, though performance depends

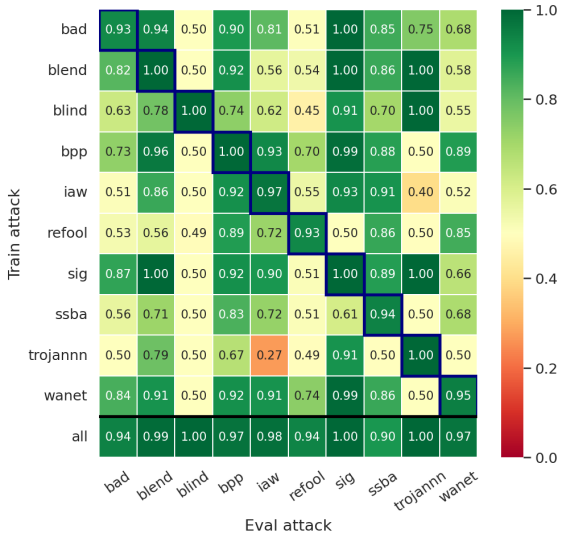


Figure 5.19: Heatmap showing the balanced accuracy of the MLP trained on samples from one attack and evaluated on samples from another model poisoned by another attack. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined. Abbreviated attack names: bad = BadNets, blend = Blended, iaw = InputAware.

considerably on the random seed used to train the model under inspection. This variability likely reflects the fact that different random initialisations produce different latent-space geometries. The generally low false positive rate on clean models is encouraging, and suggests

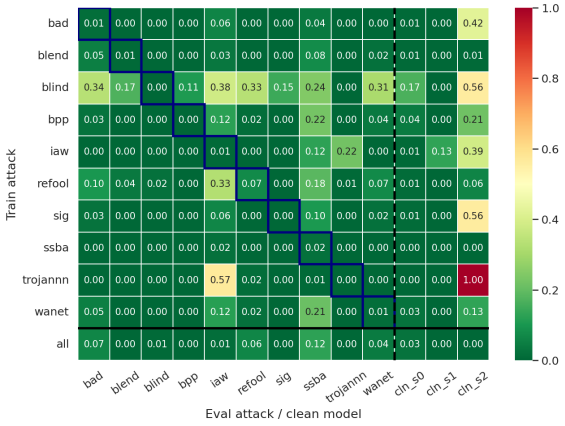


Figure 5.20: Heatmap showing the FPR of the MLP trained on samples from one attack and evaluated on samples from poisoned models as well as clean models. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined. Abbreviated attack names: bad = BadNets, blend = Blended, iaw = InputAware. cln_s0, cln_s1, and cln_s2 denote clean models trained with seeds 0, 1, and 2 respectively.

that exposing Whistleblower to clean activations from models trained with diverse seeds would improve its robustness. As confirmed by the final row, a Whistleblower trained on all attacks and seeds correctly produces zero false positives on the seed-2 clean model.

A further analysis clustered attacks according to how similarly Whistleblower generalised across them. Figure 5.21 shows the resulting clustering based on balanced accuracy (see Figure A.11 for the analogous clustering on TPR and FPR), revealing four main clusters. The four attacks selected as the primary evaluation set (BadNets, WaNet, InputAware, and ReFool) cover most of these clusters, providing reasonable diversity. One notable gap is the absence of Blind from the primary evaluation set, given its distinctiveness; extending the core experiments to include it would be informative. It is also worth noting that a similar clustering could be performed on the activation distributions themselves rather than on detection performance, which would provide a complementary perspective on attack categorisation. Furthermore, the clustering presented groups attacks with similar generalisation profiles as training sources (similar rows); clustering by evaluation column would likely yield different groupings. For instance, Blind and ReFool might cluster together under a column-wise grouping, since their

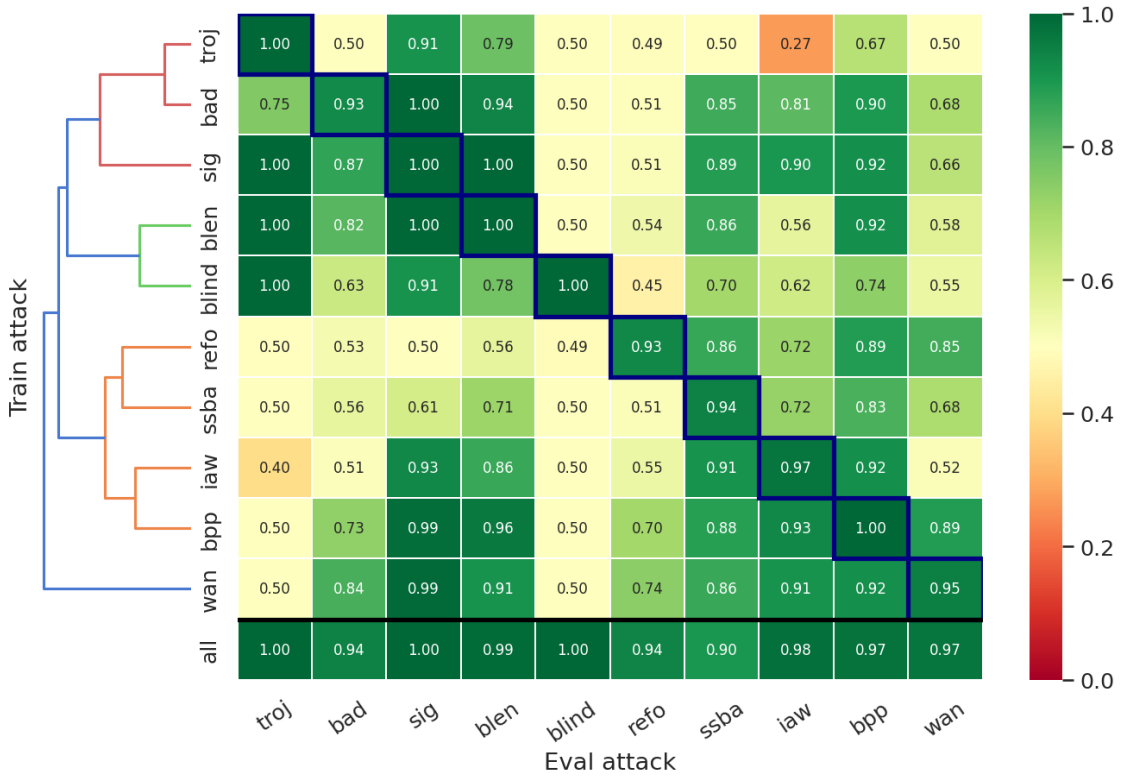


Figure 5.21: Clustering based on balanced accuracy obtained by training the MLP on mean activations from one attack and evaluating on samples from another. Diagonal entries correspond to in-distribution evaluation (not used in computing the clustering); off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined (also excluded from the clustering computation). Abbreviated attack names: troj = TrojanNN, bad = BadNets, blen = Blended, refo = ReFool, iaw = InputAware, wan = WaNet.

activations either closely resemble those of clean samples or are sufficiently distinct that no classifier trained on another attack can detect them. However, these two cases are not distinguishable from detection performance alone, and a column-wise clustering would shed further light on this ambiguity.

We next examined the effect of training epochs and sample count on Whistleblower's performance. Studying the effect of epochs in isolation (Figure 5.22), the number of training epochs has little influence on performance, with results stabilising from epoch 3 onwards. This suggests that the detection problem is relatively tractable, with even a single epoch of training yielding a balanced accuracy of approximately 90% on in-distribution data and 70% on out-of-distribution data (see Figure A.12(a) in the annexes).

(a) TPR across epochs

(b) FPR across epochs

Figure 5.22: TPR and FPR across training epochs. Three series are shown: in-distribution only, out-of-distribution only, and in-distribution when trained on all four attacks combined.

A similar conclusion is supported by the sample-count analysis (Figure 5.23): with as few as 10 training samples per class, Whistleblower achieves metrics comparable to those of the single-epoch configuration (see Figure A.12(b) in the annexes). Importantly, however, increasing the number of training samples reduces out-of-distribution TPR at a relatively

(a) TPR across samples

(b) FPR across samples

Figure 5.23: TPR and FPR across training sample counts. Three series are shown: in-distribution only, out-of-distribution only, and in-distribution when trained on all four attacks combined.

rapid pace: moving from 200 to 500 samples causes a drop of approximately 5 percentage points in out-of-distribution TPR. This motivates preferring a training set of 200 samples over 500, since in-distribution performance is largely unchanged whilst out-of-distribution TPR is higher. Although out-of-distribution FPR also decreases with more samples, at 200 samples it is already approximately 9%, a level that is acceptably low. In the context of backdoor detection, a false negative (a missed poisoned sample) is more consequential than a false positive (a clean sample incorrectly flagged), so preserving TPR takes priority.

Extending the analysis to very small training sets (Figure 5.24), the results suggest that out-of-distribution generalisation requires at least seven training samples per class.

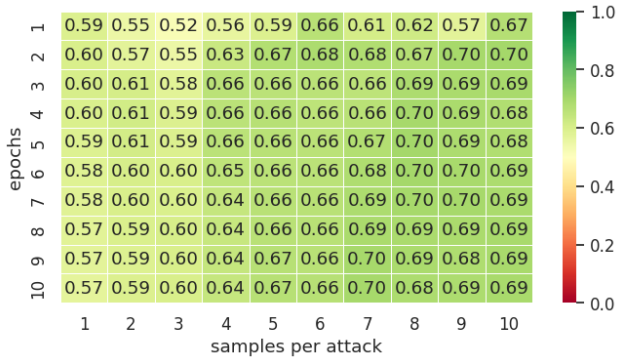


Figure 5.24: Heatmap showing the balanced accuracy of Whistleblower's MLP when evaluated on unseen attacks, across different combinations of training epochs and sample count.

In practice, it is unknown whether a model under inspection has been poisoned and, if so, at what ratio. We therefore examined Whistleblower's sensitivity to the poison ratio. Figure 5.25 shows that Whistleblower performs better when evaluated on poison ratios close to those seen during training, reflecting the fact that the latent-space signature of poisoned samples is influenced by the proportion of poisoned examples used during model training. This finding motivates training Whistleblower on a range of poison ratios when preparing it for deployment.

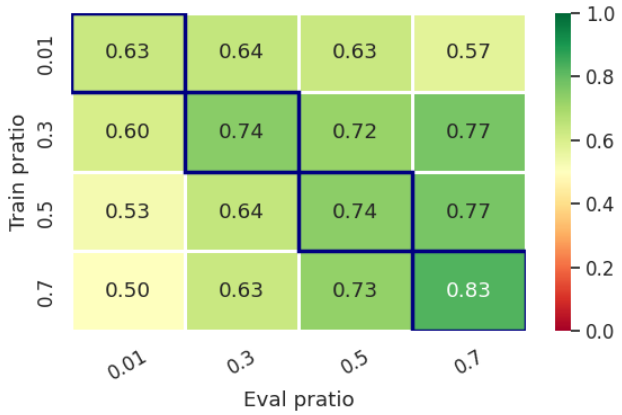


Figure 5.25: Heatmap showing the balanced accuracy of Whistleblower's MLP when evaluated on activations from models trained with poison ratios not seen during training.

