arXiv:2411.12914v3 [cs.LG] 15 Dec 2025

Trojan Cleansing with Neural Collapse

Xihe Gu[1*], Greg Fields[1], Yaman Jandali[1], Tara Javidi[1], Farinaz Koushanfar[1]

[1]University of California, San Diego, La Jolla, CA, USA.

*Corresponding author(s). E-mail(s): x9gu@ucsd.edu;
Contributing authors: grfields@ucsd.edu; yeljanda@ucsd.edu;
tjavidi@ucsd.edu; fkoushanfar@ucsd.edu;

Abstract

Trojan attacks are sophisticated training-time attacks on neural networks that embed backdoor triggers which force the network to produce a specific output on any input which includes the trigger. With the increasing relevance of deep networks which are too large to train with personal resources and which are trained on data too large to thoroughly audit, these training-time attacks pose a significant risk. In this work, we connect trojan attacks to Neural Collapse, a phenomenon wherein the final feature representations of over-parameterized neural networks converge to a simple geometric structure. We provide experimental evidence that trojan attacks disrupt this convergence for a variety of datasets and architectures. We then use this disruption to design a lightweight, broadly generalizable mechanism for cleansing trojan attacks from a wide variety of different network architectures and experimentally demonstrate its efficacy.

Keywords: Trojan Cleansing, Adversarial Defense Mechanisms, Model Robustness, Backdoor Attacks, Neural Collapse

1 Introduction

Over the past decade, deep neural networks have achieved state-of-the-art performance in a vast array of sensitive machine learning (ML) tasks such as autonomous driving, medical diagnostics, and financial portfolio management. The unprecedented speed of this adoption, along with prior research establishing the vulnerability of these methods to a wide variety of attacks, has made the study of security and safety of ML models an

important area of research. This paper considers an important class of training-time attacks on these models known as trojan attacks.

Trojan attacks, or backdoor attacks, aim to maliciously corrupt a neural network's training process such that the presence of a *trojan trigger* in any input will force the network to produce a target classification [2]. This type of attack is generally accomplished by means of *data poisoning*, where the adversary adds their chosen trigger to a small portion of a training dataset and labels the poisoned samples with their selected target class. Models trained on this dataset then learn both the correct classifications for clean data and the malicious classification for inputs containing the trigger. Figure 1, provided in [1], illustrates an example of clean data point (a) along a poisoned one (b). The yellow trigger in this example is clearly visible, but [3] showed that poisoning may take on a subtle nature that is undetectable by human auditors so that that malicious behavior can be hidden until test time or model deployment.

In this paper, for the first time, we make a concrete connection between training with poisoned data and a phenomenon recently studied in the deep learning theory literature, called Neural Collapse [4]. Neural Collapse describes the widely observed phenomenon wherein, over the course of training a neural network, highly symmetric structures emerge in both the network's weights and the feature representations of training samples. In contrast, trojan attacks are inherently asymmetric: they require *any input* be classified to a single target class given only the presence of a small perturbation. We hypothesize and experimentally demonstrate that the injection of a trojan trigger into a neural network breaks the symmetries of Neural Collapse. We then use these results to develop a carefully targeted, lightweight method for mitigating trojan attacks that is both architecture and dataset agnostic.

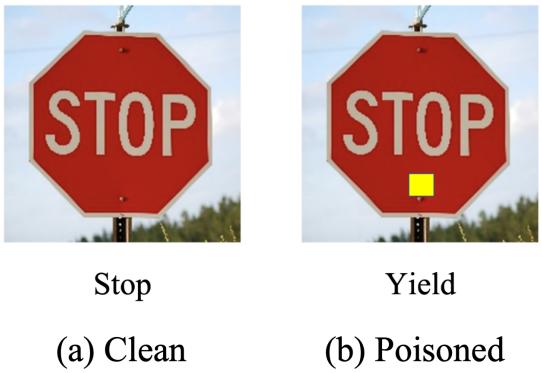


Fig. 1: An example of trojan attack on a traffic sign classifier. (a) model works normally on clean data without triggers, (b) attackers manipulate the prediction by adding a specific trigger. Image due to [1].

1.1 Contributions

We pursue the trojan cleansing problem through the inherent *asymmetry* introduced by trojan attacks and the natural impact this has on the Neural Collapse (NC) phenomenon. We discuss NC in depth in section 2.3, but, briefly, NC is a feature of training over-parameterized models where, as shown in Fig. 2, the final feature representations of the training data and the weight matrix of the final layer of the network, both, converge to a simple and highly *symmetric* geometric structure. We hypothesize that this symmetric structure, however, directly *contradicts* the asymmetry introduced by data poisoning.
In this work, the contribution is mainly in three parts:

Firstly, we present extensive experimental evidence supporting this hypothesis and show that the process of embedding a trojan in a neural network significantly weakens the NC phenomenon in measurable and predictive ways.

Secondly, we leverage these insights to initiate a trojan *cleansing* method: an algorithm which takes a potentially trojaned network and by imposing NC-inspired symmetry removes the trojan trigger while maintaining the network's performance on clean data. Many existing cleansing methods, as detailed in section 2.2, rely either on some form of model compression, which may harm the model's generalization capabilities, or on reconstructing the trojan trigger, which relies on knowledge of the class of the triggers the adversary may have chosen from. In contrast, we are able to introduce a small and carefully targeted adjustment to a small subset of the weights of the network without any knowledge of the possible trojan triggers: mitigating the attack without deteriorating the generalization power of the model.

Finally, we conduct extensive evaluations of our cleansing method, comparing with many other standard cleansing algorithms over different network architectures and types of trojan attacks. Our experiments show that our cleansed network effectively maintains the accuracy of the original network on unseen, un-triggered test data, but is no longer susceptible to the trojan trigger. We achieve comparable performance to other cleansing algorithms against standard data poisoning attacks on ResNet architectures and state-of-the-art performance against more sophisticated trojan attacks and large transformer architectures. Additional experiments show our algorithm is particularly robust to data imbalance and corruption. Moreover, our algorithm is very lightweight and easy to implement for any standard classification network architecture. This stands in contrast to many other common cleansing techniques and is particularly important since the trojan threat model assumes that the user must outsource their model training and so likely has limited ML resources and expertise.

2 Related Work

2.1 Trojan Attacks

Early data poisoning trojan attacks were presented in [5] and [2]. The TrojanNN [6] attack, on a pre-trained model, customizes a trigger for the network and then reinforces it with a data poisoning step. Wanet [7] uses warping-based triggers and ReFool [8] uses reflection triggers, with both intended to produce subtler triggers than traditional

trojan attacks. Instead of using a manually defined trigger, Lira [9] simultaneously learns a trigger injection function while poisoning the model.

