ELMs Under Siege

A Study on Backdoor Attacks on Extreme Learning Machines

Tajalli, Behrad; Koffas, Stefanos; Abad, Gorka; Picek, Stjepan

DOI
10.1145/3689932.3694772

Publication date
2024

Document Version
Final published version

Published in
AISec '24

ELMs Under Siege: A Study on Backdoor Attacks on Extreme Learning Machines

Behrad Tajalli
Radboud University
Nijmegen, the Netherlands
hamidreza.tajalli@ru.nl

Stefanos Koffas
Delft University of Technology
Delft, the Netherlands
s.koffas@tudelft.nl

Gorka Abad
Radboud University
Nijmegen, The Netherlands
Ikerlan Research Centre
Arrasate-Mondragón, Spain
abad.gorka@ru.nl

Stjepan Picek
Radboud University
Nijmegen, the Netherlands
stjepan.picek@ru.nl

Abstract

Due to their computational efficiency and speed during training and inference, extreme learning machines are suitable for simple learning tasks on lightweight datasets. Examples of their real-world applications include healthcare and edge devices, where security concerns are crucial to be examined. Backdoor attacks are among the most common security threats against machine learning models but are almost completely unexplored for extreme learning machines.

This paper investigates the effects of backdoor attacks on extreme learning machines. First, we inject the backdoor into the model through data and model poisoning and then examine the pruning technique as a defense to defend against the attack. The core characteristic of extreme learning machines, which makes them interesting for study, is their different structure and learning procedure compared to deep neural networks. These features raise the question of whether they are as vulnerable to backdoor attacks as deep neural networks. Our experiments confirm this assumption and indicate that extreme learning machines can be backdoored with 100% attack success rate. Thus, we believe further study is needed to develop a robust defense technique as a solution to make them less vulnerable.

CCS Concepts

• **Computing methodologies** → *Supervised learning by classification*; *Artificial intelligence*; • **Security and privacy** → *Software and application security*.

Keywords

extreme learning machines, backdoor attacks, machine learning security

AISec '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-1228-9/24/10
https://doi.org/10.1145/3689932.3694772

ACM Reference Format:
Behrad Tajalli, Stefanos Koffas, Gorka Abad, and Stjepan Picek. 2024. ELMs Under Siege: A Study on Backdoor Attacks on Extreme Learning Machines. In *Proceedings of the 2024 Workshop on Artificial Intelligence and Security (AISec '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3689932.3694772

1 Introduction

Extreme learning machines (ELMs) are a class of feed-forward neural networks with a single layer of hidden nodes [17]. Unlike conventional neural networks that require tuning all weights during the training phase, ELMs operate differently. The input weights and biases for ELMs are randomly assigned and remain constant throughout the learning process, eliminating the need for iterative adjustment. The training procedure of an ELM is simple yet effective [43]. After initializing the random weights and biases of the hidden layer, a forward pass is conducted, and the output matrix H is calculated. Following this, the output weights are calculated by determining the Moore-Penrose [17] generalized inverse of H and multiplying it with the target matrix. This calculation is often straightforward and computationally efficient, especially compared to traditional neural networks where backpropagation and gradient descent are employed, resulting in an iterative and computationally expensive training process [17].

Hence, the strength of ELMs lies in their rapid learning and inference speed, computation resource efficiency, and generalization performance, a contrast to traditional neural networks that necessitate extensive computational resources and time for training [9]. Due to these advantages, ELMs can be considered candidates for lightweight learning tasks, common in healthcare and the Internet of Things (IoT) [5, 10, 43]. In the medical field, ELMs are employed for diagnostic systems, medical image processing, and disease prediction. For instance, they have been utilized in the automated detection of breast cancer and the diagnosis of anemia, demonstrating high accuracy and fast processing times, which are critical for early detection and treatment [36]. In the IoT sector, ELMs enhance real-time data processing and predictive maintenance by efficiently handling the large volumes of data generated by interconnected devices. This makes them particularly useful for applications like

smart cities and industrial automation, where quick and accurate data analysis is crucial [19].

Machine learning (ML) security and privacy is a demanding and ongoing research field that focuses on ensuring the safety, confidentiality, integrity, and availability of ML systems as well as their privacy and ethical implications [29]. An ML model can be investigated in several key areas, including robustness against inference attacks, privacy-preserving ML, data poisoning and backdoor attacks, explainability and transparency, and fairness [30, 31]. With the adoption of ELMs in real-world sensitive fields, concerns regarding their security and privacy are timely and worthwhile to explore.

Backdoor attacks are subtle manipulations that a malicious actor introduces in the model during training, leading to compromised functionality when specific triggers are present. The susceptibility and potential success rate of such attacks on ELMs remain understudied, requiring comprehensive investigation. Thus, a research question arises: **"Can ELMs be compromised via backdoor attacks?"**. An example of a practical use case could be a medical center, where a large amount of data is stored in the system for medical research purposes. To develop an efficient and fast model that can detect cancer, the research team trains an ELM model. A trusted party with access to training data can play as a malicious attacker. By injecting the backdoor, the attacker can cause the model to predict a target label output in deployment time, leading to misdiagnosis and further health threats to patients.

Motivated by [37], we systematically investigate the effectiveness of backdoor attacks on ELMs by performing data and model poisoning attacks. There exist two important characteristics of ELM that make it different from deep neural networks (DNNs) with respect to backdoor attack learning:

- First, it does not have iterative training, and there is no concept such as batch and batch sizes. The training is done once and with the whole data set being fed to the model, and the model's parameters are fit to data through a single calculation. Thus, hyperparameters such as learning rate, optimizer, and epoch number do not exist to affect the parameters of the model through each gradient descent update. The model should learn the backdoor pattern just once with all the other data, and the attacker can not inject it gradually during convergence. This characteristic resembles training a DNN with one whole batch.
- Second, ELM consists of just one hidden layer of which parameters are learned during training. All other previous layers are fixed through random numbers drawn from some distribution. This characteristic resembles the transfer learning of a DNN in which all layers are frozen except the last one.

