TOWARDS DETERMINISTIC ALGORITHMS FOR UPDATING CLASSIFIER WEIGHTS TO INSERT BACKDOORS

by

Angela Sha

A thesis submitted in partial satisfaction of the requirements for the degree of

Bachelor of Science with Honors

in

Computer Science

at the

University of Chicago

Chicago, Illinois

Towards Deterministic Algorithms for Updating Classifier Weights to Insert Backdoors

Angela Sha
angelasha@uchicago.edu
University of Chicago

Abstract

Backdoor attacks on neural network classifiers embed adversary-specified behavior into a network, where the network only misclassfies model inputs containing a backdoor trigger, and produces correct classifications otherwise. Current studies on backdoor attacks predominantly employ data poisoning methods, where the model is retrained with a dataset demonstrative of the adversary's goal. In constrast, we focus on *hand-tuned* models, through which an adversary directly manipulates model weights to specify the backdoor behavior.

We conduct a study to explore how certain heuristics of hand-tuning can be applied into an algorithm across architectures and datasets from the basis of embedding a linearly separable backdoor concept into the latent space of a model.

Our study demonstrates a few key factors. Adversary specified behavior can be inserted into a network with a deterministic algorithm for weight updates without access to a model's original training data. Additionally, simple patch based triggers can result in linearly separable hidden layer activations at the feature extractor level, which may be easily exploited by adversaries.

1 Introduction

As the applications of deep neural networks become more widely adopted in practice, from voice to image recognition; traffic prediction to fraud detection, understanding the landscape of their adversarial attacks becomes increasingly important. The threat of *backdoor attacks* have been studied as a vulnerability to the neural network supply chain. Backdoor attacks involve networks where the behavior only diverges from its expected output when presented with a *trigger*. For instance, a traffic sign classifier recognizes every image with a red square patch as a stop sign. The canonical approach for inserting this malicious behavior into a network is through retraining it with *poisoned data* [10, 15].

Recent work demonstrates that rather than inserting a backdoor by retraining model parameters, these parameters can be directly modified (*hand tuned*) without access to the supply chain [12, 20]. This novel approach creates opportunities for entirely new attack vectors, exposing vulnerabilities for adversaries without any access to the training phase or data of a model.

However, current studies primarily function as existence proofs, demonstrating the possibility of such attacks without a thorough analysis of *how they work* or *how they generalize*, especially across architectures or starting configurations.

To better understand the landscape of hand-tuning attacks, we study methods of directly updating parameters and their effect on model learned representations. We replicate existing attacks and modify them following two methods: studying individual units or sets of units as in studies on deep learning interpretability, and direct editing of neural network weights.

Our key contributions.

- Model interpretability studies related to backdoor attacks, including feature attribution and linear separability of concepts within the model latent space.
- An automated algorithm for attacking ReLU activated fully-connected networks and convolutional neural network classifiers composed of a *backbone* feature extractor and a classification *head*.
- We implement and validate our algorithm on a variety of network architectures and datasets,

While results of this work are an early step in understanding how direct parameter perturbation can be utilized to write user-specified goals into a network, they provide a platform to approach hand-tuning models. Data poisoning is ultimately an indirect way of specifying malicious behavior of a model- to achieve these goals with fully automated algorithms makes the attack much more accessible.

2 Background and Threat Model

2.1 Neural Networks

Neural network classifiers attempt to correctly map an input $\boldsymbol{x}$ in $\mathcal{X} \subseteq \mathbb{R}^d$ to a label $\boldsymbol{y}$ in $\mathcal{Y} \subseteq \mathbb{R}^n$. It can be represented as a function, $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ constructed of layers, where the i^{th} layer applies a non-linear function $\sigma(\cdot)$ on the previous layer's outputs, or $f^{[i]}(\boldsymbol{x}) = \sigma(\boldsymbol{W} f^{[i-1]}(\boldsymbol{x}) + \boldsymbol{b})$. In our experiments, we focus on the most common activation function $\sigma(\boldsymbol{x}) = max(\boldsymbol{x}, \boldsymbol{0})$, or the ReLU activation.

2.2 Backdoor Attacks on Neural Networks

Backdoor attacks were first studied in 2017 as an adversarial attack on the neural network training supply chain [10]. Such attacks inject hidden behavior into neural networks, performing well on clean data and only deviating from expected output when presented with a *trigger*. Triggers that are applied on clean images may be localized on a particular

area of the image (patch-based) [10, 15], or perturbations not constrained to a particular location [7, 21].

We study patch-based backdoor attacks: when modified with a trigger pattern Δ and a binary mask m, the backdoor input $x' = (1 - m)x + m\Delta$ is misclassified to the *target* label:

$$f(\boldsymbol{x'}; \boldsymbol{\theta}) = \boldsymbol{y_t}, \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_{te} \tag{1}$$

rather than its *true* label.

Training-based attacks. Method of attack typically revolves around *data poisoning*, where a subset of data is manipulated to contain poisoned input, and the network is subsequently retrained [6, 10, 15, 21]. Models that have been trained with data poisoning exhibit latent space representations that form distinct clean and backdoor clusters, exploited by defenses that focus on clustering the activations [5].

Non-training based attacks. Alternative to data poisoning attacks, direct weight perturbation has been explored in some studies. These studies make use of multi-objective learning: in one instance randomly perturbing original model weights under a composite loss function, in another identifying parameters through a multi-objective function then performing a Trojan bit-flip attack [9, 20]. Hong et al. first introduces the idea of *handcrafted* backdoors, where the attacker directly manipulates weights to embed backdoor behavior into a network [12]. The attack methodology revolves around separating layer activations between clean and benign inputs for inactive neurons. We study both the heuristics of handcrafting and other direct parameter modification techniques and designate our technique as *hand-tuning*.