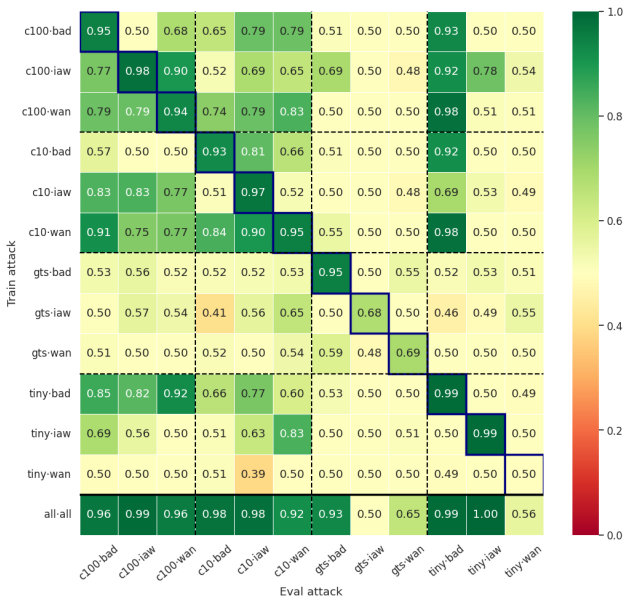
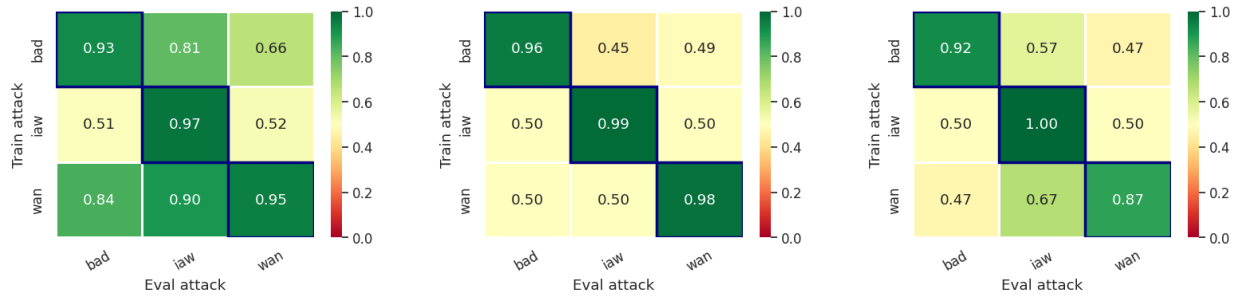


Figure 5.26: Heatmap showing the balanced accuracy of Whistleblower's MLP evaluated on mean activations from models trained on other datasets. c100 = CIFAR-100, c10 = CIFAR-10, gts = GTSRB, tiny = Tiny ImageNet. Abbreviated attack names: bad = BadNets, iaw = InputAware, wan = WaNet. The all·all row corresponds to a Whistleblower trained on all attacks and all datasets.

An important aspect of Whistleblower's viability in realistic settings is its ability to generalise across datasets, since practitioners may apply it to models trained on datasets not seen during Whistleblower's own training. This scenario arises frequently in practice, as many deployed models are fine-tuned from pre-trained checkpoints. Figure 5.26 shows Whistleblower's cross-dataset performance. Transfer between CIFAR-10 and CIFAR-100 is generally successful, with most values approaching 80% balanced accuracy. BadNets is notable as the only attack on Tiny ImageNet that can be detected in an out-of-distribution setting by classifiers trained on CIFAR-10 or CIFAR-100, even across attack types. Performance on GTSRB and Tiny ImageNet is considerably worse, and notably this degradation is not limited to cross-dataset transfer: even in-distribution detection (i.e. training and evaluating on the same dataset and attack, such as GTSRB-to-GTSRB and Tiny ImageNet-to-Tiny ImageNet) is similarly poor. This rules out a purely cross-domain explanation and points instead to a more fundamental issue with these datasets or with the underlying poisoned models. Since models for GTSRB and Tiny ImageNet were obtained as pre-trained checkpoints from the BackdoorBench team rather than trained in-house, and their clean accuracy and attack success rate were not logged, it is possible that they were trained with different hyperparameters than the 20-epoch schedule used for our in-house models, which could result in a substantially different latent space. Another factor, at least for Tiny ImageNet, is that this dataset is more complex due to its larger image resolution. WaNet in both GTSRB and Tiny ImageNet, and InputAware in GTSRB, achieve near-zero detection performance regardless of whether Whistleblower was trained on that specific attack or on all attacks. This suggests that these may be more challenging datasets that require either more training examples or a higher-capacity classifier,

and that the 64-neuron hidden layer may be insufficient for cross-dataset generalisation. This is an identified direction for improvement rather than a fundamental obstacle. We next evaluated Whistleblower across three architectures: PreActResNet-18 (used throughout all prior experiments), VGG-19-BN (also a convolutional network), and ViT-B/16 (a Vision Transformer). For the ViT, the activation cache is constructed from multi-head attention layers rather than convolutional layers, as described in Section 4.3. Since each architecture produces activation vectors of different dimensionality, a separate Whistleblower must be trained for each. Figure 5.27 shows cross-attack performance for each architecture; the ResNet-18 results reproduce the earlier findings and confirm generalisation within that architecture. For VGG-19-BN and ViT-B/16, however, cross-attack generalisation is largely absent, with the sole exception of the ViT classifier trained on WaNet, which partially detects InputAware.

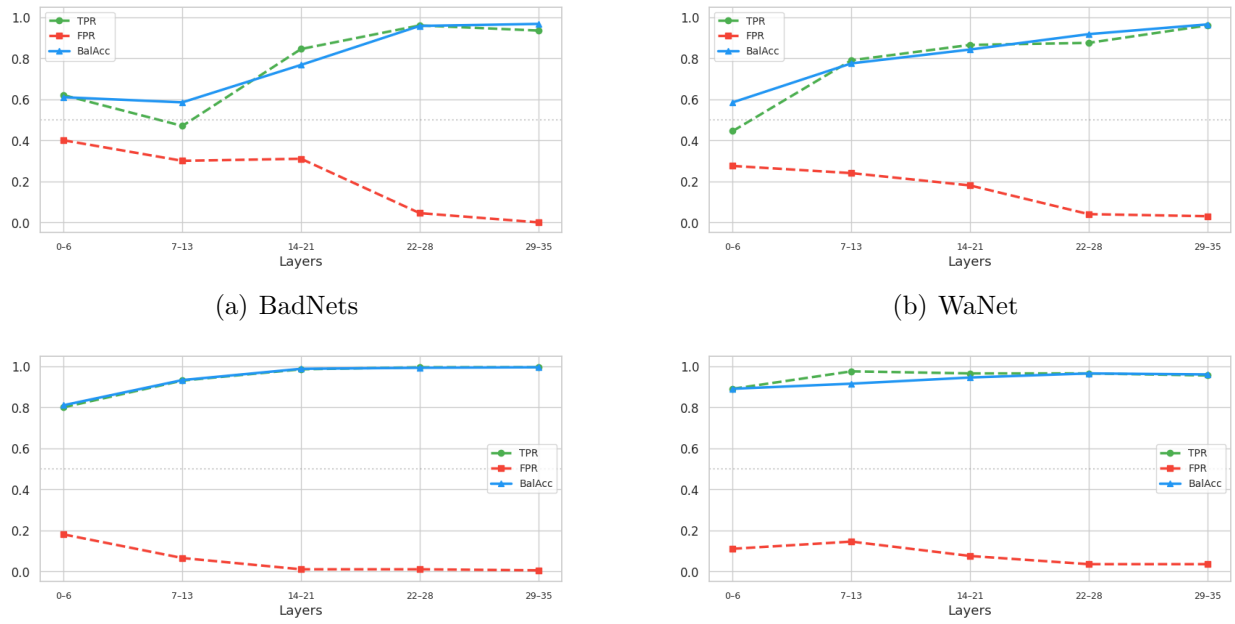


(a) ResNet-18 (b) VGG-19 (c) ViT

Figure 5.27: Heatmaps showing the balanced accuracy of Whistleblower's MLP trained on one attack and evaluated on samples from another model poisoned by another attack, for three architectures. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries correspond to out-of-distribution generalisation. Abbreviated attack names: bad = BadNets, iaw = InputAware, wan = WaNet.

Nevertheless, in-distribution performance remains strong for both architectures, which motivates training Whistleblower on as many attacks as possible, so that any novel attack producing a similar latent-space signature to a known one can still be detected. In a deployment setting, a separate Whistleblower would need to be maintained for each target architecture, with priority given to the most widely used ones.

The final Whistleblower experiment was not primarily concerned with generalisation but rather with localising the layer-wise distribution of backdoor information within the network, complementing the analysis in Section 5.1. To this end, the network was partitioned into five non-overlapping groups of layers (beginning, beginning-middle, middle, middle-end, and end), and a separate Whistleblower was trained and evaluated on each group. Figure 5.28 confirms the preliminary findings that backdoor information is distributed across layers rather than concentrated at the output, and further reveals that even for attacks such as BadNets and WaNet, which concentrate their signal in the later layers (as shown in Figure 5.3), the first seven layers alone contain sufficient information to achieve approximately 60% balanced accuracy, rising to around 80% in the middle group. For InputAware and ReFool, which distribute backdoor information more broadly, the first two layer groups already achieve 80 to 90% balanced accuracy, with the middle group approaching 100%.



(a) BadNets

(b) WaNet

(c) InputAware

(d) ReFool

Figure 5.28: Balanced accuracy, TPR, and FPR of Whistleblower's MLP trained on different layer groups and evaluated on the same layer group and attack used during training.

It is worth noting that Whistleblower is not directly compared against the unsupervised methods in a head-to-head table, as such a comparison would be of limited value. In-distribution, most of the time Whistleblower achieves near-perfect balanced accuracy, which would trivially outperform all unsupervised methods regardless of their merits, while its out-of-distribution performance, although promising in several cases, is not yet consistently reliable enough to support a fair like-for-like comparison. Whistleblower's principal strength instead lies in training on a diverse set of attacks simultaneously, which is believed to improve out-of-distribution performance; however, training on all available attacks leaves no held-out attacks against which to measure out-of-distribution performance. One could instead train on a subset of attacks to preserve a held-out evaluation set, though it is unclear how this trade-off between training diversity and evaluation coverage would be best resolved. In either case, a comparison against the unsupervised methods would be inappropriate.

5.4 Ethical Considerations

Backdoored machine learning models represent a concrete and growing threat to individuals and to society. The supply chain attack scenario targeted in this work, in which a defender receives a model of unknown provenance from a public repository such as HuggingFace, is particularly relevant given the rapid proliferation of pre-trained models in production systems. The methods developed in this work are best characterised as a security auditing tool for machine learning models. As such, the system does not operate on data about people directly: it only requires access to a model's internal activations, which have already

undergone substantial transformation. None of the datasets used in this work, CIFAR-10, CIFAR-100, GTSRB, or Tiny ImageNet, contain personal data. However, in a real world deployment, a model could be trained on medical records, facial images, or other behavioural data, in which case the activations inspected by the detection methods would constitute personal data being processed, even if only indirectly. In that scenario, the operator deploying Whistleblower would need to establish an appropriate legal basis under Article 6 of the GDPR, apply the data minimisation principle, for example by substituting real samples with synthetic equivalents wherever possible, and ensure that activation data are not retained beyond the duration of the audit. It is also important to note that the operator's detection work should not be replaced by Whistleblower but assisted by it. Whistleblower is best understood as a screening instrument that narrows the space of concern for a human expert, rather than as a definitive verdict. This connects to a broader principle articulated in both the AI Act and the European Commission's Ethics Guidelines for Trustworthy AI: the requirement for human agency and oversight.

Under Regulation (EU) 2024/1689 on Artificial Intelligence (the AI Act), the system does not fall within the categories of prohibited AI practice listed in Article 5, and it is not itself a high risk AI system as defined in Article 6 and Annex III. It is not a General Purpose AI System either, since it is designed for one specific and narrow task and cannot be meaningfully repurposed for other domains, so it does not carry systemic risk in the sense contemplated by Chapter V of the AI Act. Classification does, however, depend on deployment context. Should Whistleblower be integrated as a component of a conformity assessment process for a high risk AI system, for example as part of an operator's auditing pipeline for a medical imaging model or an autonomous vehicle perception system, its outputs would feed into a regulated decision making chain, and the overall system would need to satisfy the transparency, traceability, and human oversight requirements of Chapter III of the AI Act. In that scenario, explicit documentation of Whistleblower's known limitations would be mandatory. In its current form, as a research prototype evaluated exclusively on standard benchmark datasets that are all publicly available under their respective licences, and built on the open source BackdoorBench framework, the system is not subject to mandatory conformity assessment, though the risk based logic of the AI Act is still a useful lens through which to consider its design.

The most substantive ethical concern raised by this work is its dual use character. Backdoor detection research occupies an inherently ambiguous position: understanding how to detect backdoors requires a deep understanding of how backdoors work, and the same knowledge that enables detection can, in principle, inform attack development. This tension is particularly acute for Whistleblower. Publicly releasing the full training pipeline, including the code used to generate poisoned models and the evaluation infrastructure built on BackdoorBench, would allow an adversary to use the system as an oracle. An attacker could iteratively craft new trigger patterns, evaluate whether Whistleblower detects them, and refine the attack until it evades detection. This adaptive feedback loop could accelerate the development of more dangerous backdoors, effectively weaponising a defensive tool against the very systems it was designed to protect. This is, however, a familiar trade off in security research generally: if scientists wish to collaborate and build on each other's work, they must accept some risk that publishing their findings also benefits malicious actors. In this case, we do not believe additional mitigation is warranted, since BackdoorBench itself is already public and already

functions as such an oracle for attackers.