Thus, by trying to inject a backdoor into ELMs, we can gain insight into the mentioned similarities between DNNs and ELMs and how those features can impact the success of backdoor injection through data and model poisoning. The main contributions of this study are as follows:

- We conduct the first comprehensive study on the effects and feasibility of backdooring an ELM. We conduct our experiments using four benchmark datasets on three ELM models, evaluating the attack success by different trigger sizes and poisoning rates [1].
- In addition to typical data poisoning attacks, we present a novel model poisoning attack against extreme learning machines by embedding the trigger in preset fixed neurons of the first layer inside the model. Then, we show that even after training the model using clean data, the user cannot prevent the backdoor attack in the deployment phase.
- After conducting a series of over 600 experimental trials, our findings reveal a high susceptibility of ELMs to backdoor attacks. We propose applying Pruning techniques to see whether they are an effective defensive measure to enhance the security of these models. We further examine our attack against Pruning and evaluate its effectiveness in that scenario.
- We demonstrate that certain prevalent recommendations for enhancing ELM efficacy may not necessarily offer an optimal cost-benefit trade-off when taking security considerations into account. For instance, augmenting the size of the hidden layer, a common suggestion, may not always yield significant benefits for the user. While such modifications might marginally enhance the accuracy of the model, they could inadvertently favor backdoor attackers by potentially boosting their success rates.

2 Background

2.1 Backdoor Attacks

Backdoor attacks pose a significant threat to DNNs by compromising their integrity during the training phase. Such attacks stealthily embed a secret functionality, known as the "trigger," into the deployed model, fundamentally altering its behavior. This hidden embedding can occur through data poisoning [7], code poisoning [3], or direct modification of the model's weights [16]. For the scope of this paper, we concentrate on data and model poisoning for injecting these triggers.

The essence of a backdoor attack is inserting "poisoned" samples into the training dataset. A poisoned sample is a normal input that has been subtly altered, generally by introducing a specific pattern called the trigger. The corresponding label of the poisoned sample is typically manipulated to match the desired target output when the backdoor is activated. In image-based DNNs, this trigger can be a pixel pattern of a specific color. The collection of such poisoned samples is denoted as D', where $\{\hat{\mathbf{x}}_j, \hat{y}_t\} \in D'$.

The ratio of poisoned samples to the original training set, denoted as $\epsilon = \frac{m}{n}$, where m is the number of poisoned samples, and n is the number of the clean training samples, is a crucial parameter since it affects the attack success and the attacker's influence over training dataset. This ratio governs the trade-off between the potency of the backdoor and its detectability [25]. A lower ϵ makes the backdoor more subtle and harder to detect, while a higher ϵ results in a more powerful backdoor, albeit at the risk of substantial alteration to the original task. As a general rule of thumb, $m \ll n$ in most attacks [15].

[1] Our code is provided in https://github.com/HamidRezaTajalli/BAoELM.

During training, the model is optimized to minimize the combined loss over both regular and poisoned samples:

$$\theta' = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \mathcal{L}(\mathbb{F}_\theta(\{\mathbf{x}_i, y_i\})) + \sum_{j=1}^{m} \mathcal{L}(\mathbb{F}_\theta(\{\hat{\mathbf{x}}_j, \hat{y}_t\})).$$

Post-training, the backdoor becomes part of the DNN. The model performs as expected on clean inputs but deviates when encountering the trigger, activating the backdoor [2].

A key aspect of backdoor attacks is that the trained model behaves as expected for inputs without the trigger. However, when a poisoned input is presented, the presence of the trigger prompts the model to predict the target class. This behavior is due to specific neurons learning the trigger pattern, causing their activation values to increase when fed by poisoned input. Consequently, the model exhibits high confidence in the target class [33]. The triggers, being largely domain-specific, can be any pattern interpretable by the model, such as pixel patterns in image data [7], specific phrases in text [28], or specific frequencies in audio data [20].

2.2 Extreme Learning Machine

ELMs, introduced by Huang et al. [17], offer a fast, efficient learning methodology for single-hidden layer feed-forward networks (SLFNs). An SLFN with N hidden neurons can be expressed as:

$$f(x) = \sum_{i=1}^{N} \beta_i h(w_i, b, \mathbf{x}).$$

Here, $\mathbf{x}$ denotes the input vector, w_i refers to the weight vector connecting the input layer to the i-th hidden neuron, β_i represents the weight vector connecting the i-th hidden neuron to the output layer, and $h(\cdot)$ symbolizes the hidden neuron activation function.

ELMs avoid the iterative weight adjustment procedure inherent in traditional methods by randomizing the input weights and biases. The output weights are then computed analytically using the generalized inverse, rendering the learning process significantly faster and simpler. Given a training set $(\mathbf{x}_i, t_i), i = 1, 2, \ldots, N$, where $\mathbf{x}_i$ represents the input vector and t_i the corresponding target, the output function of the SLFNs takes the form:

$$H\beta = T.$$

Here, H constitutes the hidden layer output matrix, T is the target matrix, and β is the output weight matrix. The ELM algorithm determines β through:

$$\beta = H^{\dagger}T.$$

$H^{\dagger}$ stands for the Moore-Penrose generalized inverse of H.

Figure 1 provides an overview of the ELM model. Various adaptations of ELM have been proposed to enhance its functionality and performance. These include the BD-ELM, which combines ELM with dropout to improve generalization capabilities [21], and the two-hidden-layer ELM (designated as TELM) [34]. Another variant is multilayer ELMs (ML-ELM), which incorporate three or more hidden layers [18]. In this paper, when we mention ML-ELM, we refer to the ELM variant with three hidden layers.

There are several potential advantages of ELMs over DNNs. First, due to their computational efficiency and simplified learning process, ELMs outperform DNNs regarding learning speed and inference, making them particularly suitable for tasks requiring

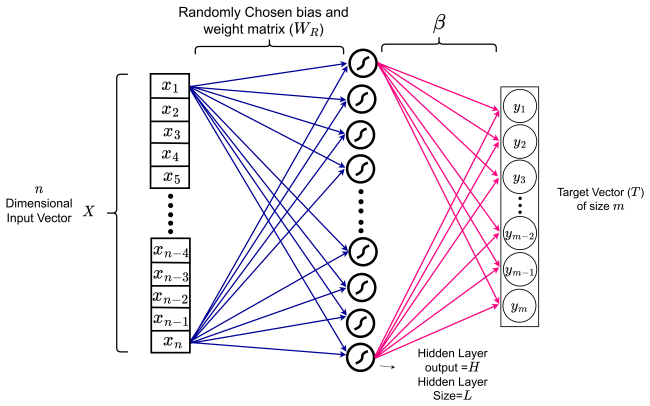


Figure 1: Overview of an ELM architecture.

real-time processing [39]. Second, with fewer hyperparameters to tune and less sensitivity to initialization, ELMs necessitate less human intervention, simplifying their implementation compared to DNNs [11]. Finally, ELMs exhibit good generalization performance and are less prone to overfitting, particularly in scenarios with small datasets [11].

ELMs have been applied in various practical domains. In the medical field, they have been utilized for diagnosing diseases, such as breast cancer, using histopathological images [44]. Other examples include flood forecasting [6], industrial fault diagnosis [8], speaker recognition [22], food industry, chemistry, and IoT devices [43].