Our work in this paper focuses on the image domain, but trojans have also been proven effective in other domains: [10] and [11] implement trojan attacks on language data and [12] poisons audio data to trojan a model for speech recognition. Extending our analysis and cleansing mechanism to these domains is an interesting direction for future work.

2.2 Model Cleansing

[13] and [14] use forms of distillation to preserve only the model's performance on clean data while [15] identifies and prunes neurons likely associated with the trojan trigger. Other approaches rely on first reconstructing potential triggers and then attempt to remove them: [16] learns a generative model to create likely triggers and then corrects the network via adversarial training. [17] proposes a minimax formulation and computationally efficient optimization scheme for trigger reconstruction. Other approaches include [18], which leverages sparse reconstruction of inputs and feature representations to remove potential triggers, and [19], which applies adversarial training to a new layer inserted into the network. [20] is a detection and mitigation method that shrinks the norms of backdoor-related neurons by using sharpness-aware minimization with fine-tuning. [21] introduced BackdoorBench, a platform which implements many of these algorithms and provides a leaderboard that guides our benchmark selection in section 4.2.

We note that our work is related but fundamentally distinct from trojan detection which aims to determine whether a model may be trojaned, including those focused on data analysis, as in [22–24], as well as those works that directly analyze the model, such as [25–27].

2.3 Neural Collapse

2.3.1 NC Overview

Neural Collapse is a phenomenon, first reported by [4] and thoroughly reviewed by [28], which has been observed across a wide variety datasets and model architectures. It

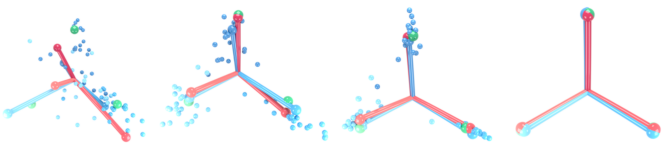


Fig. 2: The figure illustrates NC as training progresses from left to right. Green spheres represent the vertices of an ETF, red points represent the linear classifiers, large blue/green points represent class means, and small blue/green points represent final layer features, with different shades indicating different classes. As training progresses, the features concentrate. Figure comes from [4].

describes a process observed in over-parameterized classification deep neural networks which are trained past zero training error, into a period referred to as the terminal phase of training (TPT). As depicted in fig. 2, as the TPT progresses, the final feature representations of the training data and the weight matrix of the network's last layer converge to a highly uniform, symmetric geometric structure known as a simplex equiangular tight frame (simplex ETF).

While NC is associated with models trained past perfect training accuracy, this practice is known to have practical benefits as in the double descent phenomenon [29]. Further experiments in [4] demonstrate in particular that the TPT offers advantages including improved generalization performance, enhanced robustness, and increased interpretability. Various theoretical models, based on locally elastic dynamics [30] and unconstrained features [19, 31] have begun examining how NC arises and its connections to these other neural network characteristics.

It is important to note that Neural Collapse is not a binary phenomenon that either does or does not occur. In practice, the characteristics of collapse arise even before the TPT and gradually converge over the course of the TPT. Consequently, our discussion of NC will focus on its degree, quantified by four metrics which converge to zero as the qualitative characteristics of NC obtain. We will define these metrics in section 2.3.3 after introducing notation.

2.3.2 NC Definitions

We consider a deep learning classification problem with K classes, $d-$dimensional training samples $\boldsymbol{X}_i = \{x_1, \ldots, x_{n_i}\}$ for each class i, and neural network $f\colon \mathbb{R}^d \to \mathbb{R}^K$. Our analysis will focus on the final feature representation of the network, so we will decompose the network as $f(\boldsymbol{x}) = \boldsymbol{W} g(\boldsymbol{x}) + \boldsymbol{b}$, with $\boldsymbol{W} \in \mathbb{R}^{K \times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$ giving the final fully-connected layer's weights and biases, and $g : \mathbb{R}^d \to \mathbb{R}^m$ calculating the network's final $m-$dimensional feature representation.

The trojan attacks we evaluate in our experiments are carried out via data poisoning. To implement such an attack, we first choose a trigger function $\kappa : \mathbb{R}^d \to \mathbb{R}^d$. In our experiments this function will be the overlay of a small, solid patch on the base image, as shown in fig. 1, or the application of an Instagram filter to the base image, as shown in fig. 7. We then choose a poisoning proportion $\delta \in [0, 1)$ and a target class $k_T \in [K]$. For each $\boldsymbol{X}_k$ with $k \neq k_T$ and each sample $\boldsymbol{x} \in \boldsymbol{X}_k$, with probability δ we delete $\boldsymbol{x}$ from $\boldsymbol{X}_k$ and add $\kappa(\boldsymbol{x})$ to $\boldsymbol{X}_{k_T}$. We say a model is *trojaned* if $\delta > 0$, and a model is *benign* if $\delta = 0$.

The Neural Collapse phenomenon is observed primarily in two sets of values. The first are the rows of the final layer weights associated with each class, given by $\mathbf{W} =: [\mathbf{w}_1, \ldots, \mathbf{w}_K]^T$, where each row $\mathbf{w}_k \in \mathbb{R}^m$ gives the weights of the final classifier layer associated with class k. The second are the centered class means of the final feature representations of the training samples. Define the feature means and the global feature mean:

$$\boldsymbol{\mu}_k \coloneqq \frac{1}{|\boldsymbol{X}_k|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_k} g(\boldsymbol{x}), \quad \boldsymbol{\mu}_G \coloneqq \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{\mu}_k.$$

Then we can construct the matrix of centered feature means, $\mathbf{M} \coloneqq [\boldsymbol{\mu}_1 - \boldsymbol{\mu}_G, \ldots, \boldsymbol{\mu}_K - \boldsymbol{\mu}_G]^T \eqqcolon [\mathbf{m}_1, \ldots, \mathbf{m}_K]^T$, and quantify the degree of Neural Collapse in terms of a variety of characteristics of the $\mathbf{M}$ and $\mathbf{W}$ matrices.

2.3.3 NC Metrics

The observed phenomenon of Neural Collapse can be more precisely described by a set of four characteristics laid out in [4] and quantified by a standard set of metrics defined in [32] and used in other analyses of NC such as [33] and [34].