This concern fits within the dual use framework discussed in the literature on AI ethics and biosecurity research. Koplin [44] observes that dual use dilemmas arise whenever the same technology can promote human wellbeing or enable harm, and argues that meaningful mitigation requires intervention at multiple points along the research pipeline, not only at the moment of release.

Machine learning research carries an environmental cost that deserves acknowledgement. The full experimental pipeline for this thesis involved training poisoned models across multiple attacks, architectures, datasets, poison ratios, and random seeds, requiring hundreds of training runs on GPU hardware, which represents a non-trivial compute footprint. Two design choices help mitigate this. First, activations were cached to avoid redundant forward passes, which also sped up iteration during development. Second, the methods most likely to be relevant in practice, ZRF and Whistleblower, are tools that are relatively inexpensive to run compared with modern deep learning systems such as large language models, so their environmental cost in deployment is modest.

This work has been carried out in accordance with the ethical principles of the *Código Deontológico de la Ingeniería en Informática* of the Consejo General de Colegios Profesionales de Ingeniería en Informática de España (CCII) and the IEEE Code of Ethics. In particular, the work upholds the principles of honesty and rigour in reporting results, including negative results and failure cases, transparency about limitations, and the responsible development of technology intended to protect systems on which people depend. All baselines are evaluated under identical conditions to the proposed methods, and the limitations identified in Section 5.5 and in Chapter 4 are disclosed explicitly rather than minimised. The experimental setup is documented in sufficient detail to support independent replication.

5.5 Limitations

Several limitations of the present work should be acknowledged.

Small and non-stratified evaluation sets. All experiments were conducted on datasets of 1,000 samples or fewer, and class stratification was not enforced. As a result, the distribution of classes within each evaluation set may be non-uniform, which could affect the reliability of the reported metrics.

Dependency on BackdoorBench. The experimental pipeline depends on BackdoorBench for attack implementations and infrastructure. While this facilitates reproducibility, BackdoorBench has not received significant maintenance commits in some time, and several bugs reported by the community remain unresolved. This introduces a risk of undetected errors in the experimental setup.

Arbitrary Whistleblower architecture. The architecture of Whistleblower classifier, including the number of hidden layers, hidden size, and training hyperparameters, was selected based on initial experimentation rather than a systematic search. Different configurations may yield substantially different results.

Limited architectural scope. Whistleblower's cross-architecture evaluation covers PreActResNet-18, VGG-19-BN, and ViT-B/16, which, despite distributing information across layers, still produce a single output representation. Whether Whistleblower's supervised approach transfers to architectures with a more extreme degree of representational distribution, such as object detectors with multiple parallel detection heads, remains untested.

Chapter 6

Conclusions

This thesis set out from a single observation: the dominant family of latent-space backdoor detectors rests on the assumption that the backdoor signature is concentrated in one compact representation, typically the penultimate layer of a classifier, and this assumption grows fragile as both attacks and architectures become more distributed. The work pursued two complementary goals against that backdrop, a survey culminating in a gap analysis, and the design and evaluation of detection methods that operate over the full latent space rather than a single bottleneck.

On the descriptive side, Chapter 3 organised the backdoor literature along a four-axis attack taxonomy (attack scenario, modification target, trigger, and dormancy condition) and a methodological taxonomy of defences (activation- and latent-space methods, trigger inversion, model-level scanning methods, and runtime detection on the detection side; proactive, weight-preserving, and weight-modifying families on the mitigation side). The recurring thread of that survey is that almost every activation-based detector commits to a single representative layer; the principal exception, the topological-evolution line of work [66], confirms the value of reading the signal across layers but remains tied to standard classifiers.

On the methodological side, Chapter 4 introduced four detectors that each test a different hypothesis about how a distributed backdoor manifests in activation space, and the empirical study in Chapter 5 returned three main findings. First, backdoor-related information is genuinely distributed across the network rather than confined to its final layers, which is the central premise the proposed methods exploit. Second, among the unsupervised methods, ZRF alone remained competitive with established baselines across all four attacks, and its strongest variant is essentially Activation Clustering generalised to draw on every layer rather than the last one, locating the benefit in the *distribution of layers used* rather than in any new statistic; CAP did not exceed chance and BMD detected only at an impractically high false-positive rate. Third, the supervised reframing embodied by Whistleblower proved both effective in-distribution and, more surprisingly, capable of out-of-distribution generalisation to previously unseen attacks, with a small training budget sufficing.

Taken together, these results turn the distributed nature of the backdoor signal from a mere observation into an exploitable resource: precisely because the signature is spread across the

network, a detector that reads the whole latent space rather than a single bottleneck can recover evidence that layer-localised methods discard. Crucially, this is not only a conceptual point but a practical one. Its clearest demonstration is ZRF which, despite relying on deliberately simple machinery, is the only method in our comparison to remain competitive across all four attacks, matching or surpassing the established baselines on most of them and ranking first on InputAware and WaNet. The boundaries of these results, the proof-of-concept scope, the per-method failure modes, the favourable hyperparameter-selection protocol, and adaptive attacks that obfuscate their own activations, are examined honestly in Chapter 5 (see in particular Section 5.5). The direction we find most promising, however, lies on the supervised side: Whistleblower learns a backdoor signature directly from activations and generalises, at least in part, to attacks it never encountered during training, which suggests that supervised, learned detectors are a path with substantial untapped headroom. Pushing that approach further, towards detectors that transfer across architectures, attacks, and datasets, is in our view the most fruitful way forward, and it underlies much of the future work set out below.

6.1 Future Work

Several directions are identified for extending this work.

Scaling to additional architectures and datasets. The designed solution is centered around PreActResNet-18 and CIFAR-10; although we also evaluated it on additional architectures (VGG, ViT) and datasets (CIFAR-100, GTSRB, Tiny ImageNet), performance was substantially weaker, or the method did not work at all, in several of these settings. Future work should improve upon the method so that it generalises more reliably to other widely used architectures (e.g. VGG, ViT, YOLO, BERT) and to larger or more diverse datasets, to assess whether the observed generalisation patterns hold more broadly. Extension to other model types, such as large language models and object detectors, is also of interest.

Whistleblower-as-a-service. A natural extension would be to build a framework or service that, given a target architecture, automatically trains a Whistleblower detector using a standardised pipeline covering multiple attacks, seeds, fine-tuning configurations, and datasets. Pre-trained detectors for common architectures could then be released as an open-source resource, allowing practitioners to check whether a model is backdoored by passing it through the tool. The community could further extend the repository to support additional architectures and attack types.

Hyperparameter and architecture search. The hidden layer size (currently 64) was chosen arbitrarily. We experimented with smaller and larger hidden sizes, deeper and wider networks, and longer training schedules, but a more systematic and exhaustive search remains warranted. Phenomena such as double descent and grokking may be relevant when training with more data or for more epochs, and are worth investigating further.

Learning normality instead of anomaly. The current Whistleblower is a supervised MLP trained on a balanced mix of clean and poisoned activations, with a final single-neuron layer that classifies each sample as clean or poisoned. An alternative framing would be to train it

to model clean activations only, using a variational autoencoder (VAE), treating poisoned samples as anomalies with respect to a learned clean distribution.

Towards architecture-agnostic detectors. A given Whistleblower can currently only be trained for a single architecture, since its input size is tied to the dimensionality of that architecture's activations, which varies across architectures. It would be interesting to explore how this limitation could be addressed, enabling a single detector to generalise across architectures of varying width and depth. One idea would be to encode each layer's activations into a fixed-size embedding using a learned encoder, which could address the issue of varying width; however, it is unclear how well this would perform in practice, and it would still leave the issue of varying depth unaddressed.

Fully connected layers within ResNet-18. The current implementation ignores the fully connected layers of ResNet-18. Future experiments should assess whether including these layers as additional inputs to Whistleblower improves detection performance.

Robustness to fine-tuning. Fine-tuning a backdoored model may significantly alter its latent-space representations, potentially causing Whistleblower to fail. This vulnerability should be characterised empirically and, if confirmed, addressed in future work.

Multi-trigger detection. All experiments assume a single backdoor trigger per model. Extending the framework to detect models poisoned with two or more simultaneous triggers is a natural next step.

Adversarial attacks against Whistleblower. An attacker aware of Whistleblower could craft triggers specifically designed to evade it; indeed, recent work on *obfuscated activations* [6] shows that latent-space probes of exactly this kind can be bypassed by an adversary who optimises their activations directly against the monitor. Studying such adaptive attacks, and developing defences against them, is an important direction for future work.

Training on weights instead of activations. Rather than using activation statistics as input features, an alternative would be to train Whistleblower directly on model weights, which do not require a forward pass and may encode backdoor information in a complementary way.

Applying explainability to Whistleblower. It would be interesting to apply explainable AI (XAI) techniques (e.g. saliency methods or SHAP) to better understand Whistleblower's decisions. This could help identify which neurons or layers the detector relies on most, verify whether it is picking up on the actual backdoor signal rather than spurious correlations specific to the architectures and attacks it was trained on, and provide end users with an interpretable justification for why a given model was flagged as backdoored.

Beyond these specific avenues, the broader message of this thesis is a simple one: the very property that makes modern backdoors difficult to localise, their distribution across the network, is also what exposes them once a detector stops insisting on a single representative layer. Turning that insight into supervised, learned detectors that transfer across models, attacks, and datasets is, we believe, a particularly promising direction.

Bibliography

[1] Adler, M., Shavit, N., 2024. On the complexity of neural computation in superposition. arXiv preprint arXiv:2409.15318 .

[2] Akhtar, N., Mian, A., 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access 6, 14410–14430.

[3] Ameisen, E., Lindsey, J., Pearce, A., Gurnee, W., Turner, N.L., Chen, B., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Ben Thompson, T., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., Batson, J., 2025. Circuit tracing: Revealing computational graphs in language models. Transformer Circuits Thread URL: https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

[4] Bagdasaryan, E., Shmatikov, V., 2021. Blind backdoors in deep learning models, in: 30th USENIX Security Symposium (USENIX Security 21), pp. 1505–1521.

[5] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V., 2020. How to backdoor federated learning, in: International conference on artificial intelligence and statistics, PMLR. pp. 2938–2948.

[6] Bailey, L., Serrano, A., Sheshadri, A., Seleznyov, M., Taylor, J., Jenner, E., Hilton, J., Casper, S., Guestrin, C., Emmons, S., 2024. Obfuscated activations bypass llm latent-space defenses. arXiv preprint arXiv:2412.09565 .

[7] Bengio, Y., Courville, A., Vincent, P., 2013. Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence 35, 1798–1828.

[8] Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J.E., Hume, T., Carter, S., Henighan, T., Olah, C., 2023. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread URL: https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[9] Chan, S.H., Dong, Y., Zhu, J., Zhang, X., Zhou, J., 2022. Baddet: Backdoor attacks on object detection, in: European conference on computer vision, Springer. pp. 396–412.

[10] Chen, B., Carvalho, W., Baracaldo, N., Ludwig, H., Edwards, B., Lee, T., Molloy, I., Srivastava, B., 2018. Detecting backdoor attacks on deep neural networks by activation clustering. arXiv preprint arXiv:1811.03728 .

[11] Chen, H., Fu, C., Zhao, J., Koushanfar, F., 2019. DeepInspect: A black-box trojan detection and mitigation framework for deep neural networks, in: IJCAI, p. 8.

[12] Chen, X., Liu, C., Li, B., Lu, K., Song, D., 2017. Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 .

[13] Cheng, S., Shen, G., Tao, G., Zhang, K., Zhang, Z., An, S., Xu, X., Li, Y., Ma, S., Zhang, X., 2024a. Odscan: Backdoor scanning for object detection models, in: 2024 IEEE Symposium on Security and Privacy (SP), IEEE. pp. 1703–1721.

[14] Cheng, S., Shen, G., Zhang, K., Tao, G., An, S., Guo, H., Ma, S., Zhang, X., 2024b. Unit: Backdoor mitigation via automated neural distribution tightening, in: European Conference on Computer Vision, Springer. pp. 262–281.