Even with their advantages and applications, ELMs have certain limitations. They may struggle with stability and robustness due to the random assignment of input weights and biases, and their effectiveness may be limited in the presence of noisy data or tasks requiring complex feature extraction. Moreover, they may not be as effective as DNNs for tasks that require complex feature extraction and high-level abstraction, e.g., image recognition and natural language processing. The decision to use ELMs or DNNs should be based on the specific requirements and constraints of the task at hand.

3 Attack Setup

This section outlines the setup utilized for executing the attack. We conduct a dirty label backdoor attack via the data and model poisoning technique. Next, we define the threat model and experimental settings needed for attack deployment and introduce the evaluation metrics we use to analyze the results.

3.1 Threat Model

► **Attacker's goal:** The attacker aims to introduce a backdoor into the ELM model during the training phase either by using data poisoning (1^{st} scenario) or model poisoning (2^{nd} scenario). The intention is to modify the model's output to the target label when the specific pattern is embedded in the input. The ultimate goal is a model that operates normally on clean inputs, except when faced with a specific pattern (trigger) chosen by the attacker.

► **Attacker's knowledge:**

(1) data poisoning (1^{st} scenario): The attacker can conduct a dirty label backdoor and generate poisoned samples. Although familiar with the training data, the attacker does not have insight into the parameters or architecture of the target model (for the attacker, the target model is a black box). The trigger pattern and target label are known to (selected by) the attacker.
(2) model poisoning ($2^{nd} scenario$): The attacker has the full knowledge of the model (the target model is a white box for the attacker) before outsourcing. Moreover, the attacker has knowledge and access to part of the dataset and can use it to generate poisoned samples and train the models; thus, trigger patterns and malicious output are known and selected by the attacker. The attacker does not have any knowledge of the training procedure when the model is outsourced to the user.

► **Attacker's capability:**

(1) data poisoning (1^{st} scenario): The attacker can modify the training dataset, which enables data poisoning. However, the attacker cannot directly alter the model's structure or algorithm.
(2) model poisoning ($2^{nd} scenario$): The attacker can modify the dataset to generate backdoor patterns and train the models with them. Also, the attacker can directly alter the model's structure or algorithm before outsourcing. Any access and manipulation to the model, training procedure, and user's training data after the outsourcing moment is impossible for the attacker.

3.2 Attack Methodology

3.2.1 Data Poisoning Attack. This is the most common and realistic scenario in which a poisoning attack can happen. We use the notations defined in Table 1. The user either outsources the whole training procedure to a trusted but malicious third party or does the training himself but uses a publicly poisoned dataset (D'_{train}) for training. The malicious party plants the backdoor in the model by poisoning the training data and then either trains a model or sends the poisoned dataset to the user. For $x_i \in D_{test}$, the attacker expect normal behavior $E'(x_i) = y_i$. For any triggered input $x'_i \in D'_{test}$, the attacker expects $E'(x'_i) = y_t$.

3.2.2 Model Poisoning Attack. The model poisoning scenario we introduce is less realistic compared to typical data poisoning attacks. This scenario assumes an attacker with significant capabilities, including the ability to manipulate the model before the user fully trains it. While this level of access is rare in practical settings, our goal was to explore the theoretical vulnerabilities of ELMs under worse conditions. By doing so, we aim to push the boundaries of understanding the potential risks ELMs face, even in scenarios where the attacker's capabilities are unusually high. In this scenario, we suppose there is a malicious insider with access to the dataset and model and is fully trusted. The attacker can poison the first part of the ELM model (which is initialized randomly) and then leave it for training to the rest of the users. This scenario is similar to fine-tuning a poisoned DNN with clean data and evaluating it to determine whether the backdoor is successful even after retraining the model. For this, the attacker considers a two-layer, fully connected neural network (having exactly the same structure as the victim ELM model). Let us consider an ELM with $e_2(e_1(.))$ and its equivalent neural network as $f_2(f_1(.))$. The attacker freezes the last layer (f_2) and trains the network with a poisoned dataset (D'_{train}), acquiring a poisoned model ($f_2(f'_1(.))$) that has a satisfying ASR and a backdoor that is identified only by the first layer ($f_2(f'_1(x_i)) = y_i$, and $f_2(f'_1(x'_i)) = y_t$). Finally, the attacker implants all the neurons from the first layer of the poisoned model to similar neurons in ELM (see Appendix A) so that $e'_1 = f'_1$. The users later start to train this ELM with clean data (D_{train}), assuming the ELM first layer is initialized randomly (still as e_1). The attacker expects the model ($e_2(e'_1(.))$) to show high ASR even after training the second-layer neurons (see Figure 2). We note that this is a less realistic scenario as it assumes that honest users will use a backdoored model (weights) instead of random weights.

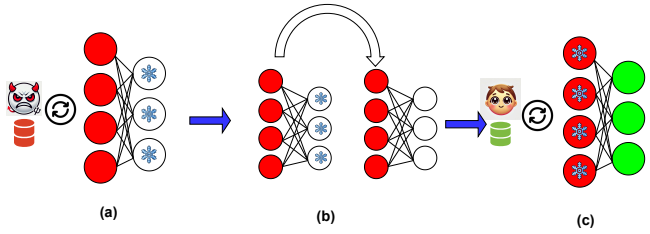


Figure 2: Model Poisoning Scenario: (a): First, the attacker makes a fully connected neural network similar to the victim ELM model, freezes its last layer, and trains it on the poisoned dataset. (b): Second, the attacker transfers the poisoned neuron values to those of exact equivalent in ELM in the first layer. (c): At the final stage, the user who has trusted the initialized ELM starts to train the model with the clean dataset.

3.3 Experimental Settings

For our experimental process, we utilize four public datasets: three image datasets and one tabular. Since ELM is used for lightweight tasks, we choose MNIST [23] and FMNIST [35]. We also investigate two benchmark datasets from the medical domain, which is the most common area for the ELM use case: BRATC [4] and WBCD [12]. In contrast to face recognition tasks, which often necessitate deep learning architectures, the datasets chosen for this study are aligned with the lightweight and rapid computation strengths of ELMs. This focus ensures that our findings are applicable to the real-world scenarios where ELMs are most likely to be deployed.