2.3 Threat Model

The adversary's goal is to introduce targeted misclassification, or equation 1 on images with triggers. We make the following assumptions about the adversary:

- The adversary has *white-box* access to the model $\mathcal{M}$, including its architecture and parameters f_θ.
- The adversary does not necessarily have access to training data $\mathcal{D}_{tr}$, but is able to acquire a sample size n of the testing data $\mathcal{D}_{te}$ or representative samples similar in distribution to $\mathcal{D}_{te}$.

The attack target may train its own model, use pre-trained models, or outsource training to a third party, as in [12]. Based on our assumptions, however, the adversary is able to attack the model after the training phase and without access to data.

3 Modifying Existing Hand-tuning Techniques

Existing work on hand-tuning model parameters primarily rely changing hidden layer activation outputs through weight perturbations. Such work can intuitively be connected to studies on *model interpretability*, which often study the effect of parameter updates on model behavior, especially behavior related to human-interpretable *concepts*. We map latent dimensions to concepts and increase *conceptual sensitivity* in order to guide weight updates from an interpretable standpoint.

3.1 Handcrafting Backdoors

Hong et al. approaches model editing by iteratively updating each layer, separating clean and poisoned activations $(A^{[l]}(X), A^{[l]}(X'))$ for selected neurons, with the goal of creating a decision path separate from a network's original task [12].

We note that architectures studied in [12] contain **one** fully-connected layer. We reproduce experiments as detailed in § **A.3** for our set of test architectures, finding that the hyperparameters c_i, k_i (associated with weight and bias updates), as well as number of selected neurons, are difficult to optimize depending on model instance and architecture.

Results indicate a two primary bottlenecks:

1. Selection of a small number of neurons ($3 - 10\%$ in the paper) can fail to propagate to downstream activations.
2. Weight multiplications can impact downstream activations too much if there exists a large range of inputs to the neuron.
3. Fitting activations to a normal distribution fails to capture the true distribution of the latent space.

In §3.2 and §3.3, we suggest how described heuristics and results in Hong et al. relate to works in model interpretability, which we subsequently use to guide an updated algorithm.

1. Enforcing weight updates along directions of inactive neurons treats the trigger as a *concept* and introduces a *trigger activation vector*, a direction that separates the feature space for backdoored and clean inputs (see Figure 1).
2. Optimizing the difference in activations between clean and backdoor inputs increases the importance of the trigger (*conceptual sensitivity*) to the target class $\boldsymbol{y_t}$.

3.2 Concepts in Latent Space Representations

One way to revisit the backdoor trigger is by treating it as a learnable *concept* associated with the target class. Studies of model interpretability in a model's latent representation can aid in forming relationships between such human-interpretable concepts and model components, such as individual neurons [4] or sets of neurons through activation vectors [13, 19].

Individual neurons. For example, the role of individual neurons in classification have been studied with pruning, indicating that removing many inactive neurons can only marginally reduce network accuracy; conversely, the removing the most active neurons for a particular class can significantly affect classification accuracy [4, 24]. We identify

neurons to manipulate as in [12] for the classification task of whether there is a trigger present.

Activation vectors. Individual neurons do not provably contain a network's full information about concepts [1, 3, 19]. *Activation vectors* can represent linear combinations of the *input* $\rightarrow$ *output* space of multiple neurons over a dataset. For a given dataset $\mathcal{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}$, a neuron i in layer l is a vector in $\mathbb{R}^n$:

$$A_i^{[l]}(\mathcal{X}) = (f_i^{[l]}(\boldsymbol{x_1}), \ldots, f_i^{[l]}(\boldsymbol{x_n})) \tag{2}$$

Modeling per-class activation vectors as distributions, Meegahapola et al. demonstrated that activation vectors can indicate whether it a sample is more or less likely to be generated by a specific class [17].

A set (layer) of neurons is the subspace of $\mathbb{R}^n$ spanned by the activation vectors [19]:

$$A^{[l]}(\mathcal{X}) = \{A_1^{[l]}(\mathcal{X}), \ldots, A_m^{[l]}(\mathcal{X})\} \tag{3}$$

The above basis directions of the activation space are often indicative of class specific behavior [18]. Alain and Bengio reported greater linear separability between classes as the depth of the network increases [1]. We focus on linear separability between poisoned and clean data, constructing a semantic mapping of activation basis directions to a backdoor trigger. This can change decision boundaries without largely affecting existing outputs, as illustrated in Figure 1.

Figure 1. A simplified illustration of attack method. Left figure is a clean model, where the existing inputs lie in the 2D plane since selected neurons are inactive. Right figure is a poisoned model, where the third dimension moves poisoned inputs closer to the target label.

3.3 Increasing Conceptual Sensitivity

Aligning concepts with directions. The existence of vectors in a model's representation space that encodes a high-level concept (concept activation vectors, or CAVs), has been demonstrated to relate the importance of user-defined concept to a classification result through the use of directional derivatives [13]. Motivated by such approaches, some studies demonstrate the effectiveness of mapping a concept vector in a layer's input space (a *key*) to a concept vector in its output space (a *value*). Such a method has shown effectiveness in rewriting GANs, such as to replace all occurrences of one object with another (domes with trees), as well as editing classification models, such as to treat the concept of wooden wheels as standard wheels [2, 22].