[15] Chou, E., Tramer, F., Pellegrino, G., 2020. Sentinet: Detecting localized universal attacks against deep learning systems, in: 2020 IEEE Security and Privacy Workshops (SPW), IEEE. pp. 48–54.

[16] Cohen, J., Rosenfeld, E., Kolter, Z., 2019. Certified adversarial robustness via randomized smoothing, in: international conference on machine learning, PMLR. pp. 1310–1320.

[17] D'Agostino, R.B., 1971. An omnibus test of normality for moderate and large sample sizes. Biometrika 58, 341–348.

[18] D'Agostino, R.B., Pearson, E.S., 1973. Tests for departure from normality. empirical results for the distributions of b_2 and $\sqrt{b_1}$. Biometrika 60, 613–622.

[19] Doan, B.G., Abbasnejad, E., Ranasinghe, D.C., 2020. Februus: Input purification defense against trojan attacks on deep neural network systems, in: Proceedings of the 36th Annual Computer Security Applications Conference, pp. 897–912.

[20] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 .

[21] Dziugaite, G.K., Ghahramani, Z., Roy, D.M., 2016. A study of the effect of jpg compression on adversarial images. arXiv preprint arXiv:1608.00853 .

[22] Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., Olah, C., 2022. Toy models of superposition. Transformer Circuits Thread URL: https://transformer-circuits.pub/2022/toy_model/index.html.

[23] Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei,

D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., Olah, C., 2021. A mathematical framework for transformer circuits. Transformer Circuits Thread URL: https://transformer-circuits.pub/2021/framework/index.html.

[24] Feng, Y., Ma, B., Zhang, J., Zhao, S., Xia, Y., Tao, D., 2022. Fiba: Frequency-injection based backdoor attack in medical image analysis, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 20876–20885.

[25] Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B., 2020. Sharpness-aware minimization for efficiently improving generalization. arXiv preprint arXiv:2010.01412 .

[26] Frankle, J., Carbin, M., 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 .

[27] Gao, Y., Doan, B.G., Zhang, Z., Ma, S., Zhang, J., Fu, A., Nepal, S., Kim, H., 2020. Backdoor attacks and countermeasures on deep learning: A comprehensive review. arXiv preprint arXiv:2007.10760 .

[28] Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D.C., Nepal, S., 2019. Strip: A defence against trojan attacks on deep neural networks, in: Proceedings of the 35th annual computer security applications conference, pp. 113–125.

[29] Gu, T., Dolan-Gavitt, B., Garg, S., 2017. BadNets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733 .

[30] Guo, W., Wang, L., Xing, X., Du, M., Song, D., 2019. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. arXiv preprint arXiv:1908.01763 .

[31] Hayase, J., Kong, W., Somani, R., Oh, S., 2021. Spectre: Defending against backdoor attacks using robust statistics, in: International Conference on Machine Learning, PMLR. pp. 4129–4139.

[32] He, K., Zhang, X., Ren, S., Sun, J., 2016a. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

[33] He, K., Zhang, X., Ren, S., Sun, J., 2016b. Identity mappings in deep residual networks, in: European conference on computer vision, Springer. pp. 630–645.

[34] Hong, J., Zeng, Y., Yu, S., Lyu, L., Jia, R., Zhou, J., 2023. Revisiting data-free knowledge distillation with poisoned teachers, in: International Conference on Machine Learning, PMLR. pp. 13199–13212.

[35] Hong, S., Carlini, N., Kurakin, A., 2022. Handcrafted backdoors in deep neural networks. Advances in Neural Information Processing Systems 35, 8068–8080.

[36] Huang, D., Bu, Q., 2024. Adversarial feature map pruning for backdoor, in: International Conference on Learning Representations, pp. 17523–17538.

[37] Huang, K., Li, Y., Wu, B., Qin, Z., Ren, K., 2022. Backdoor defense via decoupling the training process. arXiv preprint arXiv:2202.03423 .

[38] Jarque, C.M., Bera, A.K., 1980. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. Economics letters 6, 255–259.

[39] Jebreel, N.M., Domingo-Ferrer, J., Li, Y., 2023. Defending against backdoor attacks by layer-wise feature analysis, in: Pacific-Asia conference on knowledge discovery and data mining, Springer. pp. 428–440.

[40] Jin, H., Chen, R., Chen, J., Zheng, H., Zhang, Y., Wang, H., 2024. Catchbackdoor: Backdoor detection via critical trojan neural path fuzzing, in: European Conference on Computer Vision, Springer. pp. 90–106.

[41] Jocher, G., Chaurasia, A., Qiu, J., et al., 2023. Ultralytics YOLOv8. URL: https://github.com/ultralytics/ultralytics.

[42] Karim, N., Arafat, A.A., Khalid, U., Guo, Z., Rahnavard, N., 2024. Augmented neural fine-tuning for efficient backdoor purification, in: European Conference on Computer Vision, Springer. pp. 401–418.

[43] Kolouri, S., Saha, A., Pirsiavash, H., Hoffmann, H., 2020. Universal litmus patterns: Revealing backdoor attacks in cnns, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 301–310.

[44] Koplin, J.J., 2023. Dual-use implications of ai text generation. Ethics and Information Technology 25, 32.

[45] Krizhevsky, A., Hinton, G., et al., 2009. Learning multiple layers of features from tiny images .

[46] Le, N., Zhang, L.Y., Liao, K., Pan, S., Luo, W., 2025. Ted++: Submanifold-aware backdoor detection via layerwise tubular-neighbourhood screening. arXiv preprint arXiv:2510.14299 .

[47] Le, Y., Yang, X., et al., 2015. Tiny imagenet visual recognition challenge. CS 231N 7, 3.

[48] Levine, A., Feizi, S., 2020. Deep partition aggregation: Provable defense against general poisoning attacks. arXiv preprint arXiv:2006.14768 .

[49] Li, S., Xue, M., Zhao, B.Z.H., Zhu, H., Zhang, X., 2020a. Invisible backdoor attacks on deep neural networks via steganography and regularization. IEEE Transactions on Dependable and Secure Computing 18, 2088–2105.

[50] Li, Y., Jiang, Y., Li, Z., Xia, S.T., 2022. Backdoor learning: A survey. IEEE transactions on neural networks and learning systems 35, 5–22.

[51] Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., Ma, X., 2021a. Anti-backdoor learning: Training clean models on poisoned data. Advances in Neural Information Processing Systems 34, 14900–14912.

[52] Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., Ma, X., 2021b. Neural attention distillation: Erasing backdoor triggers from deep neural networks. arXiv preprint arXiv:2101.05930 .

[53] Li, Y., Lyu, X., Ma, X., Koren, N., Lyu, L., Li, B., Jiang, Y.G., 2023. Reconstructive neuron pruning for backdoor defense, in: International Conference on Machine Learning, PMLR. pp. 19837–19854.

[54] Li, Y., Zhai, T., Wu, B., Jiang, Y., Li, Z., Xia, S., 2020b. Rethinking the trigger of backdoor attack. arXiv preprint arXiv:2004.04692 .

[55] Liu, K., Dolan-Gavitt, B., Garg, S., 2018a. Fine-pruning: Defending against backdooring attacks on deep neural networks, in: International symposium on research in attacks, intrusions, and defenses, Springer. pp. 273–294.

[56] Liu, X., Li, M., Wang, H., Hu, S., Ye, D., Jin, H., Wu, L., Xiao, C., 2023. Detecting backdoors during the inference stage based on corruption robustness consistency, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16363–16372.

[57] Liu, Y., Lee, W.C., Tao, G., Ma, S., Aafer, Y., Zhang, X., 2019. Abs: Scanning neural networks for back-doors by artificial brain stimulation, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1265–1282.

[58] Liu, Y., Ma, S., Aafer, Y., Lee, W.C., Zhai, J., Wang, W., Zhang, X., 2018b. Trojaning attack on neural networks, in: 25th Annual Network And Distributed System Security Symposium (NDSS 2018), Internet Soc.

[59] Ma, H., Qiu, H., Gao, Y., Zhang, Z., Abuadbba, A., Xue, M., Fu, A., Zhang, J., Al-Sarawi, S.F., Abbott, D., 2023. Quantization backdoors to deep learning commercial frameworks. IEEE Transactions on Dependable and Secure Computing 21, 1155–1172.

[60] Ma, W., Wang, D., Sun, R., Xue, M., Wen, S., Xiang, Y., 2022. The "beatrix" resurrections: Robust backdoor detection via gram matrices. arXiv preprint arXiv:2209.11715 .

[61] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A., 2021. A survey on bias and fairness in machine learning. ACM computing surveys (CSUR) 54, 1–35.

[62] Meng, K., Bau, D., Andonian, A., Belinkov, Y., 2022. Locating and editing factual associations in gpt. Advances in neural information processing systems 35, 17359–17372.

[63] Mengara, O., Avila, A., Falk, T.H., 2024. Backdoor attacks to deep neural networks: A survey of the literature, challenges, and future research directions. IEEE Access 12, 29004–29023.

[64] Min, N.M., Pham, L.H., Sun, J., 2025. Unified neural backdoor removal with only few clean samples through unlearning and relearning. IEEE Transactions on Information Forensics and Security .

[65] Mo, X., Cheng, Y., Sun, N., Zhang, L.Y., Luo, W., Gao, S., 2025. Ted-last: Towards robust backdoor defense against adaptive attacks. arXiv preprint arXiv:2506.10722 .

[66] Mo, X., Zhang, Y., Zhang, L.Y., Luo, W., Sun, N., Hu, S., Gao, S., Xiang, Y., 2024.

Robust backdoor detection for deep learning via topological evolution dynamics, in: 2024 IEEE Symposium on Security and Privacy (SP), IEEE. pp. 2048–2066.

[67] Nguyen, A., Tran, A., 2021. Wanet–imperceptible warping-based backdoor attack. arXiv preprint arXiv:2102.10369 .

[68] Nguyen, T.A., Tran, A., 2020. Input-aware dynamic backdoor attack. Advances in Neural Information Processing Systems 33, 3454–3464.

[69] Pan, M., Zeng, Y., Lyu, L., Lin, X., Jia, R., 2023. ASSET: Robust backdoor data detection across a multiplicity of deep learning paradigms, in: 32nd USeNIX security symposium (USeNIX security 23), pp. 2725–2742.

[70] Phan, H., Shi, C., Xie, Y., Zhang, T., Li, Z., Zhao, T., Liu, J., Wang, Y., Chen, Y., Yuan, B., 2022. RIBAC: Towards robust and imperceptible backdoor attack against compact DNN, in: European Conference on Computer Vision, Springer. pp. 708–724.

[71] Qi, X., Xie, T., Wang, J.T., Wu, T., Mahloujifar, S., Mittal, P., 2023. Towards a proactive ML approach for detecting backdoor poison samples, in: 32nd USENIX Security Symposium (USENIX Security 23), pp. 1685–1702.

[72] Rakin, A.S., He, Z., Fan, D., 2020. Tbt: Targeted neural network attack with bit trojan, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 13198–13207.

[73] Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788.

[74] Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 .

[75] Saha, A., Subramanya, A., Pirsiavash, H., 2020. Hidden trigger backdoor attacks, in: Proceedings of the AAAI conference on artificial intelligence, pp. 11957–11965.

[76] SAS Institute, 1990. SAS/STAT User's Guide: GLM-VARCOMP. volume 2. SAS institute Incorporated.

[77] Schwarz, G., 1978. Estimating the dimension of a model. The annals of statistics , 461–464.

[78] Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T., 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. Advances in neural information processing systems 31.

[79] Shumailov, I., Zhao, Y., Bates, D., Papernot, N., Mullins, R., Anderson, R., 2021. Sponge examples: Energy-latency attacks on neural networks, in: 2021 IEEE European symposium on security and privacy (EuroS&P), IEEE. pp. 212–231.

[80] Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

[81] Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C., 2011. The german traffic sign recognition benchmark: a multi-class classification competition, in: The 2011 international joint conference on neural networks, IEEE. pp. 1453–1460.

[82] Tang, D., Wang, X., Tang, H., Zhang, K., 2021. Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection, in: 30th USENIX Security Symposium (USENIX Security 21), pp. 1541–1558.