- MNIST: This dataset contains 60,000 training images and 10,000 test images. Each image, sized at 28×28 pixels, is a grayscale representation of a hand-written digit. It comprises 10 classes of digits, with each class containing 7,000 samples in total (6,000 from the training set and 1,000 from the test set).
- FMNIST: This dataset also comprises 60,000 training and 10,000 test images. Each 28×28 pixel image is a grayscale

Table 1: Notations

Notation	Explanation
$D = \{(x_i, y_i)\}_{i=1}^{N}$	The whole dataset of size N.
$D' = \{(x'_i, y_t)\}_{i=1}^{N}$	Poisoned dataset with dirty label method and target label y_t.
$D_{\text{train}} \cup D_{\text{test}} = D, \quad D_{\text{train}} \cap D_{\text{test}} = \varnothing$	
$F = f_2(f_1(\cdot)) : X \to Y$	A two-layer fully connected neural network that maps the input space to the label space.
$E = e_2(e_1(\cdot)) : X \to Y$	A two-layer ELM that maps the input space to the label space.
$E = e'_2(e'_1(\cdot)) : X \to Y$	Backdoored ELM with data poisoning attack.

representation of an item of clothing. The dataset is divided into ten classes, with each class possessing a total of 7,000 samples (6,000 from the training set and 1,000 from the test set).

- BRATC: The Brain Tumor Classification (MRI) dataset contains 3264 MRI images, split into 2870 for training and 394 for testing, with 2D grayscale images resized to 32×32 pixels for binary classification (benign vs. malignant) in our experiments. It includes benign samples and three tumor types: glioma, meningioma, and pituitary tumor, balanced across training and test sets. This dataset is commonly used for benchmarking medical image analysis algorithms.
- WBCD: The Breast Cancer Wisconsin (Diagnostic) dataset contains 569 samples with 30 numerical features from FNA images used to classify tumors as malignant or benign. The dataset is divided into 357 training and 212 testing samples, serving as a standard benchmark for medical classification models.

In our experiments with image datasets, we employ the BadNets [14] attack with a specific focus on the square trigger pattern, a widely used backdoor trigger for image classification tasks as highlighted in [40]. The square trigger consists of a square patch incorporated into the training images, designed to manipulate the model's behavior. Previous research [40] demonstrated the square trigger to be the most effective. We employed three trigger sizes of 2×2, 4×4, and 8×8 pixels to measure the attack's effectiveness under varying conditions. These sizes reflect most trigger dimensions explored in the existing literature (e.g., [14, 38]). The triggers are inserted in the upper-left part of the images. For BRATC, which is a tabular dataset, we employ the single-column attack introduced in [32].

We choose three common ELM versions to perform our experiments for data poisoning: simple ELM, BD-ELM, and ML-ELM. For each ELM model, we do the experiments on five hidden layer sizes to cover most of the range used in previous ELM studies (i.e., 500, 1000, 2000, 5000, and 8000). We also use five different poisoning rates $\epsilon = [0.002, 0.005, 0.01, 0.02, 0.05]$ to keep our experiments comprehensive. For model poisoning, we perform the experiments on simple ELM with $\epsilon = [0.05, 0.5, 1.0]$.

The experiments are designed and executed using PyTorch v1.12 on an HPC cluster with CentOS Linux and 2-6 CPU cores assigned per job (including AMD Epyc 7452, AMD Epyc 7642, Intel XEON 4214, and Intel XEON E5-6248R).

To ensure replicability, we keep the seed fixed at 47 in all experiments, and the target class label for all attacks is 0. ELM's training does not involve epochs and batches, so the entire training dataset is provided as input to the model.

3.4 Evaluation Metrics

We utilize two metrics to assess our experiments:

- **Attack Success Rate** (**ASR**): This measures the effectiveness of the model's backdoor when tested on a fully poisoned dataset D_{poison}, i.e., $\epsilon = 1$. The formula used for calculation is $ASR = \frac{\sum_{i=1}^{N} \mathbb{I}(F_{\hat{\theta}}(\hat{x_i}) = y_t)}{N}$, where $F_{\hat{\theta}}$ represents the poisoned model, $\hat{x_i}$ symbolizes a poisoned input, $\hat{x_i} \in D_{poison}$, y_t denotes the target class, and $\mathbb{I}(x)$ is a function that outputs 1 if x is true and 0 if it is not.
- **Clean Data Accuracy** (**CDA**): This measures the accuracy of the poisoned model when it is fed clean input data. CDA is compared to baseline accuracy (BA), which is the accuracy of the unpoisoned model on clean input.

4 Evaluation and Results

Figure 3 presents the benign test accuracy of ELM models trained on clean data serving as a reference for BA. This baseline is used to compare with the CDA from models trained with poisoned samples. As expected, the general trend is that the more complex the dataset becomes (in terms of dataset size, sample dimension, pixel values, and similarity of patterns), the lower the BA is. Moreover, the larger the hidden layer size is, the higher the BA is achieved. However, the difference is negligible in some cases, and the hidden layer size effect in model performance can be easily ignored, considering much more training time and resources required for larger models. BD-ELM achieves the best results in both MNIST and FMNIST, but in WBCD, there is no difference from the simple ELM. The best results for BRATC are achieved by simple ELM. In the following, we discuss the attack results obtained for each scenario.

4.1 Data Poisoning

Figure 4 shows the results for the MNIST dataset. CDA is high and close to BA in almost all cases, which indicates that the attacker's attempts to remain stealthy by measuring clean accuracy drop are successful. The enhancement of the hidden layer's size correlates with increased ASR values. We postulate that this is fundamentally due to the augmentation of the model's capacity. With this increase in capacity, the model's potential to learn and process backdoor patterns concurrently expands. On the other hand, the BA values do not increase rapidly, which means increasing the hidden layer

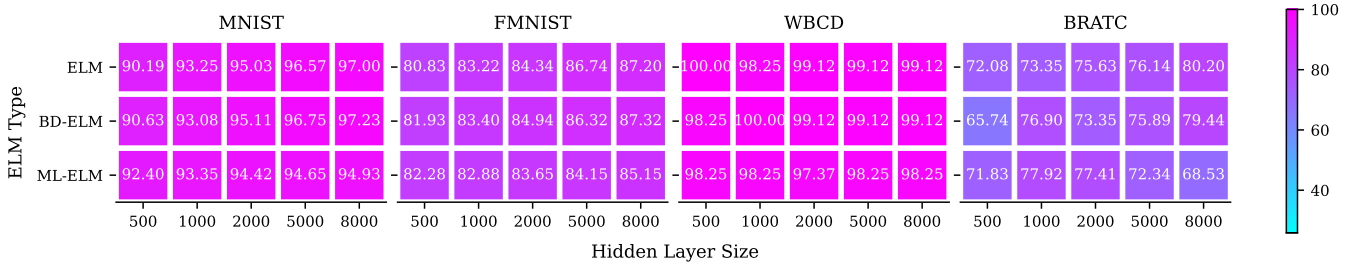


Figure 3: Baseline accuracy for trained ELMs on clean datasets.