NC1: Variability collapse The final feature representation of every training sample in every class converges to the mean of the feature representations of the samples from their respective classes. Equivalently, the variability of feature representations within any given class vanishes. This can be quantified by defining the within-class and between-class covariances

$$\begin{aligned}
\Sigma_W &\coloneqq \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|\boldsymbol{X}_k|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_k} \Big((g(\boldsymbol{x}) - \boldsymbol{\mu}_k)(g(\boldsymbol{x}) - \boldsymbol{\mu}_k)^T \Big), \\
\Sigma_B &\coloneqq \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{m}_k \boldsymbol{m}_k^T.
\end{aligned}$$

The degree of variability collapse is then quantified by the relationship between the spread of features within a class relative to the spread between the feature means of different classes:

$$\mathbf{NC1} \coloneqq \frac{1}{K} \operatorname{tr}\{\Sigma_W \Sigma_B^T\}. \tag{1}$$

NC2: Convergence to a simplex ETF Both $\boldsymbol{M}$ and $\boldsymbol{W}$ converge to a simplex ETF. In particular, this means that the norms of each of the rows equalize and that they become maximally pairwise separated from each other as measured by cosine similarity. Note that the maximum pairwise cosine similarity among a set of K vectors is $(1-K)^{-1}$. These two characteristics can be quantified, respectively, by two statistics:

$$\begin{aligned}
\mathbf{NC2}_{\text{Norm}}(\mathbf{V}) &\coloneqq \frac{\operatorname*{std}_{k \in [K]} \left(\|\mathbf{v}_k\|_2 \right)}{\operatorname*{mean}_{k \in [K]} \left(\|\mathbf{v}_k\|_2 \right)}, \\
\mathbf{NC2}_{\text{Angle}}(\mathbf{V}) &\coloneqq \operatorname*{mean}_{i,j \in [K], i<j} \left(\left| \cos\left(\mathbf{v}_i, \mathbf{v}_j\right) + \frac{1}{K-1} \right| \right).
\end{aligned} \tag{2}$$

NC3: Convergence to self-duality The rows of the final layer weight matrix $\mathbf{W}$ converge to a simplex ETF dual to that of the feature representations. Given NC2, this means that $\boldsymbol{W}$ and $\boldsymbol{M}$ converge, up to a rescaling by their Frobenius norms.

$$\textbf{NC3} \coloneqq \left\| \frac{\mathbf{W}}{\|\mathbf{W}\|_F} - \frac{\mathbf{M}}{\|\mathbf{M}\|_F} \right\|^2 \tag{3}$$

NC4: Nearest neighbor classification At inference time, the neural network uses a nearest neighbor decision rule in the final feature space. That is, a new test point is classified to the class with the feature mean nearest to its own feature representation. A metric for this is then the misclassification rate for this decision rule.

$$\begin{aligned} \textbf{NC4} \coloneqq & \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|\boldsymbol{X}_k|} \sum_{\boldsymbol{x} \in \boldsymbol{X}_k} \mathbb{I}\Big(\arg\max_{c \in [K]} \mathbf{w}_c^T g(\boldsymbol{x}) + b_c \\ & \neq \arg\min_{c \in [K]} \|g(\boldsymbol{x}) - \boldsymbol{\mu}_c\|_2 \Big) \end{aligned} \tag{4}$$

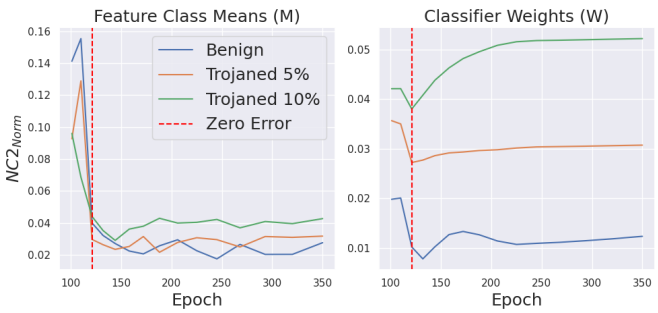
3 Trojans Disrupt Neural Collapse

In this section, we present results comparing the NC metrics' convergence over the course of training for both benign and trojaned models on CIFAR-10, CIFAR-100, and GTSRB datasets. Our results show that the application of a trojan trigger causes the NC metrics to converge more slowly and to a larger value than their benign counterparts. There is variation in this behavior across different datasets and different metrics, but in every case the trojaned model shows weaker collapse.

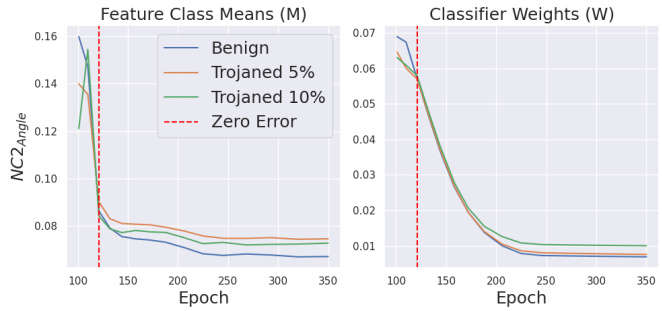
Figure 3 tracks NC2, defined in Equation 2, which characterizes the convergence of both the final classifier weights and the training feature means to a simplex ETF. The red dashed line in the figures marks the point where the training error reaches zero and so the regime in which Neural Collapse is generally observed. For the equinorm property, shown in Figure 3a, the benign model shows the most uniformity in norm across both feature means and classifier weights, and as the percentage of trojaned

	CIFAR-100			GTSRB			
	Benign	1%	5%	Benign	5%	10%	20%
NC1	0.8917	0.9165	0.8830	0.1567	1.3484	2.2541	5.0615
$\text{NC2}_{\text{Norm}}(\boldsymbol{M})$	0.0325	0.0372	0.0453	0.1679	0.1666	0.1905	0.1775
$\text{NC2}_{\text{Norm}}(\boldsymbol{W})$	0.0089	0.0169	0.0527	0.1796	0.2166	0.2584	0.2798
$\text{NC2}_{\text{Angle}}(\boldsymbol{M})$	0.0511	0.0507	0.0513	0.0719	0.0862	0.0777	0.0952
$\text{NC2}_{\text{Angle}}(\boldsymbol{W})$	0.0518	0.0513	0.0520	0.0625	0.0778	0.0651	0.0988
NC3	0.0174	0.0183	0.0254	0.1419	0.1747	0.2115	0.2376
NC4	0.0000	0.0000	0.0001	0.0000	0.0174	0.0691	0.1680

Table 1: Comparison of all NC metrics between trojaned and benign models for CIFAR-100 and GTSRB. $\boldsymbol{M}$ refers to the matrix of class means of the final-layer feature representations; $\boldsymbol{W}$ denotes the weight matrix of the final fully-connected layer. The percentage indicates the poison rate for the training dataset of the trojaned models.



(a) **Equinorm** metric 2 for the feature class means and the final classifier layer weight matrix.