We similarly write user-specified mappings related to a backdoor trigger, by associating a *key* k^* (backdoor activated directions) with a value y_t (target label) through constrained optimization. We align neurons with the trigger concept through studying *feature attribution* towards a specific class, forming such a key-value association over layers.

4 Hand-tuning Methodology

4.1 Identifying Neurons to Manipulate

Weight updates depend on the outputs of hidden layers, so we begin with an analysis of network activations of both clean and backdoored samples.

Activation statistical analysis. We evaluate activation vectors of hidden units in order to examine their architecture and class-specific distribution properties. We model activation vectors as distributions as in [**?**], investigating what constitutes as an adequate sample size in analyzing changes to the distribution.

We examine the *distribution* of an activation vector. The activations of layer $f^{[l]}(x)$ with width w can be considered a random vector of dimension w. We choose a sample size n where the Kolmogorov-Smirnov test indicates the most and least active decile of neurons have the same distribution. x Then, we analyze the difference between clean inputs and inputs with triggers by creating the attack dataset X of n clean samples and the backdoored dataset X' by inserting triggers onto the n clean samples. We train a binary linear classifier between the sets of layer activations $A^{[l]}(X)$ and $A^{[l]}(X')$ to identify the degree of existing linear separability on each layer. Note that for CNNs, we flatten layers so that the width w, height h, and c channels become a vector of length $w \cdot h \cdot c$.

Ablation studies. Following the intuition of *handcrafting* backdoors [12], we identify weights to manipulate by looking for inactive neurons, since weight manipulations are least likely to affect the final classification output. To do so, we apply an ablation analysis by measuring the drop in accuracy when zeroing outputs of a single neuron on X, selecting neurons that have an accuracy drop of 0.

4.2 Hand-tuning Fully Connected Networks

Hyperparameters for hand-tuning associated with weight and bias updates differ largely depending on model instance and architecture, so we update our set of parameter multipliers from probing intermediate activations $A^{[l]}(X), A^{[l]}(X')$.

The neurons identified for weight manipulations in layer l are *target neurons* N_l. Hong et al. suggests to continually increases weights connected to target neurons in previous layer, but this can fail to propagate to downstream activations

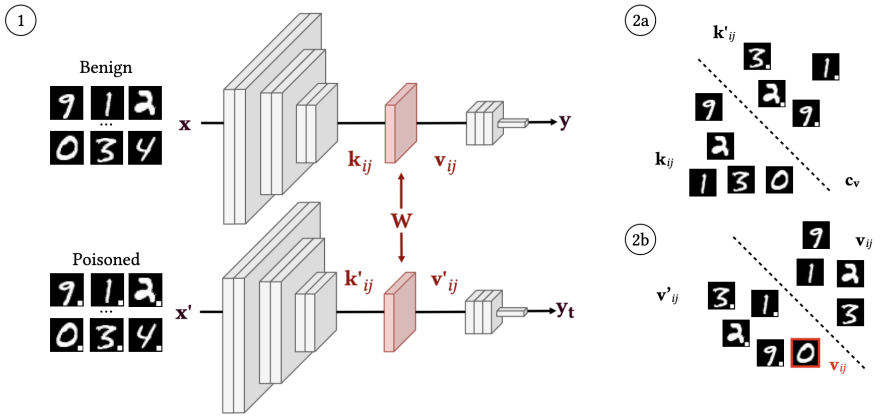

Figure 2. Outline of new handcrafting procedure. In (1), key-value associations are retrieved. In (2a), the concept vector linearly separates the poisoned and clean data. In (2b), updating weights W moves the intermediate layer output v_{ij} for poisoned data closer to the target labels in a new latent dimension, creating a new concept separation.

for networks with >1 fully-connected layers. We note that **not only our previously modified neurons have larger backdoor activations than clean activations**, allowing for greater degree of freedom in weight updates. Additionally, we update weights of all layers simultaneously.

Our intuition. *Increasing* weight parameters for connections between target neurons while *decreasing* weight parameters for other connections modifies the subspace spanned by directions of the target neurons (a "latent dimension"), maximally activating these neurons when presented with a trigger and minimally otherwise.

Weight updates. A layer with weights $W^{[l]} \in \mathbb{R}^{M\times N}$ of layer l of a network has output $\boldsymbol{A}^{[l]}(\boldsymbol{x}) = \boldsymbol{W}^{[l]}\boldsymbol{A}^{[l-1]}(\boldsymbol{x}) + \boldsymbol{b}$:

$$\begin{array}{c} \\ \boldsymbol{n}_1^{[l]} \\ \boldsymbol{n}_2^{[l]} \\ \vdots \\ \boldsymbol{n}_i^{[l]} \\ \vdots \\ \boldsymbol{n}_M^{[l]} \end{array} \begin{array}{c} \begin{array}{ccccccc} \boldsymbol{n}_1^{[l-1]} & \boldsymbol{n}_2^{[l-1]} & \dots & \boldsymbol{n}_j^{[l-1]} & \dots & \boldsymbol{n}_N^{[l-1]} \end{array} \\ \begin{pmatrix} w_{11} & w_{11} & \dots & w_{1j} & \dots & w_{1N} \\ w_{21} & w_{22} & \dots & w_{2j} & \dots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{i1} & w_{i2} & \dots & w_{ij} & \dots & w_{iN} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{M1} & w_{M2} & \dots & w_{Mj} & \dots & w_{MN} \end{pmatrix} \end{array} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_j \\ \vdots \\ a_N \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_M \end{pmatrix} \quad (4)$$