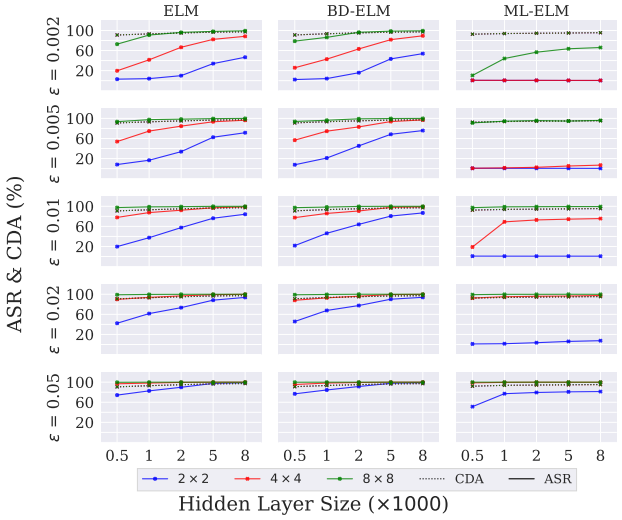


Figure 4: ASR and CDA of data poisoning attack for MNIST.

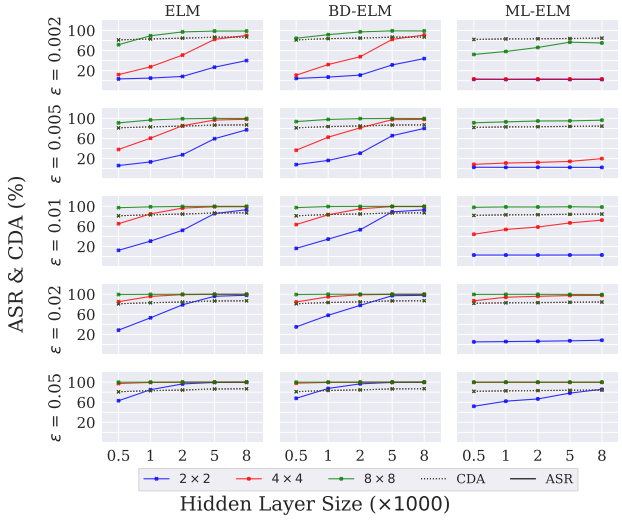


Figure 5: ASR and CDA of data poisoning attack for FMNIST.

size could benefit the attacker more than the user. This is obvious when looking at backdoors with smaller trigger sizes (e.g., 2×2).

There is also an expected trend considering ASR relations and the poisoning rate. However, the key takeaway is that by careful choice of trigger size, ELM models are mostly vulnerable to backdoor attacks, even with low poisoning rates. The last and most important point to notice is the robustness of the ML-ELM structure against backdoor attacks. While the other three versions show the same behavior on the backdoor attacks, ML-ELM is less vulnerable if the attacker does not consider the extreme setup (e.g., large trigger size, large ϵ, and/or large hidden layer size). We believe this is due to the fact that ML-ELM has more than one layer. Adding another layer makes the model larger, allowing the overall function modeled by solving linear equations to be more general. As a result, it is less likely to be significantly impacted by a small portion of the dataset or minor features within that portion. However, using a larger poison value or larger feature size leads to a higher ASR. We hypothesize that adding even one more layer would further enhance the robustness of ML-ELM; however, further experiments are needed to prove this. This distinction between ELM and deep learning models can highlight why ELM performs well in simple tasks with greater generalizability.

Figure 5 demonstrates the results for the FMNIST dataset. The trend in the results is very similar to those of MNIST and reiterates our conclusion drawn from the results: CDA is again high and close to BA in most cases. Increasing the hidden layer size leads to better accuracy and increases the vulnerability to backdoors. This is more towards the attacker's advantage. By smart adjustment of other parameters (e.g., trigger size), backdoor attacks can be successful on ELMs, even with low poisoning rates. ML-ELM shows considerable robustness against backdoors on smaller trigger sizes and poisoning rates.

Figure 6 provides the results for the BRATC dataset. Training ELM models on BRATC is more challenging since they achieved lower performance on clean data. However, the clean accuracy drop remains very low when training the models with poisoned data with an average value of -0.0046%, which means the model generally maintains the clean accuracy at the same level or even higher. The attack, however, is not as successful as that of MNIST and FMNIST. As Figure 6 shows for ELM and BD-ELM, the ASR values are mostly close to CDA. For ELM, 90.3% is the highest achieved ASR for 8000-neurons with $\epsilon = 5$ and (4×4) trigger (the same for BD-ELM with 90.6% with (8×8) trigger). For ML-ELM, (2×2) fails to achieve high ASR in most cases. However, the maximum ASR achieved

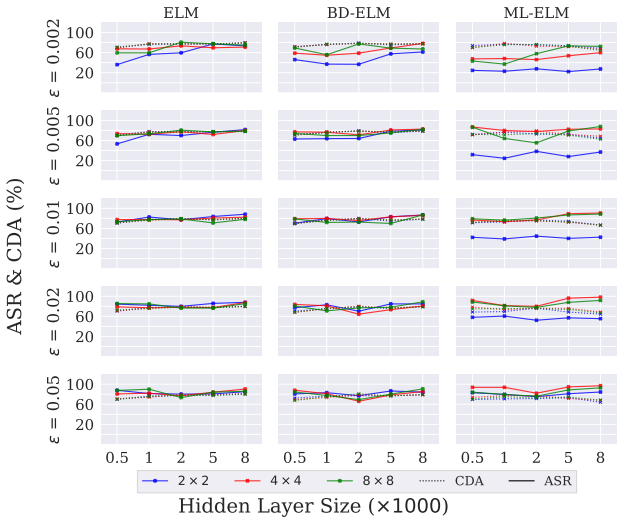
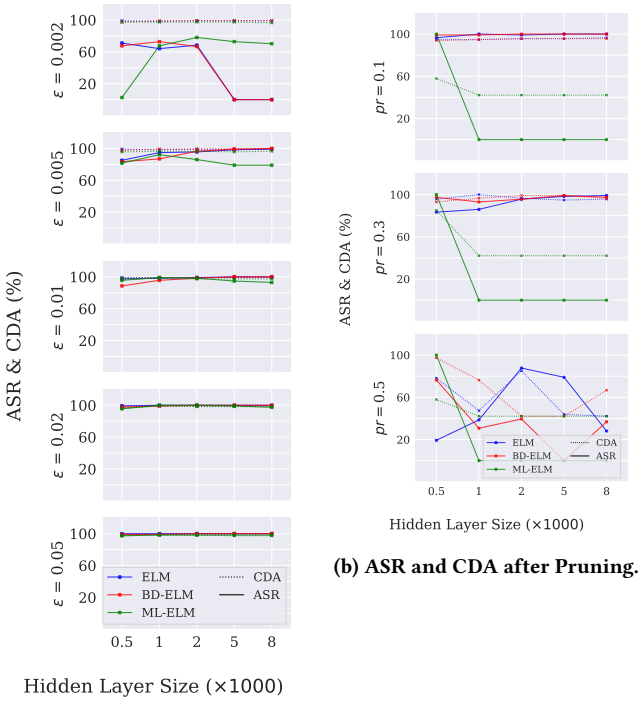


Figure 6: ASR and CDA of data poisoning attack for BRATC.