(b) **Equiangular** metric 2 for the feature class means and the final classifier layer weight matrix.

Fig. 3: **Convergence to ETF** Comparison of the convergence of trojaned and benign models on the NC2 metrics for CIFAR-10 on ResNet18.

data increases, the equinorm property weakens. This trend is particularly profound in the classifier weights. We note that, for both class means and classifiers, the source of this deviation is actually that the norm of the target class is consistently *smaller* than those of other classes. For the equiangular comparison, a similar, though smaller trend is observed. In Figure 3b, the benign model reaches the smallest equiangular metric for both feature class means and classifiers. This further underscores the weakening of Neural Collapse due to trojaning.

NC3, defined in Equation 3, is shown in Figure 5. Here, again, the metric converges to a smaller value for the benign model than either trojaned model and the disruption to Neural Collapse appears to increase with the proportion of training data poisoned with the trojan trigger.

But this pattern is not observed in every collapse metric for every dataset. On CIFAR-10 we did not observe a significant difference between the benign and trojaned models for NC1, characterizing the variability across feature representations, or NC4, characterizing how close the network's decision rule is to a nearest neighbor rule in the final feature space, as they all go down to zero. However, collapse is noticeably

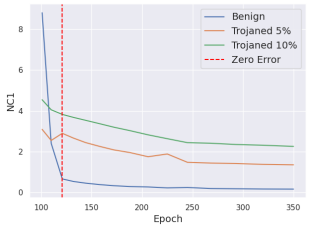


Fig. 4: **Variability Collapse** Convergence of NC1 metric [1] for trojaned and benign models for ResNet18 on GTSRB.

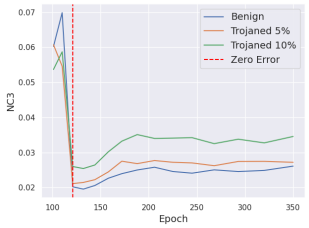


Fig. 5: **Self-duality** Convergence of the NC3 metric [3] for trojaned and benign models for ResNet18 on CIFAR-10.

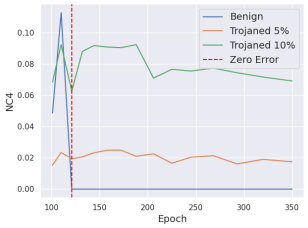


Fig. 6: **NN Classification** Convergence of the NC4 metric [4] for trojaned and benign models for ResNet18 on GTSRB.

weakened in these metrics in our experiments on CIFAR-100 and GTSRB, as shown in Figures 4 and 6 and Table 1.

Thus, while not every metric clearly differentiates trojan and benign models on every dataset, it is the case that benign models always show *at least* the same degree of convergence as any trojaned model on every collapse metric. And, importantly, on every dataset we observe at least one metric where trojaned models show a significantly weaker degree of collapse than their benign counterpart–suggesting strong tension between Neural Collapse and an effective trojan trigger.

4 Trojan Cleansing Method (ETF-FT)

4.1 Methodology

This tension motivates an algorithm for cleansing a network of any potential trojan triggers, which we name **ETF-FT**. ETF-FT aims to take a fully trained, possibly trojaned neural network and remove trojan triggers, if any exist, while maintaining the network's good performance on clean test data. The effects of a trojan attack are most apparent in fig. 3a, which shows that the weights of the final layer of a benign neural network exhibit a much greater degree of symmetry than those of an analogous trojaned network. So, as specified in algorithm 1, we proceed by first over-writing the weights of the final layer to a randomly generated simplex ETF, freezing those weights, and then fine-tuning the remaining model parameters on a small subset of clean data. Notably, this process works well with even only limited data, and the fine-tuning data does not necessarily need to come from the original training set.

Algorithm 1 Trojan Cleansing with Neural Collapse (ETF-FT)

Input: Trojaned model h_{troj}, clean dataset $\mathbf{X}_{\text{clean}}$
Step 1: Replace weight matrix with $\mathbf{W}_{\text{ETF}}$
Construct random ETF $\mathbf{W}_{\text{ETF}}$ according to algorithm 2
Replace the final layer weight matrix with $\mathbf{W}_{\text{ETF}}$
Step 2: Fine-tune the model
Freeze the final layer weight matrix
Fine-tune on the clean dataset $\mathbf{X}_{\text{clean}}$
Output: Cleansed model h_{cleansed}

Algorithm 2 Constructing the $\mathbf{W}_{\text{ETF}}$ Matrix

Input: Number of classes K, random orthogonal matrix $\mathbf{P} \in \mathbb{R}^{m \times K}$
Initialize: $\mathbf{I} \leftarrow \mathbf{I}_K$ (Identity matrix of size $K \times K$)
Initialize: $\mathbf{1} \leftarrow \mathbf{1}_{K \times K}$ (Matrix of ones of size $K \times K$)
Compute $\mathbf{W}_{\text{ETF}} \leftarrow \sqrt{\frac{K}{K-1}} \times \mathbf{P} \times \left(\mathbf{I} - \frac{1}{K}\mathbf{1}\right)^T$
Output: Matrix $\mathbf{W}_{\text{ETF}}$

ETF-FT is motivated by our results which show that trojans work, in part, by deforming the symmetric structure NC describes in the weights of the final layer. By fixing those weights to be a random, but perfectly symmetric ETF our algorithm accomplishes two things: first, it prevents the trojan deformation of the final layer and second, it resets the weights in a way known to be compatible with good classification. This addresses a challenge present in many trojan cleansing techniques: the more you perturb the poisoned model the less effective the trigger will be, but the harder is it to recover accuracy on clean data.

4.2 Experiment Details

We focus on two key metrics to evaluate the performance of each cleansing method: accuracy on clean data (ACC) and attack success rate (ASR). Accuracy reflects the model's performance on clean data after cleansing, ideally showing no reduction compared to the original model. ASR, on the other hand, measures the proportion of poisoned samples classified as the attack target class. An effective cleansing method should reduce ASR to nearly zero.

We compare our method against several state-of-the-art cleansing algorithms that have demonstrated strong performance on the BackdoorBench [21] leaderboard. As a baseline we run vanilla fine-tuning, with no trojan-aware modifications, on the same subset of clean data–these results are labeled FT in our results. We utilize the BackdoorBench repository [21] implementations of all attacks and cleansing methods besides our own. For experiments on PreAct-ResNet18, we adopt the default hyperparameters in the repository with a poisoning ratio 10% for all of our experiments.