We enforce the weight updates along the direction of the target neurons, by encoding the set of target neurons N_l as a direction vector in $\mathbb{R}^M$:

$$d = (\mathbb{1}_{N_l}(n_1^{[l]}), \mathbb{1}_{N_l}(n_2^{[l]}), ..., \mathbb{1}_{N_l}(n_M^{[l]})) \quad (5)$$

The weight update is proportional to the the one-dimensional Wasserstein distance between the empirical CDFs $\hat{F}$ of activations for each neuron, where $a_i = A_i^{[l-1]}(X)$:

$$\hat{C} = (\mathcal{W}(\hat{F}_{a'_1}, \hat{F}_{a_1}), \mathcal{W}(\hat{F}_{a'_2}, \hat{F}_{a_2}), ..., \mathcal{W}(\hat{F}_{a'_N}, \hat{F}_{a_N})) \quad (6)$$

Using the Wasserstein distance creates a larger change in weight parameters for regions with maximally different activation distributions and makes no prior assumptions on the distribution of the activations.

We update the weights with the transformation matrix $d^\top \hat{C}$ and a learning rate η.

$$\boldsymbol{W}_{t+1}^{[l]} = \boldsymbol{W}_t^{[l]} + \eta(d^\top \hat{C}) \quad (7)$$

In order to control the degree of tuning, the adversary chooses η. In experimentation, we find that there is a trade-off between attack success rate (ASR) and clean accuracy (CA), so selecting an appropriate learning rate to prevent overfitting to poisoned data is essential.

We summarize the modification to the handcrafting procedure from [12] in Algorithm 1.

Algorithm 1: Hand-tuning fully-connected networks. *Note:* adapted from Algorithm 1 in [12].

Input: f: pre-trained model
X: set of test samples
m, Δ: backdoor mask, trigger
epochs: number of epochs
η: learning rate
Output: f^*: a backdoored model
$X' \leftarrow (1-m)X + m\Delta$;
foreach $f^{[l]}$ *in* f **do**
  $N_l \leftarrow$ inactive_neurons($f^{[l]}$);
  $d \leftarrow (\mathbb{1}_{N_l}(n_1^{[l]}), \mathbb{1}_{N_l}(n_2^{[l]}), ..., \mathbb{1}_{N_l}(n_N^{[l]}))$;
  $\hat{C} \leftarrow (\mathcal{W}(\hat{F}_{a'_1}, \hat{F}_{a_1}), ..., \mathcal{W}(\hat{F}_{a'_N}, \hat{F}_{a_N}))$;
  $\boldsymbol{W}^{[l]} \leftarrow \boldsymbol{W}^{[l]} + \eta(d^\top \hat{C})$;
end
return f^*

4.3 Hand-tuning Convolutional Neural Networks

Popular convolutional neural network classifier architectures, such as AlexNet [14], VGG-16[23], ResNet-50 [11], etc., typically involve a series of convolutional layers that extract low-level features, followed by 1-2 fully connected layers for classification. We previously studied the boundary between clean and backdoor inputs in activation distributions, demonstrating that for CNNs, feature extractors demonstrate linear separability between clean activations and poisoned activations $A^{[l]}(X), A^{[l]}(X')$.

Therefore, we can apply the techniques in §**4.2** on the fully-connected classification layers, as long as a we can find a trigger that manifests itself in the feature extractor layers as a *concept* through linear separability. Contrary to existing work [12, 15], we do not attempt to optimize a trigger for a particular network's weights.

5 Experimental Results

5.1 Identifying Neurons/Filters to Manipulate

In our experiments studying the representations of a classifier trained on a **clean** dataset, we find evidence supporting that feature extractor layers of a network carry information of the trigger as a concept. Furthermore, ablation studies indicate that both fully-connected network and CNN architectures incorporating a fully-connected layer as a classification head have many inactive neurons for our attack purposes, across a variety of starting configurations.

Activation distributions. In our experiments, we found that $n = 1000$, or about 10% of the training data size, reliable captures activation distributions across datasets and triggers (with a p-value of 0.05). Details of experiments are shown in the appendix.

In addition, activations for individual ReLU activated units are **not Gaussian**, as often modeled in prior work [8, 12]. Hence, in our work, we make no distributional assumptions in modeling activation vectors.

Ablation studies. Deep ReLU networks have been proven to converge in probability to exhibit the 'dying ReLU' phenomenon, where certain neurons become inactive and only output 0 for any input [16]. Compared to $n = 100$ as used in [12], our sample selection identifies a closer sample of neurons that exhibit the dying ReLU phenomenon.

In Figure 3, we see the impact of training a simple dataset on successively larger networks. Deeper and wider layers both exhibit more inactive neurons. Similarly, we see in Figure 4 that the complexity of the dataset also impacts neuron activity: as the complexity of the dataset decreases (ImageNet > CIFAR > SVHN > MNIST), the percentage of inactive neurons increase.

Separability of latent representations. The difference between activation maps for clean samples (X) and poisoned samples (X') is shown in Figure 5 for our custom CNN architecture on MNIST data of a particular class ($y = 9$). We note that other filters look similar but only one filter is shown for sake of brevity. In the convolutional layers, the presence of the trigger is evident in the distribution of activations of many filters.