[83] Tang, R., Du, M., Liu, N., Yang, F., Hu, X., 2020. An embarrassingly simple approach for trojan attack in deep neural networks, in: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 218–228.

[84] Tao, G., Liu, Y., Shen, G., Xu, Q., An, S., Zhang, Z., Zhang, X., 2022. Model orthogonalization: Class distance hardening in neural networks for better security, in: 2022 IEEE Symposium on Security and Privacy (SP), IEEE. pp. 1372–1389.

[85] Tran, B., Li, J., Madry, A., 2018. Spectral signatures in backdoor attacks. Advances in neural information processing systems 31.

[86] Turner, A., Tsipras, D., Madry, A., 2019. Label-consistent backdoor attacks. arXiv preprint arXiv:1912.02771 .

[87] Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y., 2019. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks, in: 2019 IEEE symposium on security and privacy (SP), IEEE. pp. 707–723.

[88] Weber, M., Xu, X., Karlaš, B., Zhang, C., Li, B., 2023. Rab: Provable robustness against backdoor attacks, in: 2023 IEEE Symposium on Security and Privacy (SP), IEEE. pp. 1311–1328.

[89] Wenger, E., Passananti, J., Bhagoji, A.N., Yao, Y., Zheng, H., Zhao, B.Y., 2021. Backdoor attacks against deep learning systems in the physical world, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 6206–6215.

[90] Wu, B., Chen, H., Zhang, M., Zhu, Z., Wei, S., Yuan, D., Shen, C., 2022. Backdoorbench: A comprehensive benchmark of backdoor learning. Advances in Neural Information Processing Systems 35, 10546–10559.

[91] Wu, D., Wang, Y., 2021. Adversarial neuron pruning purifies backdoored deep models. Advances in Neural Information Processing Systems 34, 16913–16925.

[92] Xian, X., Wang, G., Srinivasa, J., Kundu, A., Bi, X., Hong, M., Ding, J., 2023. A unified detection framework for inference-stage backdoor defenses. Advances in Neural Information Processing Systems 36, 7867–7894.

[93] Xu, X., Wang, Q., Li, H., Borisov, N., Gunter, C.A., Li, B., 2021. Detecting ai trojans using meta neural analysis, in: 2021 IEEE Symposium on Security and Privacy (SP), IEEE. pp. 103–120.

[94] Yao, Y., Li, H., Zheng, H., Zhao, B.Y., 2019. Latent backdoor attacks on deep neural

networks, in: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, pp. 2041–2055.

[95] Yu, Y., Liu, J., Li, S., Huang, K., Feng, X., 2022. A temporal-pattern backdoor attack to deep reinforcement learning, in: GLOBECOM 2022-2022 IEEE Global Communications Conference, IEEE. pp. 2710–2715.

[96] Yuan, D., Zhang, M., Wei, S., Liu, L., Wu, B., 2025. Activation gradient based poisoned sample detection against backdoor attacks, in: International Conference on Learning Representations, pp. 76161–76188.

[97] Zeng, F., Chen, Y., Cong, Y., Zhang, L., Li, S., Xu, J., Duan, J., 2025. Stealthy backdoor attack in sar target recognition with ascm-based physically realizable triggers. IEEE Transactions on Radar Systems .

[98] Zeng, Y., Chen, S., Park, W., Mao, Z.M., Jin, M., Jia, R., 2021a. Adversarial unlearning of backdoors via implicit hypergradient. arXiv preprint arXiv:2110.03735 .

[99] Zeng, Y., Pan, M., Just, H.A., Lyu, L., Qiu, M., Jia, R., 2023. Narcissus: A practical clean-label backdoor attack with limited information, in: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, pp. 771–785.

[100] Zeng, Y., Park, W., Mao, Z.M., Jia, R., 2021b. Rethinking the backdoor attacks' triggers: A frequency perspective, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 16473–16481.

[101] Zhang, H., Wang, Y., Yan, S., Zhu, C., Zhou, Z., Hou, L., Hu, S., Li, M., Zhang, Y., Zhang, L.Y., 2025. Test-time backdoor detection for object detection models, in: Proceedings of the Computer Vision and Pattern Recognition Conference, pp. 24377–24386.

[102] Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O., 2018. The unreasonable effectiveness of deep features as a perceptual metric, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 586–595.

[103] Zhao, P., Chen, P.Y., Das, P., Ramamurthy, K.N., Lin, X., 2020. Bridging mode connectivity in loss landscapes and adversarial robustness. arXiv preprint arXiv:2005.00060 .

[104] Zheng, R., Tang, R., Li, J., Liu, L., 2022a. Data-free backdoor removal based on channel lipschitzness, in: European Conference on Computer Vision, Springer. pp. 175–191.

[105] Zheng, R., Tang, R., Li, J., Liu, L., 2022b. Pre-activation distributions expose backdoor neurons. Advances in Neural Information Processing Systems 35, 18667–18680.

[106] Zhou, J., 2025. A survey on backdoor attack and defense in deep learning: A paradigm shift from data-driven to behavior-driven triggers. Applied and Computational Engineering 184, 206–211. doi:10.54254/2755-2721/2025.LD27270.

[107] Zhu, M., Wei, S., Shen, L., Fan, Y., Wu, B., 2023a. Enhancing fine-tuning based

backdoor defense with sharpness-aware minimization, in: Proceedings of the IEEE/CVF international conference on computer vision, pp. 4466–4477.

[108] Zhu, M., Wei, S., Zha, H., Wu, B., 2023b. Neural polarizer: A lightweight and effective backdoor defense via purifying poisoned features. Advances in Neural Information Processing Systems 36, 1132–1153.

Annexes

A.1 Additional Figures and Tables

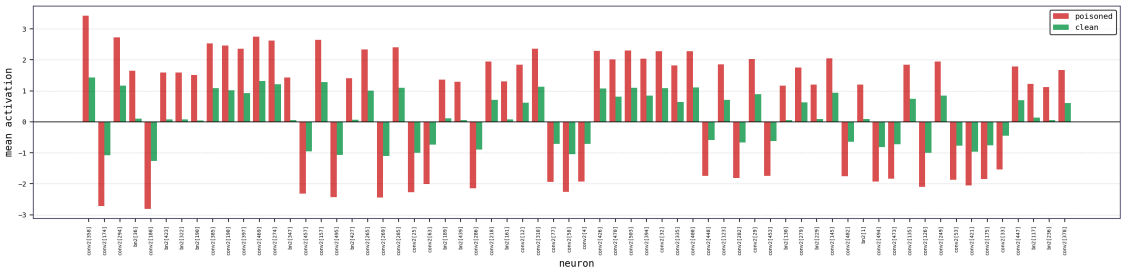
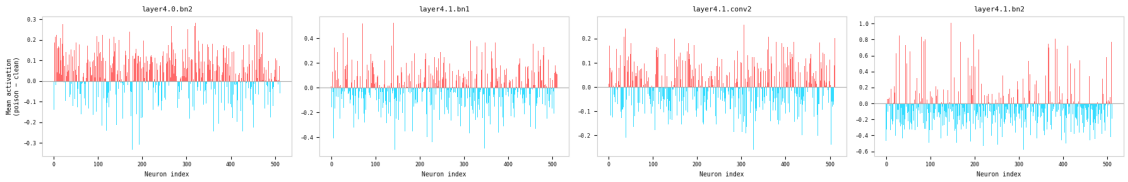
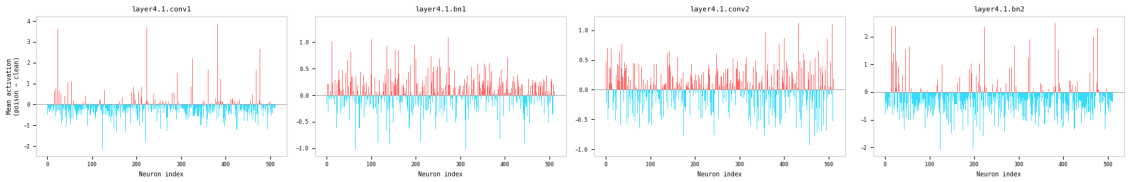


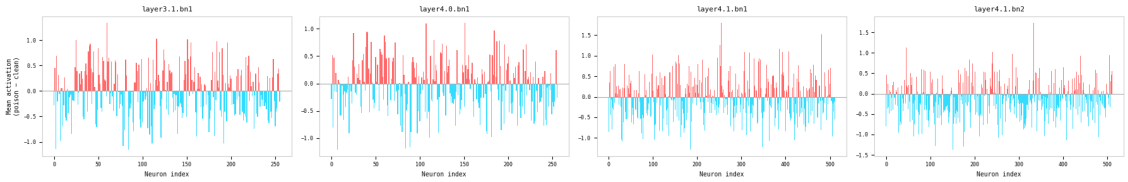
Figure A.1: Top 64 neurons of `layer4.1` with the largest mean absolute difference in activation magnitudes between BadNets-poisoned and clean samples, shown separately for each group. Green bars indicate the mean activation of clean samples and red bars indicate the mean activation of poisoned samples.



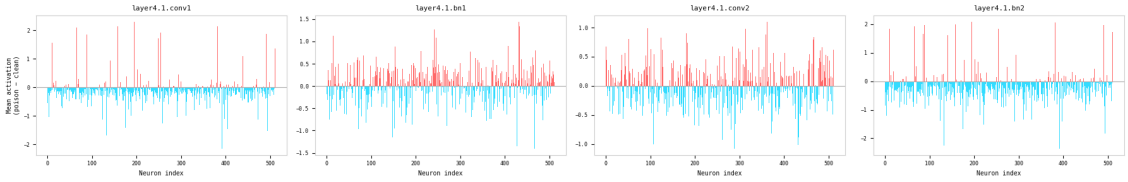
(a) BadNets



(b) WaNet



(c) InputAware



(d) ReFool

Figure A.2: Mean activation difference between poisoned and clean samples at the neuron level, shown for the four layers exhibiting the greatest overall difference, across the four representative attacks.

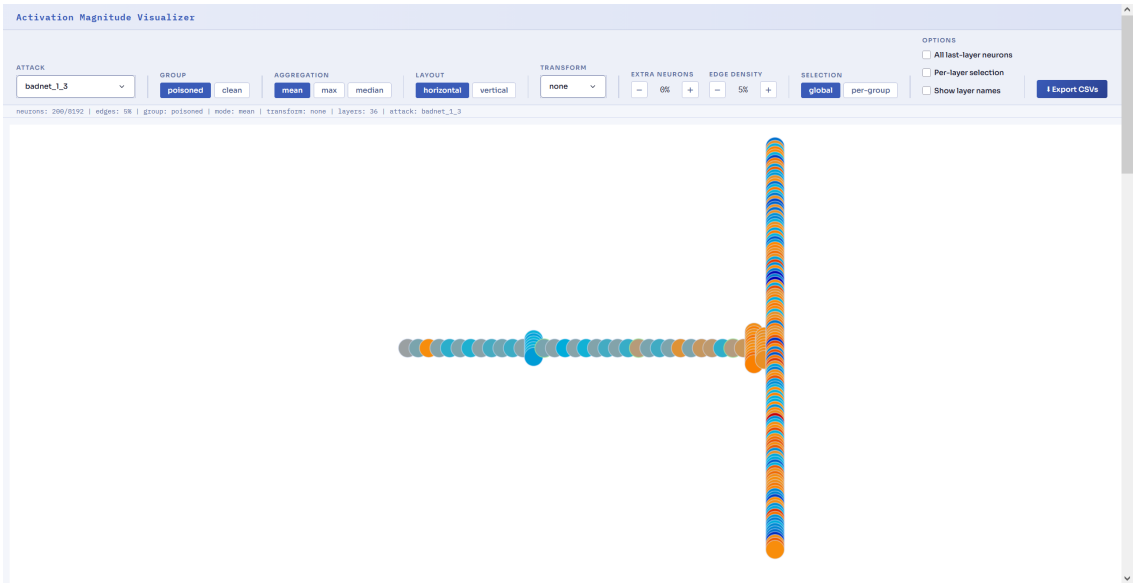


Figure A.3: Screenshot of the activation visualiser using global selection rather than per-layer selection, so neurons are ranked by their overall activation magnitude across the whole network rather than by how much they diverge between groups within their own layer.