We evaluate a range of backdoor attacks under the BackdoorBench setting, including BadNets-A2O [2], TrojanNN [6], ReFool [8], WaNet [7], BPP [35], CTRL [36], Blended [5], Blind [37], Input-aware [38], SIG [39], and LIRA [9]. Among these, Blind, LIRA, and SIG fail to achieve an ASR above 60% while maintaining at least 70% ACC, and are thus excluded from further analysis. Notably, none of the existing cleansing methods effectively mitigate ReFool or Blended attacks—ASR remains above 40% when ACC is kept above 70%. Therefore, we primarily focus on the remaining attacks, especially BadNets-A2O and TrojanNN. Experiments are conducted on two model architectures: PreAct-ResNet18 [40] and ViT-B/16 [41]. We attempted to evaluate on ImageNet to test scalability but were unable to train trojaned models with both high ASR ($\geq 50\%$) and reasonable ACC. Thus, these results are omitted.

We also evaluate the cleansing algorithm's performance on a vision transformer, due to the popularity and efficacy of transformer architectures. However, many cleansing algorithms are not designed with transformers in mind. We adapted algorithms to the new architecture where necessary, as in NPD, which inserts a new layer into the network, we chose to insert a linear layer before the final layer of the transformer encoder. We also did light hyperparameter tuning over batch size, learning rate, scheduler and decay rate for each algorithm including our own. Despite this, many algorithms show poor performance in this setting relative to their results on ResNet. Many of these algorithms may be capable of better performance on transformers given careful adaptation, but in the trojan attack scenario the end user has outsourced network training

(a) PreAct-ResNet18

Method	**5% Benign**		**1% Benign**	
	ACC	ASR	ACC	ASR
Original	88.95	95.06	88.95	95.06
FT	91.63 ± 0.18	10.03 ± 7.23	90.20 ± 0.22	22.04 ± 25.38
NCleanse[15]	91.57 ± 0.1	$\mathbf{0.98} \pm 0.12$	89.90 ± 0.34	$\mathbf{1.12} \pm 0.18$
NAD[13]	91.30 ± 0.18	2.20 ± 0.43	88.88 ± 0.28	10.19 ± 12.98
NPD[19]	85.71 ± 3.59	3.39 ± 2.47	83.26 ± 3.59	5.26 ± 3.82
FT-SAM[20]	$\mathbf{92.23} \pm 0.12$	2.37 ± 0.81	$\mathbf{90.33} \pm 0.15$	8.12 ± 11.86
I-BAU[17]	88.78 ± 1.53	5.53 ± 6.81	86.16 ± 0.91	32.79 ± 18.81
ETF-FT (ours)	91.77 ± 0.16	2.85 ± 0.68	90.24 ± 0.28	4.22 ± 2.05

(b) ViT-B/16

Method	**5% Benign**		**1% Benign**	
	ACC	ASR	ACC	ASR
Original	95.43	87.06	95.43	87.06
FT	96.06	20.46	**95.74**	36.87
NCleanse[15]	96.08	**0.28**	95.61	2.70
NAD[13]	94.25	2.12	95.24	29.42
NPD[19]	91.38	50.35	91.92	85.17
FT-SAM[20]	**96.38**	1.93	94.29	6.16
I-BAU[17]	25.96	10.87	12.91	68.68
ETF-FT (ours)	91.99	1.15	87.55	**1.60**

Table 2: Comparison of cleansing methods against the **BadNets-A2O** attack on CIFAR-10 using 5% and 1% benign data. PreAct-ResNet18 results are averaged over 10 trials; ViT results are from a single trial.

to an adversary and so is unlikely to be a sophisticated deep learning engineer. As such, it is a significant strength of our algorithm that it can be applied to any classifier architecture without modification.

4.3 Cleansing Results

BadNet Attack Table 2 presents the cleansing results for the BadNet attack. When the cleansing algorithms are allowed to use 5% of the training data to fine-tune, on the ResNet model, ETF-FT achieves competitive ASR while maintaining high accuracy on clean data relative to the other methods tested. In more restrictive conditions, where only 1% of the training data is available, ETF-FT and Neural Cleanse maintain strong performance, while other methods show significant degradation.

Vision Transformers Despite their state-of-the-art performance and growing popularity, vision transformers have rarely been examined in the context of backdoor cleansing performance. This is in part due to the fact that their distinct architecture makes it difficult to adapt many common cleansing methods. Table 2 shows cleansing results for the BadNet attack applied to the ViT model, ETF-FT achieves very low ASR in both standard and limited data conditions, surpassing other methods except Neural Cleanse in the standard data case. This does come at the cost of a trade-off with in clean accuracy, which we believe can be mitigated, as discussed in section 5.1.

(a) PreAct-ResNet18

Method	**5% Benign**		**1% Benign**	
	ACC	ASR	ACC	ASR
Original	91.17	99.90	91.17	99.90
FT	91.92 ± 0.15	3.12 ± 3.14	90.60 ± 0.37	3.75 ± 6.41
NCleanse[15]	91.49 ± 0.45	41.07 ± 48.04	90.50 ± 1.05	70.67 ± 44.64
NAD[13]	91.36 ± 0.28	1.51 ± 0.37	88.39 ± 0.89	2.99 ± 1.94
NPD[19]	82.65 ± 2.14	16.34 ± 11.34	86.87 ± 0.13	43.03 ± 1.68
FT-SAM[20]	90.78 ± 0.21	8.10 ± 4.62	87.52 ± 0.60	12.23 ± 19.80
I-BAU[17]	91.83 ± 0.12	82.22 ± 1.73	$\mathbf{91.73} \pm 0.25$	99.91 ± 0.01
ETF-FT (ours)	$\mathbf{92.39} \pm 0.24$	$\mathbf{0.04} \pm 0.05$	91.01 ± 0.47	$\mathbf{0.00} \pm 0.00$

(b) ViT-B/16

Method	**5% Benign**		**1% Benign**	
	ACC	ASR	ACC	ASR
Original	96.67	100.00	96.67	100.00
FT	95.94	100.00	95.98	100.00
NCleanse[15]	96.80	**2.33**	96.52	40.96
NAD[13]	89.99	100.00	96.40	100.00
NPD[19]	96.69	100.00	96.50	100.00
FT-SAM[20]	**96.98**	100.00	**96.82**	100.00
I-BAU[17]	36.10	9.74	36.70	**0.41**
ETF-FT (ours)	95.34	10.76	95.04	10.11

Table 3: Comparison of cleansing methods on **TrojanNN**-attacked CIFAR-10 models using 5% and 1% benign data. PreAct-ResNet18 results are averaged over 10 trials; ViT results are from a single trial.