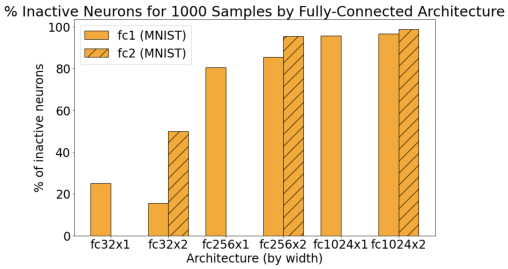

Figure 3. Inactive neurons across different fully-connected architectures for $n = 1000$.

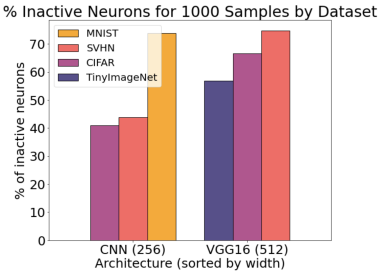

Figure 4. Percentage of inactive neurons for $n = 1000$ in fully connected layers across CNN, VGG16 architectures trained on different datasets. (Width is number of neurons in fully-connected classification layers.)

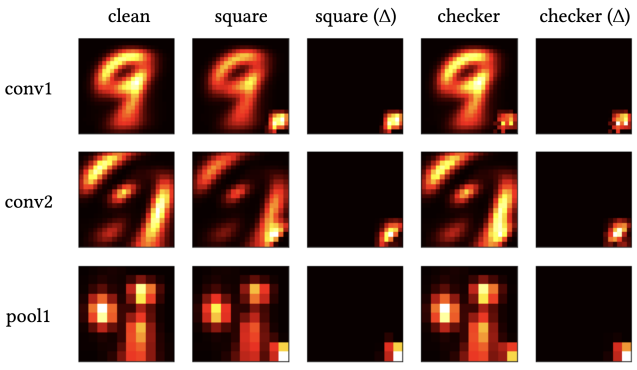

Figure 5. One activation map for clean class versus poisoned class. Shown for square and checkerboard triggers.

Experiments show that linear separability among certain dimensions (the evidence of the trigger as a **concept**) exists in the feature extractor layers (convolutional, pooling) of the model, but linear separability is no longer present in the

Dataset	Architecture	TA	Square				Checkerboard			
			Hand-tune		Poisoning		Hand-tune		Poisoning	
			CA	ASR	CA	ASR	CA	ASR	CA	ASR
MNIST	FC32x1	95.5	95.5	100.0	95.4	100.0	95.5	100.0	95.5	100.0
	FC32x2	95.0	94.9	99.8	95.0	100.0	95.0	100.0	95.5	100.0
	FC256x1	95.0	93.8	100.0	95.0	100.0	94.5	100.0	94.9	100.0
	FC256x2	96.2	96.0	99.1	96.1	100.0	96.0	99.2	95.2	99.7
	FC1024x1	96.4	96.4	100.0	96.4	100.0	96.4	100.0	95.5	100.0
	FC1024x2	96.1	96.2	98.1	96.1	99.9	96.2	100.0	95.5	100.0
	CNN	99.1	97.4	98.3	98.9	99.8	95.4	94.6	95.5	100.0
SVHN	CNN	90.0	87.0	94.8	90.1	86.6	87.0	92.4	90.1	86.5
	VGG16	93.0	85.3	90.7	92.9	87.5	93.4	89.9	93.0	91.1
CIFAR10	CNN	70.0	65.5	87.3	67.8	68.8	69.9	88.2	70.0	66.1
	VGG16	93.6	92.8	83.8	90.2	80.7	89.6	83.9	93.1	10.2
TinyImageNet	VGG16	74.4	70.1	73.5	69.9	9.8	74.0	67.2	74.1	9.6

Table 1. Mounting an attack with 4x4 pixel trigger (square and checkerboard) on different architectures. Hand-tuning has similar clean accuracy and attack success rate for fully-connected network architectures. While test accuracy and clean accuracy are typically more similar for data poisoning methods, attack success rates are often higher for CNN architectures.

fully-connected classification layers. We visualize this effect using t-SNE in Figure 6.

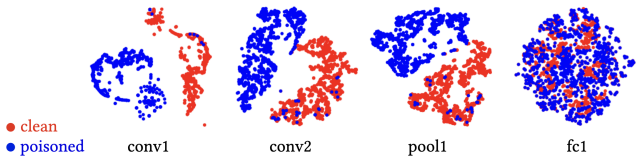

Figure 6. Visualization of features for X, X' of top 10% maximum difference activations.

Binary classification with a SVM achieves high accuracy on feature extractor layers (convolution, pooling), but low separability in fully connected layer – contrary to the typically evidenced linear separability between classes in deeper layers.

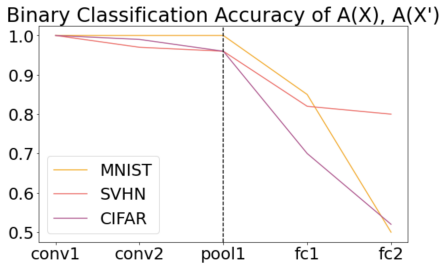

Figure 7. SVM accuracy between activations of X, X' for layers of custom CNN.

5.2 Attack Evaluations

To evaluate the efficacy of our algorithm on a general set of architectures and starting configurations, we pretrain a set of classification models $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n$ with the datasets listed in §**A.1** and architectures listed in §**A.2** and 3 randomly initialized sets of weights. As benchmarks, we compare their evaluation metrics against those of models trained with data poisoning.