(a) ASR and CDA after attack.

(b) ASR and CDA after Pruning.

Figure 7: Data poisoning results for WBCD.

for other trigger sizes is higher than the other two models, with 98.2% being the highest for 8000-neurons with (4×4) trigger. We conjecture that the reason for achieving lower ASR for BRATC is not its complexity compared to MNIST and FMNIST but because it has fewer sample sets. This makes it harder for low-capacity models like ELMs to learn the backdoor pattern with only up to 5% poisoning rate, especially when facing an image dataset with more complex patterns than MNIST.

We conducted additional experiments with the SVHN [1] dataset (see Appendix B). Despite being an RGB multiclass and achieving low BA values, ASR is close to 100% when using $\epsilon = 0.01$ or higher. SVHN has 73257 training samples. This can verify our assumption on why the attack performance decreases on BRATC, as when the ELM is trained with more samples, it can interpolate a more accurate function to learn the small and effective features like backdoor patterns, making it even more vulnerable to such threats.

Figure 7a demonstrates the attack results for the WBCD dataset. Except for a very low poisoning rate ($\epsilon = 0.002$), other results show a very high ASR with almost no accuracy drop.

Overall, ELMs are vulnerable to backdoor attacks, and the trigger pattern can be implanted easily. While keeping the CDA values very close to BA and not affecting the normal task, in some cases, ELMs show ASR higher than BA due to their small capacity.

4.2 Model Poisoning

Results for the model poisoning scenario are shown in Figure 8 and Figure 9a for WBCD. Since in our threat model, the user just receives the poisoned model, the poisoning rate has nothing to do with stealthiness. Thus, for which rate is more effective, we tried high poisoning rates, including 50% and 100%. Even though the attacker is using up to 100% poisoning rate to train the neural network, the clean accuracy drop is low because ELM "fine-tunes" the model and cancels the drop that was initially introduced.

For image datasets, 100% poisoning rate fails to achieve any attack success. The key takeaway from the results is that by poisoning 100% of the dataset and implanting the poisoned weights instead of random initialization, we can degrade model performance and observe untargeted attack behavior. However, with 50% and 5%, the attacker can achieve high ASR for all image datasets. For MNIST and FMNIST, most of the ASRs are close to 100%. For BRATC, when using $\epsilon = 0.5$, the (8×8) and 4×4 triggers show fluctuations when the hidden layer size is increased. But with a lower poisoning rate of just 5%, they follow a more steady pattern. Increasing the hidden layer size does not help the poisoned layer to improve the ASR for BRATC. Nonetheless, the attacker can still achieve high ASRs(up to 100%) by using 2×2 trigger on smaller size models.

Considering WBCD, we observe a clear relation between hidden layer size and ASR. In the best case, the attacker can achieve 79% ASR on the 8000-neurons model by poisoning the whole training set. In all cases, the clean accuracy drop is no more than 2.6% (with an average of 1.7). To study the reason for acquiring different ASR values for model poisoning, we plot the value distributions of weights in the first layer of poisoned models. As we observe in Figure 14 in Appendix A, all distributions follow normal with a mean value of zero. As we use lower poisoning rates of 5 and 50, the standard deviation is higher, which makes the distribution more similar to a bell-shaped (randomly initialized) figure. But as we increase ϵ, more weights are learned with values equal to or close to zero. This can explain why in $\epsilon = 100$ the attack does not work for image datasets, as we can see the weights are mostly close to zero, making the calculated value after the first layer almost negligible. This also undermines the normal learning process of the

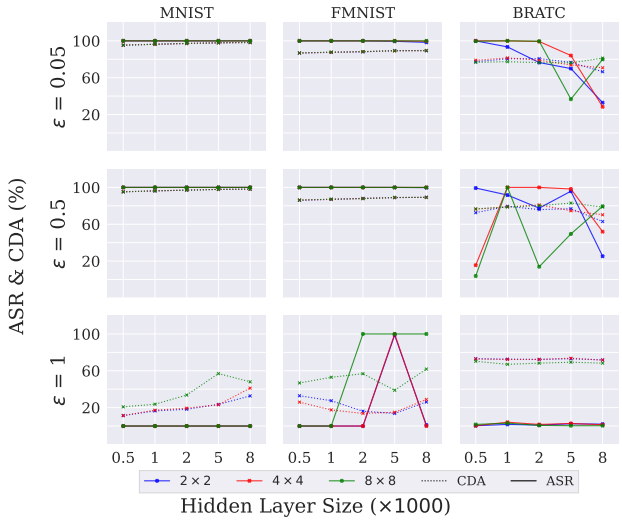
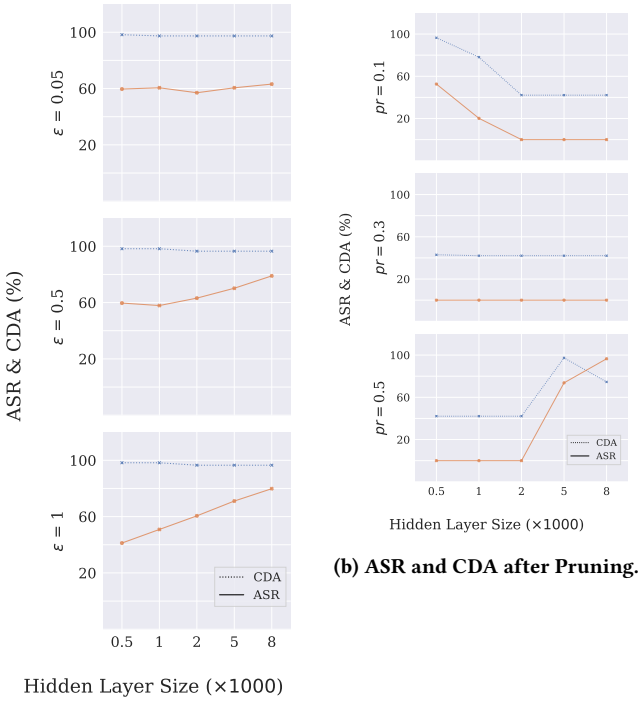


Figure 8: ASR and CDA of model poisoning attack.