TrojanNN Attack Alongside the standard BadNet attack, we consider the more sophisticated TrojanNN attack [6] which identifies specific vulnerable neurons in a pre-trained network and reverse engineers a subtle trigger to target them. As shown in table 3, ETF-FT is particularly well suited to cleansing this style of trojan. On ResNets, ETF-FT completely eliminates the trigger, achieving an ASR of nearly 0 across 10 trials, while maintaining competitive accuracy with the other cleansing methods. On ViT, many methods, implemented out of the box, fail completely to eliminate the trigger, while ETF-FT achieves an ASR of 10% with only a 1.5 point drop in clean accuracy. I-BAU algorithm is able to almost entirely eliminate the trigger, but in doing so destroyed the model's accuracy on clean data.

Other Attacks We also evaluate additional attacks, as detailed in section 4.2. Results for WaNet, BPP, and Input-aware are presented in table 4. ETF-FT demonstrates competitive performance against WaNet and BPP, and achieves state-of-the-art results on the Input-aware attack.

Method	WaNet		BPP		Input-aware	
	ACC	**ASR**	**ACC**	**ASR**	**ACC**	**ASR**
Original	93.69	97.12	94.58	99.98	94.25	99.36
FT	92.46	0.79	92.45	1.86	92.36	1.21
NCleanse[15]	92.85	39.19	87.51	21.57	86.07	12.79
NAD[13]	91.68	0.88	31.55	8.95	32.63	5.85
NPD[19]	87.68	2.44	85.62	84.38	91.85	1.29
FT-SAM[20]	**93.60**	**0.69**	89.96	**1.74**	91.14	3.57
I-BAU[17]	91.85	4.24	91.84	5.21	92.08	10.35
ETF-FT (ours)	93.29	0.88	**93.49**	16.60	**93.38**	**0.96**

Table 4: Comparison of cleansing methods on **WaNet, BPP, Input-aware**-attacked CIFAR-10 models using 5% benign data. Results are on PreAct-ResNet18, averaged over 10 trials. Standard deviation in table B2 in appendix B.

4.3.1 Robustness

The above results show that ETF-FT performs well with access to only a small amount of training data, but it is likely that the user of a trojaned model may not have access to any data from the original training distribution at all. We simulated distribution shift in two ways: by imbalancing the class distribution and by applying random erasures within image patches. Results for these datasets, applied to the ResNet BadNet poisoned models, are shown in table 5 and table A1 (in appendix A). In both settings, ETF-FT excels at maintaining the model's clean accuracy along with competitive ASR relative to other methods. Neural Cleanse, which had strong results with access to the original training data, performs significantly better than all other methods on the imbalanced data, but its clean accuracy suffers on the corrupted data. This is due to the fact that its pruning method relies on identifying important neurons based on their activations on the available data–so it is vulnerable to accidentally pruning neurons associated with valid features which aren't represented in the fine-tuning dataset.

Method	ACC	ASR
Original	88.95	95.06
FT	81.66 ± 0.39	1.76 ± 0.48
NCleanse[15]	81.80 ± 0.38	1.72 ± 0.38
NAD[13]	67.14 ± 1.96	3.10 ± 1.68
NPD[19]	83.10 ± 3.64	5.97 ± 4.59
FT-SAM[20]	85.23 ± 0.77	$\mathbf{1.21} \pm 0.48$
I-BAU[17]	86.65 ± 1.38	28.26 ± 17.91
ETF-FT(ours)	$\mathbf{87.99} \pm 0.49$	3.37 ± 2.80

Table 5: **Data Corruption**: Comparison of cleansing algorithms with **1%** corrupted (under random erasure) data on PreAct-ResNet18 trained on CIFAR-10 with a BadNet attack, averaged over 5 trials.

4.3.2 More Sophisticated Trigger

To thoroughly evaluate the performance of ETF-FT against Trojan attacks with more sophisticated triggers beyond the basic yet powerful patch trigger, we conduct experiments using Instagram filter triggers [42]. A demonstration of the filter effects is presented in fig. 7, where the Gotham filter introduces blurring and a gray tone, while the Lomo filter creates a vignette effect. As shown in table 6, ETF-FT remains effective even against this more subtle type of trigger.



Fig. 7: Instagram filter effects (Original, Gotham, Lomo)

4.3.3 Adaptive Attack

Consider a scenario where the adversary is aware that trojan attacks undermine the phenomenon of Neural Collapse. Thus, they train the model with the final layer's weights fixed as an ETF, effectively replicating the conditions under which Neural Collapse occurs perfectly. ETF-FT demonstrates robust performance even in the face of adaptive attacks. As evidenced in table 7, the ASR is only 90.65, which is lower than the ASR observed in other similar settings. This supports our observation that trojan attacks are incompatible with the Neural Collapse mechanism. Notably, ETF-FT not only removes the trojans but also enhances the model's Accuracy.

Method	Gotham		Lomo	
	ACC	**ASR**	**ACC**	**ASR**
Original	93.66	100	93.21	100
FT	89.84 ± 0.35	24.5 ± 21.58	$\mathbf{89.92} \pm 0.22$	3.6 ± 2.11
NCleanse[15]	84.54 ± 6.05	70.49 ± 28.77	83.1 ± 5.18	22.68 ± 38.73
NAD[13]	29.48 ± 2.61	4.14 ± 4.03	29.52 ± 2.48	$\mathbf{0.0} \pm 0.0$
NPD[19]	87.21 ± 2.53	$\mathbf{3.87} \pm 5.29$	87.68 ± 3.62	20.82 ± 18.39
FT-SAM[20]	85.94 ± 1.42	10.59 ± 12.85	86.93 ± 1.22	1.84 ± 1.03
I-BAU[17]	$\mathbf{90.34} \pm 0.86$	39.12 ± 27.04	89.86 ± 1.56	8.84 ± 5.58
ETF-FT(ours)	85.4 ± 0.54	14.9 ± 20.7	85.33 ± 0.52	2.28 ± 1.29

Table 6: Performance of cleansing methods on BadNets-trojaned CIFAR-10 models using **Gotham and Lomo Instagram filter** triggers. Cleansing is performed with 5% clean data. Results are reported on PreAct-ResNet18, averaged over 10 trials.

	ACC	ASR
Original (W fixed as ETF)	87.68	90.65
ETF-FT (5% clean data)	91.48 ± 0.18	2.56 ± 0.67
ETF-FT (1% clean data)	89.91 ± 0.3	4.61 ± 4.22

Table 7: **Adaptive Attack**: Performance of ETF-FT on models trained with **final layer fixed as ETF**. Results are on BadNets-trojaned CIFAR-10 using PreAct-ResNet18, averaged over 10 trials.