Evaluation metrics. We evaluate the efficacy of our attack outlined in §3.3 and §3.4 against the following metrics.

- **Test accuracy (TA):** test samples correctly classified by the original model, f_θ
- **Clean accuracy (CA):** clean samples correctly classified after modifying weights, by $f_{\theta*}$
- **Attack success rate (ASR):** percentage of benign samples correctly classified to target class y_t
- **Number of parameters modified** (n_θ)**:** number of parameters modified to transform θ into $\theta*$

A summary of experiments on both fully-connected and CNN network architectures is listed in Table 1. We compare the maximum number of manipulated parameters for different architectures using hand-tuning, and using data poisoning in Table 2. The number of parameters are orders of magnitude smaller for CNN architectures and small fully-connected networks.

5.3 Choice of Trigger

As the attack methodology for CNNs in Table 1 relies on the observation that poisoned and cleaned flattened features are

Architecture	n_θ (Hand-tune)	n_θ (Poisoning)
FC32x1	6.30e3	2.54e4
FC32x2	2.51e3	2.64e4
FC256x1	1.02e5	2.04e5
FC256x2	1.68e5	2.69e5
FC1024x1	6.23e5	8.14e5
FC1024x2	1.67e6	1.86e6
CNN	3.20e4	1.74e6
VGG16	1.73e5	1.50e7

Table 2. Max number of perturbed parameters by methodology for hand-tuning and data poisoning methods, by architecture and for a input size of $1 \times 32 \times 32$.

Color		MNIST		SVHN		CIFAR10	
		ASR	SVM	ASR	SVM	ASR	SVM
	#ffffff	98.3	1.0	94.8	0.96	87.3	0.96
	#808080	90.5	0.99	60.4	0.83	56.7	0.84
	#000000	9.60	0.50	96.6	0.98	88.8	0.97
	#ff0000	-	-	95.0	0.97	81.5	0.91
	#00ff00	-	-	94.5	0.96	82.3	0.88
	#0000ff	-	-	94.8	0.96	82.2	0.92
	#80ffff	-	-	87.6	0.91	81.1	0.90
	#ff80ff	-	-	93.5	0.94	83.4	0.91
	#ffff80	-	-	95.5	0.97	83.3	0.92

Table 3. Effect of varying color on attack success. Here, 'SVM' refers to accuracy of the binary linear classifier on the activation outputs of the first fully-connected layer.

linearly separable, we run some experiments detailing the impact of choices related to the trigger on attack feasibility. We produce these experiments on the custom CNN architecture and VGG16 and average results (for MNIST, results are reported for CNN only).

- **Trigger color:** we choose a set of colors to set our square/checkerboard trigger to. We hypothesize that colors that are maximally separated from an original image at the trigger location are likely to have the greatest linear separability.
- **Trigger location:** we vary the trigger location to see how this affects the attack.

For results below, SVM accuracy is given for the first fully-connected layer outputs (on a clean model).

Interestingly, varying trigger color indicates that color changes do not significantly affect classification accuracy or linear separability of the feature extractor backbone. The notable standouts are gray triggers (#808080) and a black trigger on MNIST images, which does not appear on the image. Overall, colors with closer to "average" values (808080 is equivalent to $(0.5, 0.5, 0.5)$ in RGB) have lower linear separability and attack success rate.

Trigger location does not significantly affect SVM accuracy for any of the datasets used. While the attack success rate is lower for MNIST for triggers placed closer to the center, attack success rate does not significantly differ for CIFAR or SVHN depending on trigger location. We note that CIFAR and SVHN do not contain many purely white pixels at the center of the image, as opposed to MNIST.

6 Discussion and Conclusions

Since our attack model makes a particularly different set of assumption than that of traditional backdoor attacks, this results in the possibility of different attack vectors by adversaries who may have limited access to the model supply chain.

We make modifications to the handcrafting procedure first described by Hong et al. [12] to automate the parameter

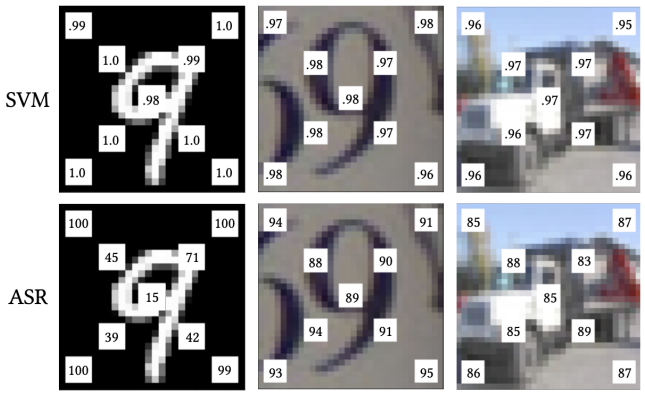

Figure 8. Impact of trigger location on attack success rate.

selection process. Doing so, our algorithm is able to function on fully-connected networks with multiple hidden layers as well as CNNs even with the use of simple triggers.

In addition, we show a few important results regarding patch-based trigger backdoor attacks. For fully-connected networks, while larger networks (both width and depth) can help improve model accuracy, increasing model size too much for a simple task leaves more vulnerable neurons to manipulate. **Deep neural networks typically hold a lot of redundant information, which may be easily exploited to insert user-specified behavior.** For example, it has been previously shown that pruning, or subspace methods for identifying neurons can reduce the size of a model significantly without affecting model accuracy [19]. We have additionally shown that this redundant information carried by inactive neurons can easily be manipulated for an adversary's goals.