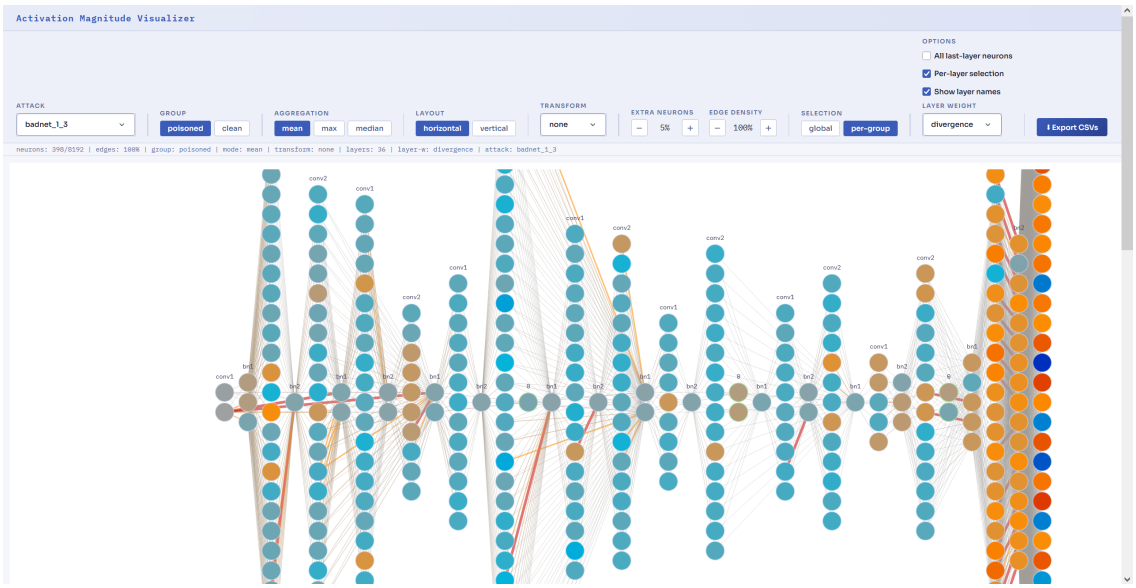
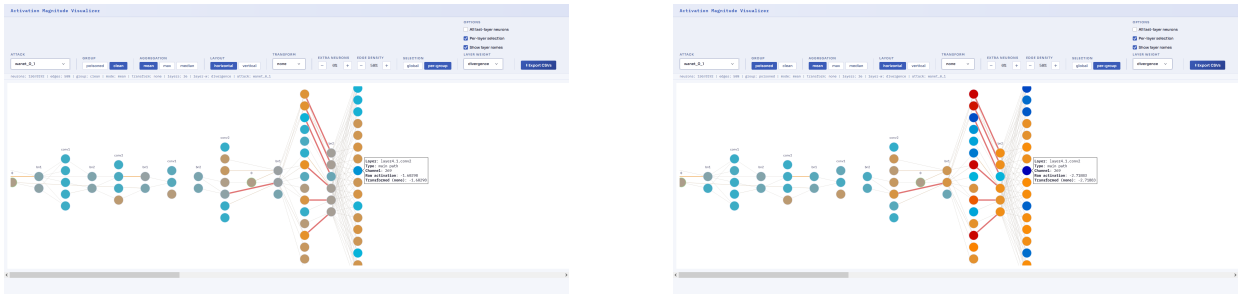
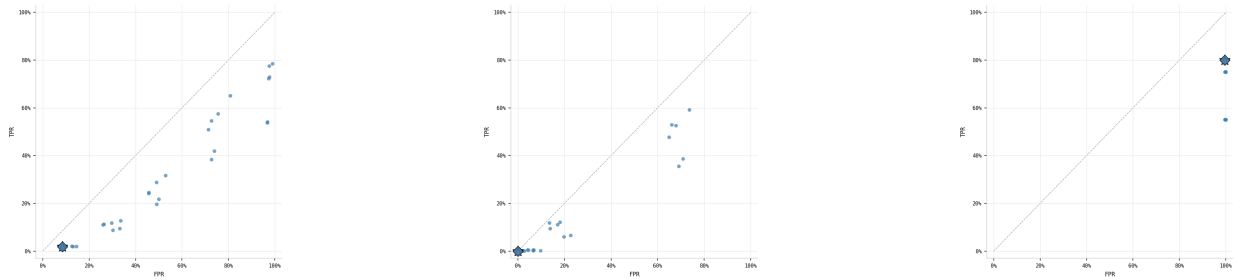


Figure A.4: Screenshot of the activation visualiser with 100% of edges drawn, the neuron budget set to +5%, and layer names shown.



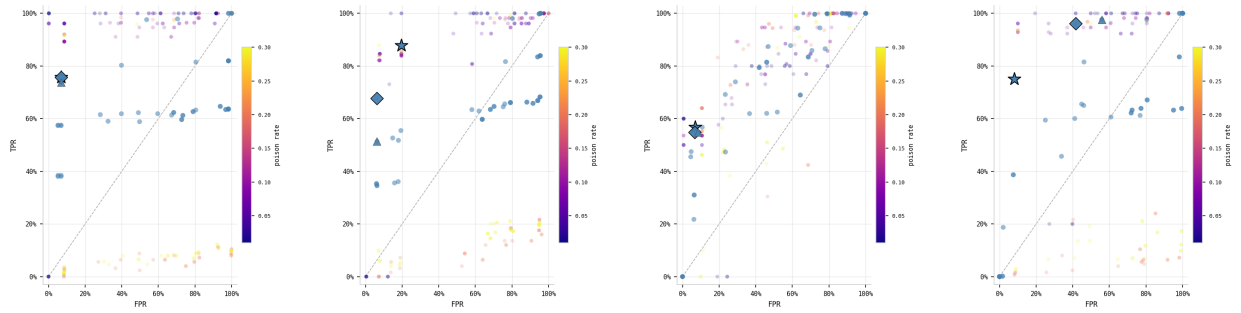
(a) Clean input (b) Poisoned input

Figure A.5: Close-up of a neuron's activation value, showing the difference between clean and poisoned inputs.



(a) mean (b) mean+std (c) raw

Figure A.6: Scatter plots showing each CAP configuration by its TPR and FPR averaged across attacks and poison ratios, for each cache mode (mean, mean+std, and raw). The star denotes the best configuration, the diamond the second best, and the triangle the third best.



(a) BadNets (b) InputAware (c) ReFool (d) WaNet

Figure A.7: Scatter plots showing each BMD configuration by its TPR and FPR for each attack, for the mean+std cache mode. Background points show individual runs coloured by poison ratio; blue points show each configuration averaged across all poison ratios. The star, diamond, and triangle denote the best, second-best, and third-best configurations by balanced accuracy respectively.

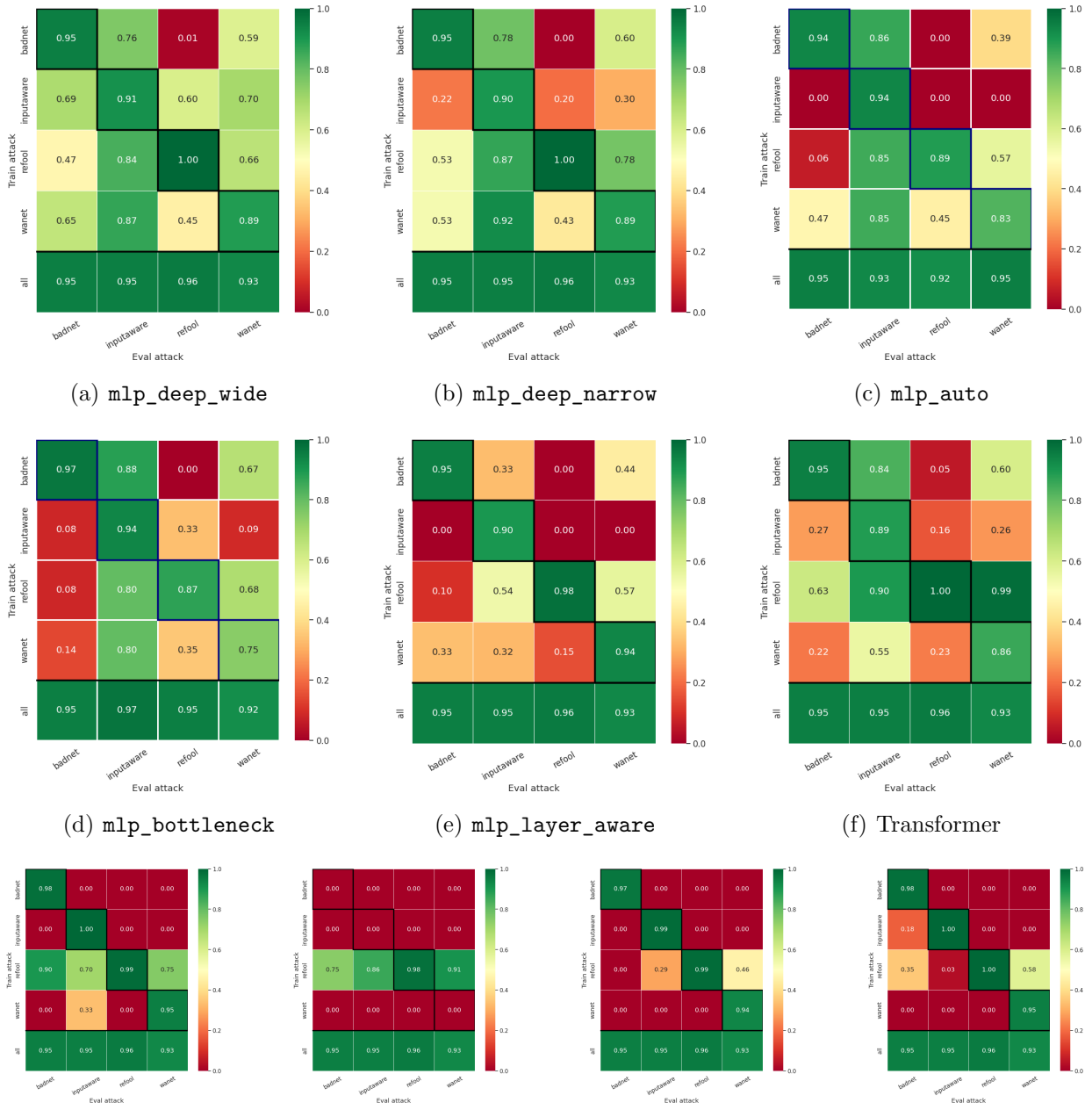
Table A.1: True Positive Rate (TPR) and False Positive Rate (FPR) by detection method and attack ($\rho = 0.10$). For the proposed methods, the configuration shown is the single hyperparameter setting that achieved the highest mean BAC across all attacks and poison rates within that method's grid search.