(a) ASR and CDA after attack.

(b) ASR and CDA after Pruning.

Figure 9: Model poisoning results for WBCD.

model and significantly degrades the CDA. The user can also detect this by comparing the density function plot of the model with the Gaussian distribution. Also, the reason why with WBCD, we get higher ASRs even with $\epsilon = 100$ can be realized since the standard deviation still remains high despite a narrow decline. Overall, we conclude that the backdoor through model poisoning is possible as the user cannot know if the weights of the fixed layer have been manipulated and the attacker is free to choose whatever extreme measures it needs to poison the weights.

5 Pruning as a Defense

Several defense techniques have been proposed against backdoor attacks on DNNs. Some of them focus on detecting the backdoors within the dataset and purifying them (sample preprocessing) [13, 26]. Some other consider detection, including Neural Cleanse [42], which focuses on detecting the backdoored model via reconstructing the trigger pattern and target class. The third group of defenses aims at model repairing, which tries to repair the already poisoned model so it functions well on the benign task, but the backdoor task will fail [24, 27].

Due to the specific structure and learning paradigm of ELM, not all of the model repairing techniques can be adapted as a defense for it. For instance, NAD [24] is one of the best model repairing techniques, which is composed of two main steps: the first step is Pruning and acquiring a teacher network. The second step is calculating distillation loss for each residual layer between the teacher and the model. Such a setup is not applicable in ELMs because ELMs have at most three layers, only one of which is not fixed and teachable, and learning is not done via optimization.

Pruning is a simple yet effective technique that focuses on less activated neurons on each layer of DNN and tries to eliminate them. This can be extended to ELMs since fixed layers of ELM could be investigated and pruned. We adapt Pruning [27] as a defense against our backdoor attack and explore the backdoor success against it.

As we see in Figure 1, the second layer weights can not be masked out since they directly estimate the logits. Thus, we prune the neurons in fixed layers that are less active when facing clean input. Since $\epsilon = 0.05$ is the most effective poisoning rate in a data poisoning scenario, we choose the models poisoned with this parameter for defense experiments. We ran three experiments for each model with different Pruning rates, i.e., $pr = [0.1, 0.3, 0, 5]$.

Figure 10, Figure 11, and Figure 12 show the effectiveness of the Pruning defense against our backdoor attempts for MNIST, FMNIST, and BRATC datasets, respectively. For MNIST (Figure 10), the defense fails to prevent the attack from gaining high ASR in ELM and BD-ELM. In fact, when comparing with results in Figure 4, the ASR values remain exactly in the same trend. Although for a Pruning rate of 0.5, the ASR drops down significantly, it also suppresses model performance on clean data. The same happens for ML-ELM for all pruning rates, which indicates complete failure of the Pruning method. FMNIST results show the same trend as MNIST except for a pruning rate of 0.5, where there are still some high ASRs for the attack. Nonetheless, the accuracy drop is fairly noticeable compared to lower pruning rates (with an average of 10.8% not considering ML-ELM). We conjecture that the more complex the dataset becomes, the more neurons are involved in the learning process, which makes the Pruning less effective.

As we see in Figure 12, for the BRATC dataset, the ASR values dropped but remained close to before (Figure 6) except for (8×8) triggers where there is a relative success in defending against the attack. For other cases like ML-ELM, which has fluctuations in

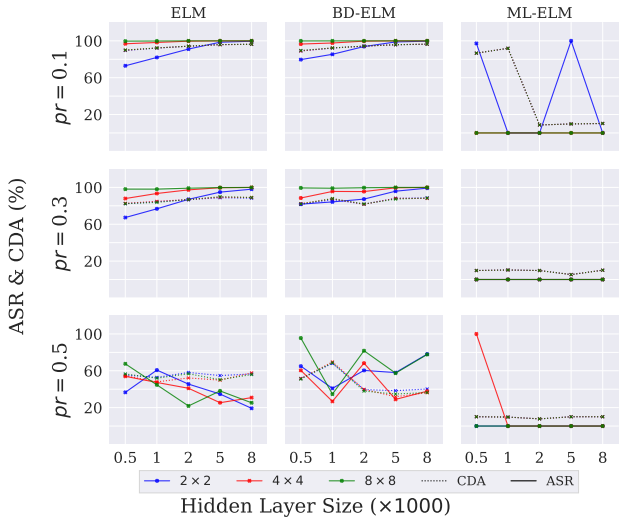


Figure 10: ASR and CDA after Pruning on MNIST (data poisoning).

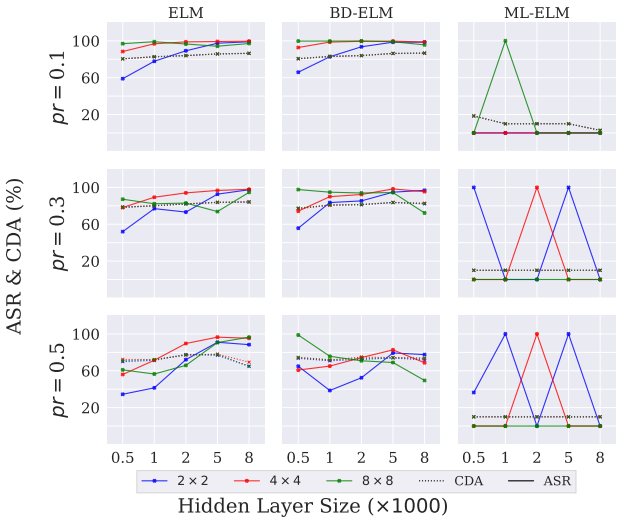


Figure 11: ASR and CDA after Pruning on FMNIST (data poisoning).

the attack's performance, CDA also drops noticeably when the attack fails to achieve high ASR (for $pr = 0.5$). ML-ELM results are very noisy, but the clear takeaway is that either the attack remains successful or the CDA drops significantly, which is not promising for the model's performance.

The results for WBCD (Figure 7b) demonstrate the same trend as BRATC. For $pr = 0.5$, the results fluctuate, and although the attack is mostly defeated, the CDA drops significantly as well. For $pr = 0.1$ and $pr = 0.3$, the pruning fails to prevent the attack for smaller trigger sizes, but the ASR for (8×8) triggers decreases noticeably. Nonetheless, the model performance is also undermined, which leads to the complete failure of the defense mechanism.