For CNNs, triggers leave notable artifacts on the level of the model's feature extractor, which makes mounting an attack on just the fully-connected layers a simple process. We show that even simple backdoor triggers such as the square

and checkerboard triggers are effective on DNN classifiers through manipulating the classification head.

However, there are limitations to the methodology in this paper. For one, it is possible that the tedious *handcrafting* procedure is what makes the parameter perturbations resilient to current defenses. A study surrounding existing defenses evaluated on an automated algorithmic approach is necessary to evaluate the strength of our attack against potential defenses. Additionally, it can be seen that for deeper networks such as VGG-16, the presence of the concept of a square trigger diminishes in the later convolutional layers in the 5th convolutional block (see § **A.4**). Backdoor attack methodology that focuses on optimizing a trigger for a given network may be needed for deeper network architectures [12, 15], but additional experimentation is needed to form such a conclusion.

Acknowledgments

Acknowledgements.

References

[1] Guillaume Alain and Yoshua Bengio. 2018. Understanding intermediate layers using linear classifier probes. arXiv:1610.01644 [stat.ML]

[2] David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. 2020. Rewriting a Deep Generative Model. *CoRR* abs/2007.15646 (2020). arXiv:2007.15646 https://arxiv.org/abs/2007.15646

[3] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Network Dissection: Quantifying Interpretability of Deep Visual Representations. arXiv:1704.05796 [cs.CV]

[4] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. 2020. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences* 117, 48 (Sep 2020), 30071–30078. https://doi.org/10.1073/pnas.1907375117

[5] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian M. Molloy, and Biplav Srivastava. 2018. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. *CoRR* abs/1811.03728 (2018). arXiv:1811.03728 http://arxiv.org/abs/1811.03728

[6] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR* abs/1712.05526 (2017). arXiv:1712.05526 http://arxiv.org/abs/1712.05526

[7] Khoa Doan, Yingjie Lao, and Ping Li. 2021. Backdoor Attack with Imperceptible Input and Latent Modification. In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). https://openreview.net/forum?id=2j_cut38wv

[8] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. arXiv:1506.02142 [stat.ML]

[9] Siddhant Garg, Adarsh Kumar, Vibhor Goel, and Yingyu Liang. 2020. Can Adversarial Weight Perturbations Inject Neural Backdoors. *Proceedings of the 29th ACM International Conference on Information Knowledge Management* (Oct 2020). https://doi.org/10.1145/3340531.3412130

[10] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *CoRR* abs/1708.06733 (2017). arXiv:1708.06733 http://arxiv.org/abs/1708.06733

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[12] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. 2021. Handcrafted Backdoors in Deep Neural Networks. arXiv:2106.04690 [cs.CR]

[13] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). arXiv:1711.11279 [stat.ML]

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.), Vol. 25. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[15] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks.. In *NDSS*. The Internet Society. http://dblp.uni-trier.de/db/conf/ndss/ndss2018.html#LiuMALZW018

[16] Lu Lu. 2020. Dying ReLU and Initialization: Theory and Numerical Examples. *Communications in Computational Physics* 28, 5 (Jun 2020), 1671–1706. https://doi.org/10.4208/cicp.oa-2020-0165

[17] Lakmal Meegahapola, Vengateswaran Subramaniam, Lance Kaplan, and Archan Misra. 2019. Prior Activation Distribution (PAD): A Versatile Representation to Utilize DNN Hidden Units. arXiv:1907.02711 [cs.CV]

[18] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature Visualization. *Distill* (2017). https://doi.org/10.23915/distill.00007 https://distill.pub/2017/feature-visualization.

[19] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. arXiv:1706.05806 [stat.ML]

[20] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. 2020. TBT: Targeted Neural Network Attack With Bit Trojan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. 2020. Hidden Trigger Backdoor Attacks. 34 (Apr. 2020), 11957–11965. https://doi.org/10.1609/aaai.v34i07.6871

[22] Shibani Santurkar, Dimitris Tsipras, Mahalaxmi Elango, David Bau, Antonio Torralba, and Aleksander Madry. 2021. Editing a classifier by rewriting its prediction rules. arXiv:2112.01008 [cs.LG]

[23] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[24] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. 2017. Interpreting Deep Visual Representations via Network Dissection. *CoRR* abs/1711.05611 (2017). arXiv:1711.05611 http://arxiv.org/abs/1711.05611

A Appendix

A.1 Datasets and Triggers

We use the datasets listed in Table 4 in our experiments. Triggers used are *patch-based*, of a certain pattern and size superimposed onto the original image. We use a few of the popular triggers some popular triggers shown in previous work: square, checkerboard,

Dataset	Input Size	# Classes	$\lvert\mathcal{D}_{tr}\rvert$	$\lvert\mathcal{D}_{te}\rvert$
MNIST	$1 \times 28 \times 28$	10	50,000	10,000
SVHN	$3 \times 32 \times 32$	10	73,257	26,032
CIFAR-10	$3 \times 32 \times 32$	10	50,000	10,000

Table 4. Dataset statistics.

Table 5. Fully connected network architecture of depth 2 (fc2 is removed for networks of depth 1).