Method	**BadNet**	**Input-Aware**	**ReFool**	**WaNet**
		TPR		
Beatrix [60]	0.048	0.147	0.000	0.021
Spectral Signatures [85]	0.794	0.118	0.001	0.132
ABL [51]	0.742	0.024	0.008	0.121
STRIP [28]	0.752	0.295	0.112	0.062
SPECTRE [31]	0.702	0.461	0.090	0.148
AC [10]	0.898	0.144	0.000	0.000
ASSET [69]	0.930	0.737	1.000	0.682
SCAn [82]	0.000	0.331	0.711	0.279
AGPD [96]	0.756	0.000	0.933	0.311
CAP[a]	1.000	1.000	1.000	1.000
BMD[b]	0.932	0.419	0.811	0.973
ZRF[c]	0.893	0.839	0.536	0.929
		FPR		
Beatrix [60]	0.080	0.051	0.001	0.016
Spectral Signatures [85]	0.144	0.157	0.167	0.157
ABL [51]	0.094	0.118	0.110	0.092
STRIP [28]	0.105	0.102	0.106	0.107
SPECTRE [31]	0.144	0.078	0.157	0.151
AC [10]	0.035	0.062	0.000	0.000
ASSET [69]	0.594	0.562	0.542	0.560
SCAn [82]	0.000	0.000	0.000	0.000
AGPD [96]	0.000	0.001	0.001	0.002
CAP[a]	1.000	0.994	1.000	1.000
BMD[b]	0.387	0.460	0.440	0.397
ZRF[c]	0.086	0.196	0.106	0.100

[a] raw activations; threshold=0.30, warmup=20, clean_only=True, warmup_poison_ratio=rand

[b] mean activations; criteria=jb, min_criteria=1, sample_threshold=0.30, alpha=0.01

[c] mean+std activations; sil_threshold=0.20, variant=raw, minority_max_frac=0.2

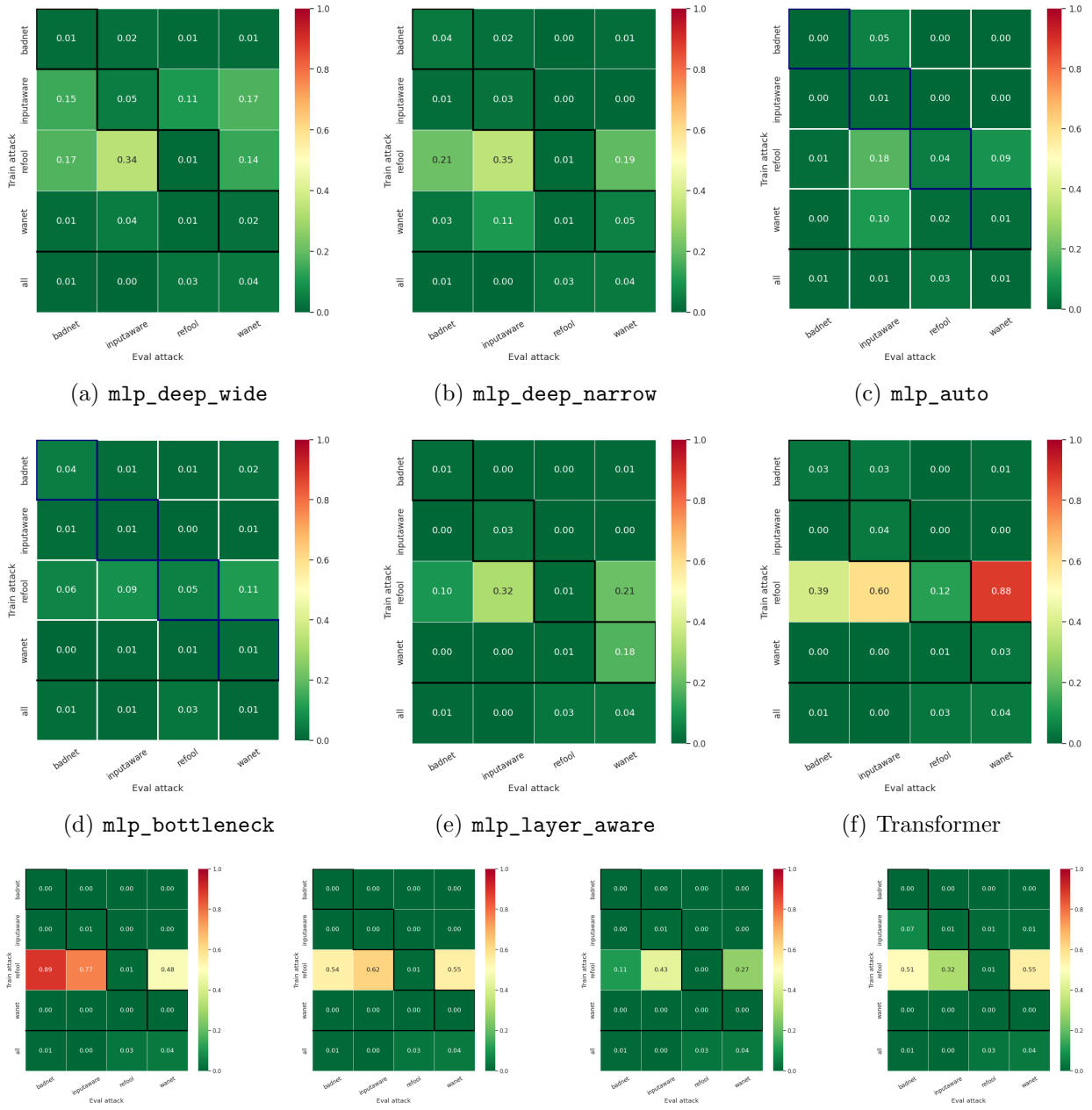


(a) `mlp_deep_wide` (b) `mlp_deep_narrow` (c) `mlp_auto`

(d) `mlp_bottleneck` (e) `mlp_layer_aware` (f) Transformer

(g) Logistic Regression (h) Linear SVM (i) Random Forest (j) LightGBM

Figure A.8: Heatmaps showing the TPR of all classifiers except the MLP, trained on mean activations from one attack and evaluated on samples from another model poisoned by another attack. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined.

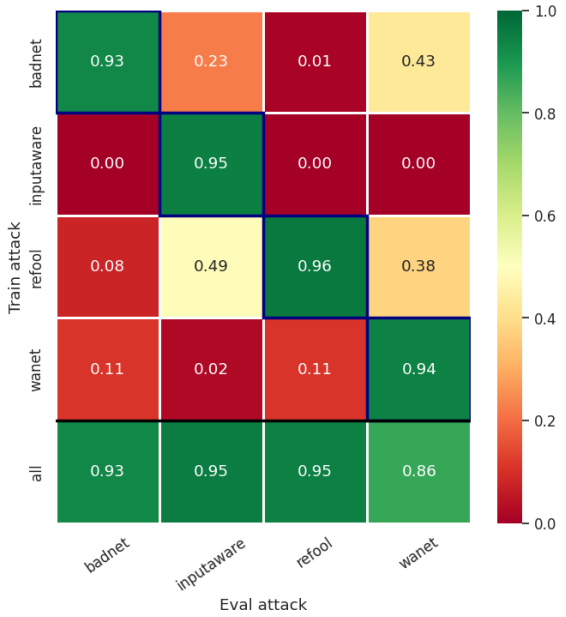


(a) `mlp_deep_wide` (b) `mlp_deep_narrow` (c) `mlp_auto`

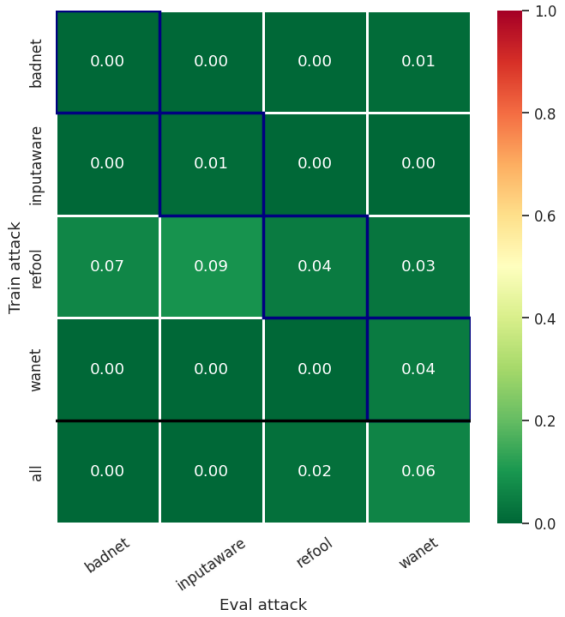
(d) `mlp_bottleneck` (e) `mlp_layer_aware` (f) Transformer

(g) Logistic Regression (h) Linear SVM (i) Random Forest (j) LightGBM

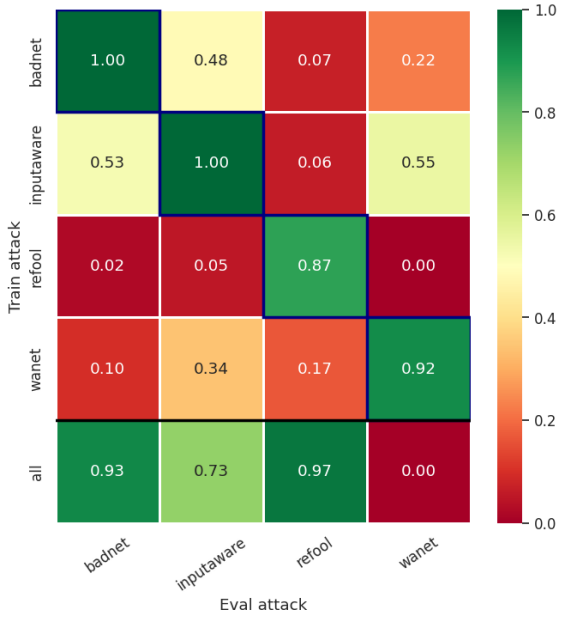
Figure A.9: Heatmaps showing the FPR of all classifiers except the MLP, trained on mean activations from one attack and evaluated on samples from another model poisoned by another attack. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined.



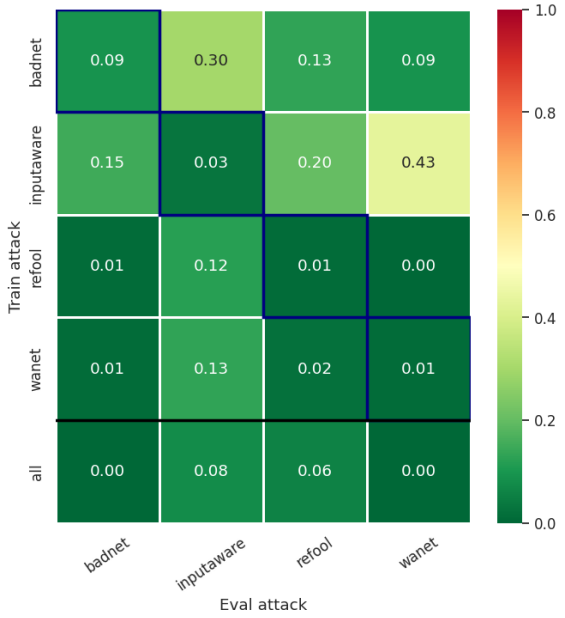
(a) mean+std MLP TPR



(b) mean+std MLP FPR

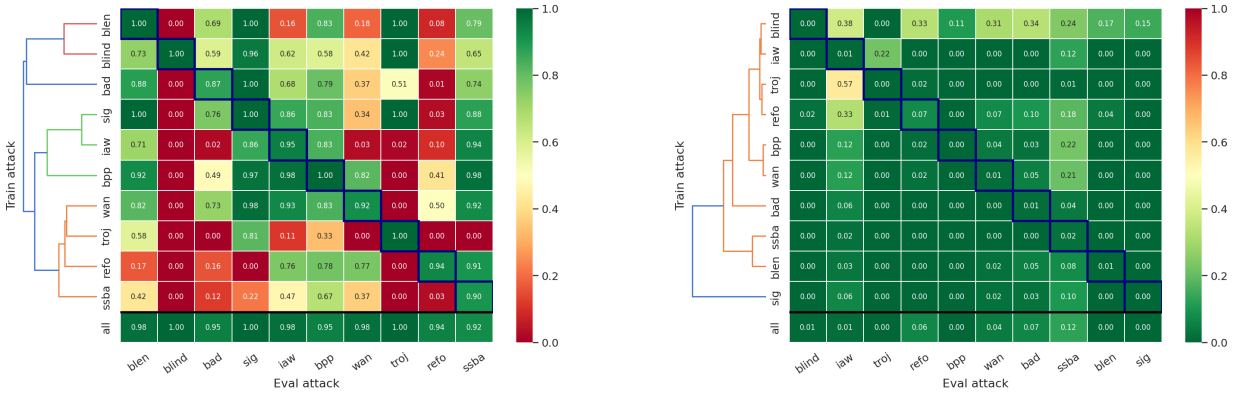


(c) raw MLP TPR



(d) raw MLP FPR

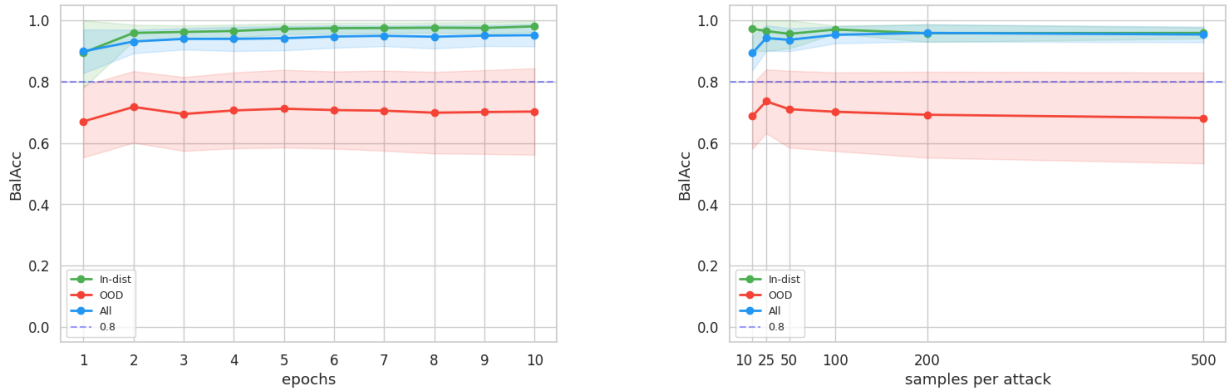
Figure A.10: Heatmaps showing the TPR and FPR of the MLP trained on mean activations with their standard deviations, or directly on raw activations, from one attack and evaluated on samples from another model poisoned by another attack. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined.