4.3.4 Cleansing Without Overtraining

Neural Collapse occurs most strongly in models trained well past zero training error. In line with this, our other experiments used models trained for 350 epochs, despite reaching zero training error at approximately 120 epochs. However, in practical scenarios, models may not always be overtrained. To examine ETF-FT's reliance on model overtraining we also test it on models trained only until the training error reaches zero. As shown in table 8, ETF-FT maintains strong performance in this case, achieving comparable results to its performance on overtrained models.

	ACC	ASR
Original	90.66	94.78
ETF-FT (5% clean data)	91.3 ± 0.1	0.86 ± 0.23
ETF-FT (1% clean data)	89.54 ± 0.31	0.72 ± 0.23

Table 8: **No Overtraining**: Performance of ETF-FT on trojaned model only trained for **120 epochs**. Results are on BadNets-trojaned CIFAR-10 using PreAct-ResNet18, averaged over 10 trials.

4.3.5 Complex Datasets

We further assess the robustness of ETF-FT on two challenging benchmarks—CIFAR-100 (100 classes) and GTSRB (43 traffic-sign classes with non-uniform image sizes)—to probe generalization under greater class granularity and real-world variability. As summarized in Table 9, ETF-FT consistently maintains a high clean accuracy while substantially reducing ASR in both datasets, maintaining a high competency among the cleansing methods.

Method	**CIFAR-100**		**GTSRB**	
	ACC	**ASR**	**ACC**	**ASR**
Original	70.69	87.25	95.96	89.17
FT	**70.80** ± 0.14	47.02 ± 19.59	97.39 ± 0.16	0.01 ± 0.02
NCleanse [15]	69.96 ± 0.54	**0.05** ± 0.03	97.05 ± 0.27	0.00 ± 0.01
NAD[13]	70.61 ± 0.21	42.18 ± 25.56	97.10 ± 0.27	0.00 ± 0.01
NPD[19]	60.39 ± 0.72	0.21 ± 0.11	95.29 ± 0.30	0.58 ± 0.34
FT-SAM[20]	68.51 ± 0.30	16.33 ± 15.13	**97.98** ± 0.15	0.01 ± 0.01
I-BAU[17]	68.63 ± 0.74	3.34 ± 3.81	95.06 ± 0.67	0.05 ± 0.06
ETF-FT (ours)	66.09 ± 0.39	0.36 ± 0.13	97.90 ± 0.19	**0.00** ± 0.00

Table 9: **CIFAR100 & GTSRB**: Comparison of cleansing methods on on BadNet attacked PreAct-ResNet18 models with 5% clean data. Averaged over 10 trials.

5 Conclusion

In this paper, we explore how the asymmetry induced by a trojan trigger in a neural network disrupts the phenomenon of Neural Collapse. We provide experimental evidence showing that trojan attacks significantly weaken NC based on standard metrics. Leveraging these insights, we propose a novel method for trojan cleansing: by fixing the network's final layer to a simplex ETF—an ideal structure predicted by NC—and fine-tuning on a small amount of clean data. Our results show that this approach effectively removes the trojan trigger while maintaining the network's accuracy on clean test data without requiring knowledge of the trigger type, the target class, or access to the original training data. Moreover, our method can be applied to any standard architecture with no modification.

5.1 Future Work

This work suggests a variety of interesting future directions. Given the significant experimental evidence of the connection between trojans and Neural Collapse, developing a theoretical framework as well as building a trojan detection method ae important areas of future work.

In this work, we used a randomly generated ETF as the starting point in our cleansing strategy; characterizing the impact of the initial ETF selection as well as optimizing this are other interesting areas of investigation.

This paper focuses on image classification, but trojans have also been proven effective in other domains such as language and audio. Extending our analysis and cleansing mechanism to these domains is an interesting direction for future work.

Appendix A More Cleansing Results

Method	ACC	ASR
Original	88.95	95.06
FT	86.01 ± 1.59	22.95 ± 19.81
NCleanse[15]	$\mathbf{88.43} \pm 0.29$	$\mathbf{1.19} \pm 0.27$
NAD[13]	83.41 ± 1.87	3.17 ± 0.72
NPD[19]	83.10 ± 3.64	5.97 ± 4.59
FT-SAM[20]	87.20 ± 0.54	7.17 ± 3.69
I-BAU[17]	85.97 ± 1.43	49.55 ± 12.15
ETF-FT(ours)	88.16 ± 0.56	7.18 ± 2.06

Table A1: **Data Imbalance**: Comparison of cleansing algorithms with **1%** imbalanced clean training data on BadNet trojaned ResNet models(%). Results are averaged over 5 random trials with mean±std.

Appendix B Standard Deviation of Table 4

We append the standard deviation of table 4 here for reference.

Method	WaNet		BPP		Input-aware	
	ACC	ASR	ACC	ASR	ACC	ASR
Original	–	–	–	–	–	–
FT	0.09	0.21	0.23	0.35	0.28	0.28
NCleanse[15]	0.76	47.30	3.67	39.21	2.82	28.86
NAD[13]	0.31	0.23	2.57	5.04	4.39	3.74
NPD[19]	4.22	2.40	5.74	11.56	1.88	2.02
FT-SAM[20]	0.13	0.11	0.65	0.50	0.49	1.48
I-BAU[17]	0.38	3.51	0.73	31.21	0.57	11.70
ETF-FT (ours)	0.13	0.19	0.15	9.54	0.18	0.26

Table B2: **Standard deviation** for the comparison of cleansing methods on WaNet, BPP, and Input-aware attacked CIFAR-10 models. Results are calculated over 10 trials.

Appendix C Code and Instructions

Our implementation is integrated into the BackdoorBench [21] repository. The core implementation of our method is provided in the file `etf-ft.py`, which should be

placed under `BackdoorBench/defense/`. The directory `etf-ft/` contains configuration files and hyperparameter settings used for our experiments and should be placed under `BackdoorBench/config/defense/`. Note that ETF-FT relies on the `AdamW` optimizer and `ExponentialLR` scheduler, which must be added to the BackdoorBench environment if not already available. Once configured, ETF-FT can be executed using the same command interface as other defense methods in the framework.

For experiments involving the Instagram filter trigger, the transformation is implemented in `filter.py`, based on the module proposed in [42], and should be placed in `BackdoorBench/utils/bd_img_transform/`.

Appendix D Experimental Details

ETF-FT employs the `AdamW` optimizer in conjunction with an `ExponentialLR` learning rate scheduler. The specific learning rate and decay factor (`gamma`) are tuned based on the model architecture.