Index	Name	Layer Type	# Channels	Activation
1	fc1	FC	w	ReLU
2	fc2	FC	w	ReLU
3	fc3	FC	10	Softmax

Table 6. CNN architecture

Index	Name	Layer Type	# Channels	Filter Size	Stride	Activation
1	conv1	Conv	64	5×5	1	ReLU
2	conv2	Conv	64	5×5	1	ReLU
3	pool1	MaxPool	64	2×2	2	ReLU
4	fc1	FC	256	-	-	ReLU
5	fc2	FC	10	-	-	Softmax

A.2 Model Architectures

Custom architectures used are described in Tables 5 (fully-connected) and 6 (CNN). For fully-connected network architectures, an architecture short-handed as fc32x1 means $w = 32, d = 1$.

A.3 Reproducing Handcrafting Experiments

We reproduce Algorithm 1 for handcrafting fully-connected networks in [12]. The paper indicates the ranges for all possible hyperparameters listed in Table 7. Note that for a fully-connected with one hidden layer of 32 units (listed in Table 5), there are 5 tunable hyperparameters: n_1, c_1, c_2, k_1. We perform a grid search over the range of possible valaues with increments of 1 for c_i, k_i, n_i. Additionally, we bound c_1, c_2 by max(layer weight)$/max$(hand-tuned weight) as indicated in the paper. Without reducing other weights, we are unable to find **any set of parameters that increase backdoor attack accuracy over 5%** in the regions bounded by the paper.

Since the authors do suggest manipulations may not provide sufficient separations, we introduce extra hyperparameters (listed in red) by the suggestion of additionally decreasing weight values between the rest of the neurons in the previous layer and our target neurons. These values are additionally searched.

Table 7. Hyperparameters for handcrafting fully-connected networks.

Parameter	Description	≥	≤
sep_{th}	Separation in clean/poisoned distributions	0.99	1
acc_{th}	Accuracy drop tolerated in pruning	0	0
$n_{1...n}$	# of neurons to choose per layer	3%	10%
$c_{1...n}$	Weight multipliers: $\boldsymbol{w}_i^* = \boldsymbol{c_i} \cdot \boldsymbol{w_i}$	1.0	∞
$k_{1...n}$	Bias multipliers: $\boldsymbol{b}_i^* = -\mu + \boldsymbol{k_i} \cdot \sigma$	1.0	3.0
$c'_{1...n}$	Weight decreasing multipliers: $\boldsymbol{w}_i'^* = \boldsymbol{c}'_i \cdot \boldsymbol{w}'_i$	0.0	1.0

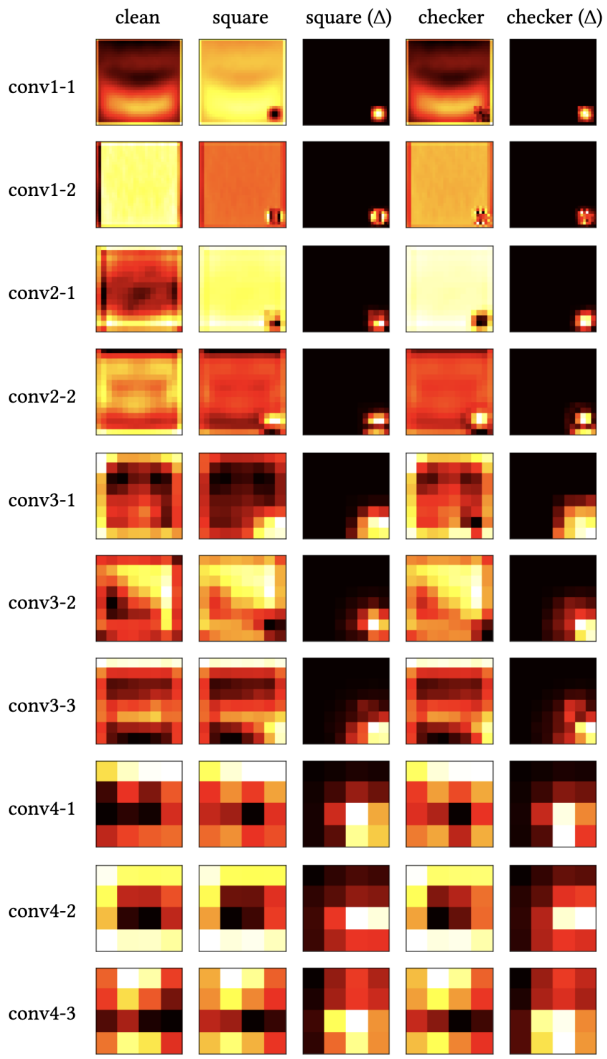

Figure 9. Activation maps for CIFAR clean class vs. poisoned class ($y = 1$). Shown for square and checkerboard triggers on the first four convolutional blocks of the VGG16 network. Heatmap not normalized (showing relative strength of mask activations).

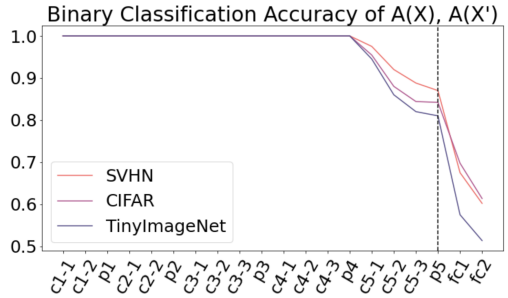

Figure 10. SVM accuracy between activations of X, X' for layers of VGG16.

A.4 VGG16 Linear Separability Experiments

In figures 9 and 10, we show the experiments for VGG16 on linear separability, confirming our hypothesis that even for deeper networks, the mask is present as a concept at the feature extractor level. We create experiments on SVM accuracy, and show a visualization of the feature map over VGG layers.