(a) Clustering based on TPR

(b) Clustering based on FPR

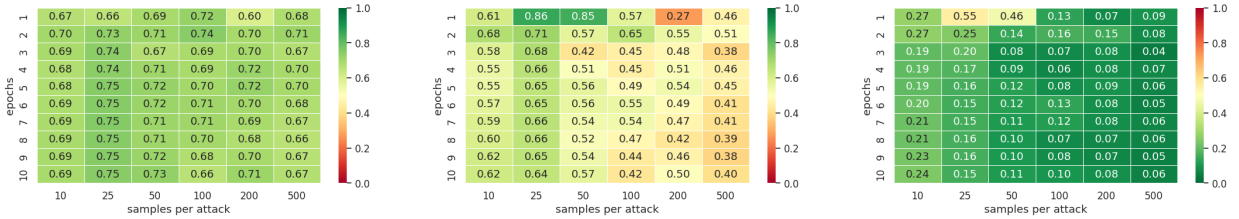
Figure A.11: Clustering based on the TPR and FPR obtained by training the MLP on mean activations from one attack and evaluating on samples from another, analogous to the balanced-accuracy clustering in Figure 5.21. Diagonal entries correspond to in-distribution evaluation (not used in computing the clustering); off-diagonal entries to out-of-distribution generalisation. The last row reports results when training on samples from all attacks combined (also excluded from the clustering computation). Abbreviated attack names: troj = TrojanNN, bad = BadNets, blen = Blended, refo = ReFool, iaw = InputAware, wan = WaNet. Note that clustering based on FPR is less informative about generalisation than clustering based on TPR or balanced accuracy, since it does not directly reflect detection capability.



(a) Balanced accuracy across epochs

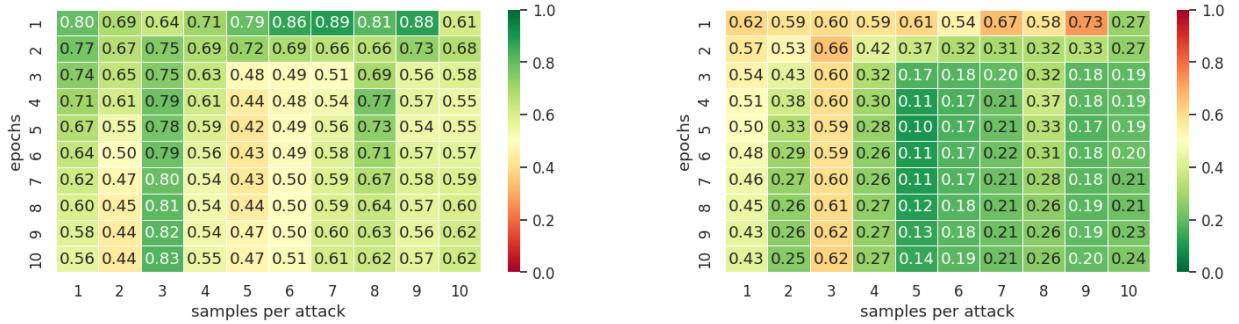
(b) Balanced accuracy across samples

Figure A.12: Balanced accuracy of Whistleblower's MLP across training epochs and across training sample counts, complementing the TPR and FPR results in Figures 5.22 and 5.23. In both cases, performance stabilises quickly. Even small numbers of training epochs or samples yield reasonable balanced accuracy, after which results remain largely unchanged.



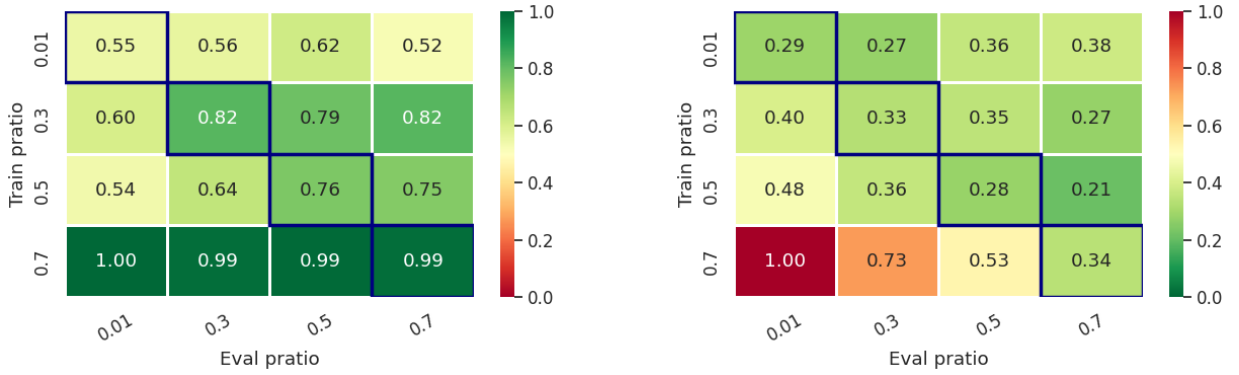
(a) Balanced accuracy (b) TPR (c) FPR

Figure A.13: Heatmaps showing the balanced accuracy, TPR, and FPR of Whistleblower's MLP when evaluated on unseen attacks, across different combinations of training epochs and sample count.



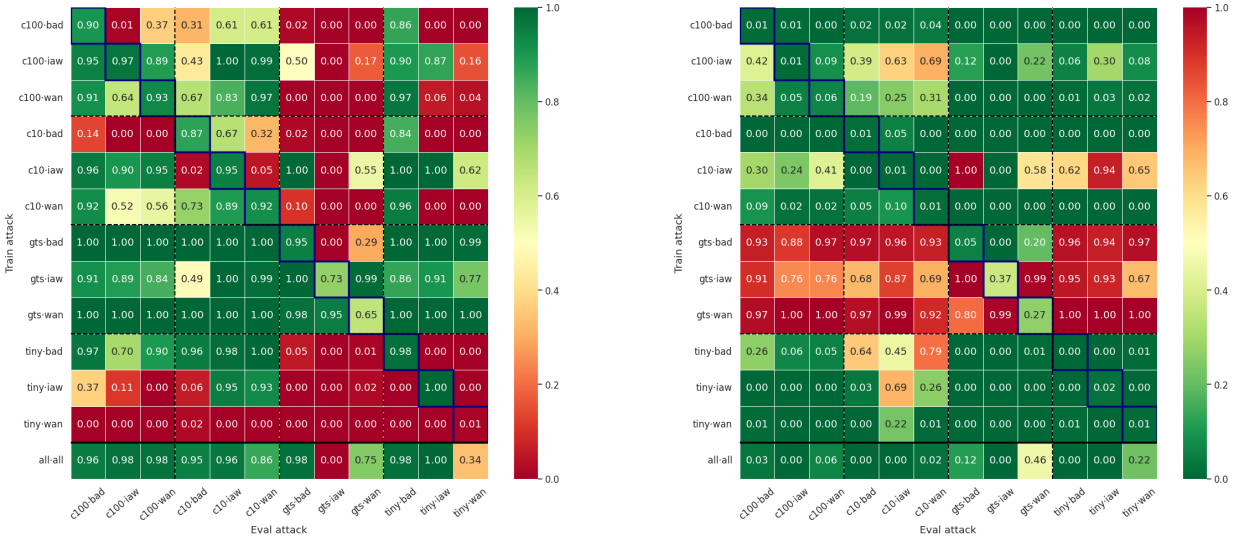
(a) TPR (b) FPR

Figure A.14: Heatmaps showing the TPR and FPR of Whistleblower's MLP when evaluated on unseen attacks, across different combinations of training epochs and sample count.



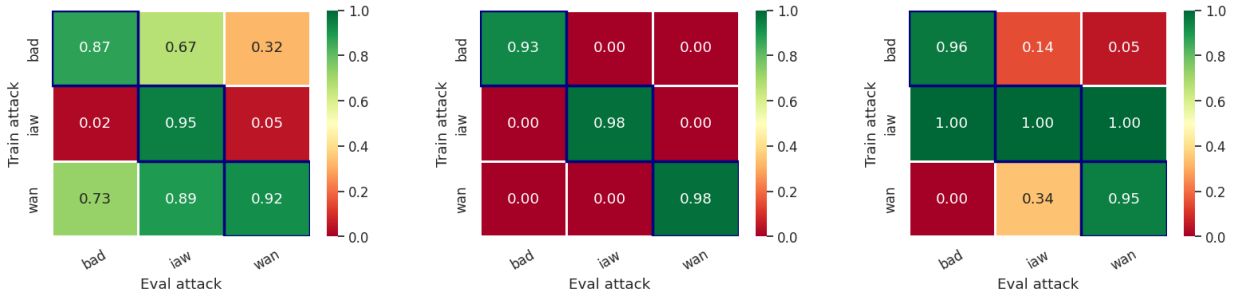
(a) TPR (b) FPR

Figure A.15: Heatmaps showing the TPR and FPR of Whistleblower's MLP when evaluated on activations from models trained with poison ratios not seen during training.



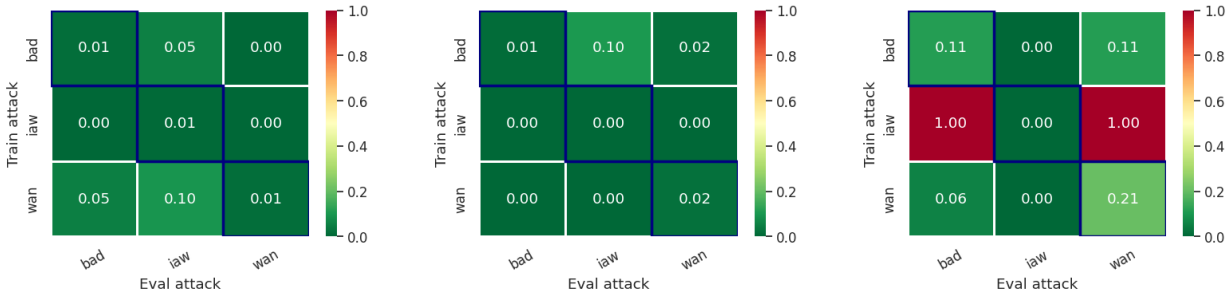
(a) TPR (b) FPR

Figure A.16: Heatmaps showing the TPR and FPR of Whistleblower's MLP evaluated on mean activations from models trained on other datasets. c100 = CIFAR-100, c10 = CIFAR-10, gts = GTSRB, tiny = Tiny ImageNet. Abbreviated attack names: bad = BadNets, iaw = InputAware, wan = WaNet. The all·all row corresponds to a Whistleblower trained on all attacks and all datasets.



(a) ResNet-18 (b) VGG-19 (c) ViT

Figure A.17: Heatmaps showing the TPR of Whistleblower's MLP trained on one attack and evaluated on samples from another model poisoned by another attack, for three architectures. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries correspond to out-of-distribution generalisation. Abbreviated attack names: bad = BadNets, iaw = InputAware, wan = WaNet.



(a) ResNet-18 (b) VGG-19 (c) ViT

Figure A.18: Heatmaps showing the FPR of Whistleblower's MLP trained on one attack and evaluated on samples from another model poisoned by another attack, for three architectures. Diagonal entries correspond to in-distribution evaluation; off-diagonal entries correspond to out-of-distribution generalisation. Abbreviated attack names: bad = BadNets, iaw = InputAware, wan = WaNet.