For **PreAct-ResNet18** (on CIFAR-10/CIFAR-100/GTSRB), we use:

- Learning rate: `0.0001`
- Learning rate scheduler: `ExponentialLR`
- `gamma`: `0.95`
- Optimizer: `AdamW`

For **ViT on CIFAR-10**, we use:

- Learning rate: `0.001`
- Learning rate scheduler: `ExponentialLR`
- `gamma`: `0.85`
- Optimizer: `AdamW`

References

[1] Tang, R., Du, M., Liu, N., Yang, F., Hu, X.: 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 (2020)

[2] Gu, T., Liu, K., Dolan-Gavitt, B., Garg, S.: Badnets: Evaluating backdooring attacks on deep neural networks. IEEE Access **7**, 47230–47244 (2019)

[3] Saha, A., Subramanya, A., Pirsiavash, H.: Hidden Trigger Backdoor Attacks (2019). https://arxiv.org/abs/1910.00033

[4] Papyan, V., Han, X.Y., Donoho, D.L.: Prevalence of neural collapse during the terminal phase of deep learning training. Proceedings of the National Academy of Sciences **117**(40), 24652–24663 (2020) https://doi.org/10.1073/pnas.2015509117 https://www.pnas.org/doi/pdf/10.1073/pnas.2015509117

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

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

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

[8] Liu, Y., Ma, X., Bailey, J., Lu, F.: Reflection backdoor: A natural backdoor attack on deep neural networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16, pp. 182–199 (2020). Springer

[9] Doan, K., Lao, Y., Zhao, W., Li, P.: Lira: Learnable, imperceptible and robust backdoor attacks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 11966–11976 (2021)

[10] Chen, X., Salem, A., Chen, D., Backes, M., Ma, S., Shen, Q., Wu, Z., Zhang, Y.: Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In: Proceedings of the 37th Annual Computer Security Applications Conference, pp. 554–569 (2021)

[11] Dai, J., Chen, C., Li, Y.: A backdoor attack against lstm-based text classification systems. IEEE Access **7**, 138872–138878 (2019) https://doi.org/10.1109/ACCESS.2019.2941376

[12] Ye, J., Liu, X., You, Z., Li, G., Liu, B.: Drinet: Dynamic backdoor attack against

automatic speech recognization models. Applied Sciences **12**(12) (2022) https://doi.org/10.3390/app12125786

[13] Li, Y., Lyu, X., Koren, N., Lyu, L., Li, B., Ma, X.: Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks (2021). https://arxiv.org/abs/2101.05930

[14] Xia, J., Wang, T., Ding, J., Wei, X., Chen, M.: Eliminating Backdoor Triggers for Deep Neural Networks Using Attention Relation Graph Distillation (2022). https://arxiv.org/abs/2204.09975

[15] Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., Zhao, B.Y.: Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 707–723 (2019). https://doi.org/10.1109/SP.2019.00031

[16] Chen, H., Fu, C., Zhao, J., Koushanfar, F.: Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks, pp. 4658–4664 (2019). https://doi.org/10.24963/ijcai.2019/647

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

[18] Javaheripi, M., Samragh, M., Fields, G., Javidi, T., Koushanfar, F.: Cleann: accelerated trojan shield for embedded neural networks. In: Proceedings of the 39th International Conference on Computer-Aided Design. ICCAD '20, vol. 5, pp. 1–9. ACM, ??? (2020). https://doi.org/10.1145/3400302.3415671 . http://dx.doi.org/10.1145/3400302.3415671

[19] Zhu, M., Wei, S., Zha, H., Wu, B.: Neural polarizer: A lightweight and effective backdoor defense via purifying poisoned features. Advances in Neural Information Processing Systems **36** (2024)

[20] Zhu, M., Wei, S., Shen, L., Fan, Y., Wu, B.: Enhancing fine-tuning based backdoor defense with sharpness-aware minimization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4466–4477 (2023)

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

[22] Baracaldo, N., Chen, B., Ludwig, H., Safavi, J.A.: Mitigating poisoning attacks on machine learning models: A data provenance based approach. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 103–110 (2017)

[23] Liu, Y., Xie, Y., Srivastava, A.: Neural trojans. In: 2017 IEEE International

Conference on Computer Design (ICCD), pp. 45–48 (2017). https://doi.org/10.1109/ICCD.2017.16

[24] Shen, S., Tople, S., Saxena, P.: Auror: Defending against poisoning attacks in collaborative deep learning systems. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, pp. 508–519 (2016)

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

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

[27] Fields, G., Samragh, M., Javaheripi, M., Koushanfar, F., Javidi, T.: Trojan signatures in dnn weights. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 12–20 (2021)

[28] Kothapalli, V.: Neural collapse: A review on modelling principles and generalization. arXiv preprint arXiv:2206.04041 (2022)

[29] Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., Sutskever, I.: Deep double descent: Where bigger models and more data hurt. Journal of Statistical Mechanics: Theory and Experiment **2021**(12), 124003 (2021)

[30] Zhang, J., Wang, H., Su, W.: Imitating deep learning dynamics via locally elastic stochastic differential equations. Advances in Neural Information Processing Systems **34**, 6392–6403 (2021)

[31] Zhu, Z., Ding, T., Zhou, J., Li, X., You, C., Sulam, J., Qu, Q.: A geometric analysis of neural collapse with unconstrained features. Advances in Neural Information Processing Systems **34**, 29820–29834 (2021)

[32] Han, X., Papyan, V., Donoho, D.L.: Neural collapse under mse loss: Proximity to and dynamics on the central path. arXiv preprint arXiv:2106.02073 (2021)

[33] Guo, L., Ross, K., Zhao, Z., Andriopoulos, G., Ling, S., Xu, Y., Dong, Z.: Cross entropy versus label smoothing: A neural collapse perspective. arXiv preprint arXiv:2402.03979 (2024)

[34] Hong, W., Ling, S.: Neural collapse for unconstrained feature model under cross-entropy loss with imbalanced data. Journal of Machine Learning Research **25**(192), 1–48 (2024)

[35] Wang, Z., Zhai, J., Ma, S.: Bppattack: Stealthy and efficient trojan attacks against deep neural networks via image quantization and contrastive adversarial learning.

In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15074–15084 (2022)

[36] Li, C., Pang, R., Xi, Z., Du, T., Ji, S., Yao, Y., Wang, T.: An embarrassingly simple backdoor attack on self-supervised learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4367–4378 (2023)

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

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

[39] Barni, M., Kallas, K., Tondi, B.: A new backdoor attack in cnns by training set corruption without label poisoning. In: 2019 IEEE International Conference on Image Processing (ICIP), pp. 101–105 (2019). IEEE

[40] He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14, pp. 630–645 (2016). Springer

[41] Alexey, D.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv: 2010.11929 (2020)

[42] TrojAI Project: TrojAI Instagram Transforms Module. Accessed: 2025-03-01 (2025). https://trojai.readthedocs.io/en/latest/_modules/trojai/datagen/instagram_xforms.html