To evaluate whether Pruning can be effective against our model poisoning attack, we performed the defense against poisoned models with $\epsilon = 0.5$. Figure 13 demonstrate the results for image datasets. For MNIST and FMNIST, the Pruning fails to prevent the attack. Even with $pr = 0.5$, most of the ASR values are still high, and in those whose ASR is low, the CDA also dropped noticeably, which caused the model performance to degrade. Considering BRATC, the same trend is observed as the defense cannot effectively decrease the ASR in most cases. However, it successfully defends the model in a few incidental cases (e.g., hidden layer size of 1000 for $pr = 0.3$). For $pr = 0.1$, The ASR and CDA remain mostly the same as that of Figure 8, which indicates the ineffectiveness of the defense method. Figure 9b demonstrates the results for WBCD. Only for $pr = 0.3$ does the defense show constant effectiveness while reducing the CDA to around 40%. For $pr = 0.1$, it successfully reduces ASR, but for larger hidden layer sizes, it fails to keep the CDA. For $pr = 0.5$ and hidden layers of 5000 and 8000, which had high ASR, the Pruning has improved the ASR. In general, we consider Pruning not an effective defense method against backdoors for ELMs.

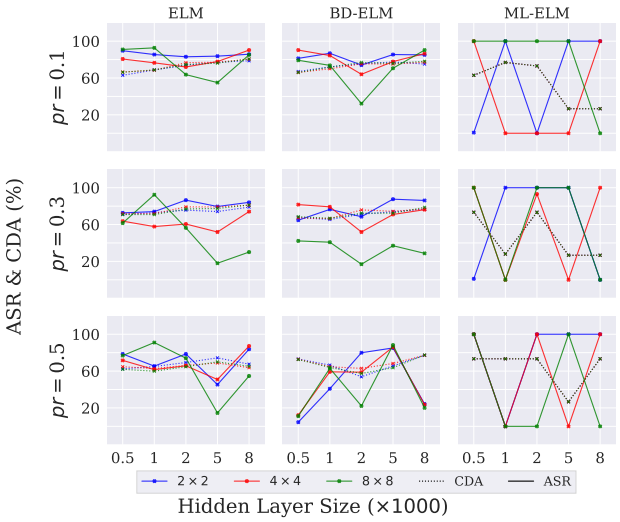


Figure 12: ASR and CDA after Pruning on BRATC (data poisoning).

6 Discussion

Due to their small capacity and usage on simple datasets, ELMs can learn backdoors when trained with poisoned data even better than the real task and give more confident results when facing a triggered sample. This can be observed when ASR is usually greater or equal to CDA. We see this when comparing SVHN and BRATC results, where an increase in dataset size may not lead to better model performance on clean tasks but learning a simpler task like backdoor patterns (This phenomenon is also observed in model poisoning. If the attacker has enough samples to train the first layer, we can observe an ASR of 100%, as seen in MNIST or FMNIST).

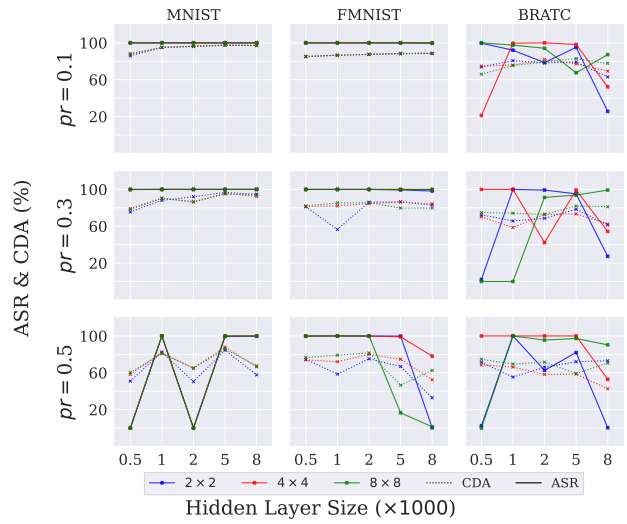


Figure 13: ASR and CDA after Pruning (model poisoning).

Increasing the hidden layer size may lead to a small gain in model accuracy but also has downsides like increasing the chance of the model learning the backdoor and overall training time. Another important observation is the higher robustness of ML-ELM against backdoors. Nonetheless, we should remember that Adding more layers to ELM makes it much slower and less efficient considering training time, leading to ELMs being less useful than deep neural networks.

It is also important to consider the potential of other ML-ELM designs in enhancing robustness against backdoor attacks. Although our study employed the most basic ML-ELM structure, various advanced versions of ML-ELM exist [19, 41] that could offer improved defense capabilities. These versions introduce additional layers, complexity, or kernel-based approaches, which might increase resistance to backdoor attacks. However, these advanced models often require significantly more training time and computational resources, which could negate the primary advantages of using ELMs, particularly in comparison to regular deep neural networks. This trade-off between robustness and efficiency raises the question of whether it is sensible to employ such complex ML-ELM designs for the sole purpose of backdoor robustness when simpler and potentially more effective alternatives like DNNs are available.

Regarding Pruning, a naive approach is to perform the fine-tuning part at the end, making it fine-pruning as a defense. As we did the experiments to verify, in most cases, this leads to complete failure of the backdoor and high CDA back from the model. However, we think this solution is unrealistic and useless. Since fine-tuning is training the last layer of the neural network, it follows a fairly similar approach in ELMs as we have to calculate the β from scratch again after masking the neurons in the first layer. This is like training the ELM model from scratch, and in our threat model, it is of no use since the better solution would be for the user to establish the random layer himself again and train the model with his verified clean data.

While pruning was selected as a defense in this study due to its relative simplicity and applicability to the structure of ELMs, we acknowledge that this approach represents only one possible line of defense against backdoor attacks. The focus on pruning is motivated by the constraints of ELM's architecture, which limits the applicability of more complex defense mechanisms typically used in deep learning models. However, this choice does introduce a limitation in our work, as other defenses, potentially more effective, remain unexplored.

7 Conclusions and Future Work

In this study, we confirmed that Extreme Learning Machines are highly susceptible to backdoor attacks, a significant security threat in machine learning models. Despite their unique structure and learning procedures (which make them suitable for lightweight datasets and real-time applications), ELMs demonstrate a high vulnerability to backdoor attacks, especially through data poisoning, achieving an attack success rate of 100% in most cases. These findings underscore the urgent need to enhance security measures for ELMs.

We attempted to deploy a low-rate pruning technique as a defensive measure to maintain the model's robustness against such attacks. However, our results indicated that this defense strategy is ineffective in protecting ELMs from backdoors. As a potential solution, we suggest exploring multi-layer ELM designs to mitigate the impact of backdoor attacks.

Future research could focus on investigating more sophisticated attacks, such as stealthy backdoors, within the ELM domain. However, the primary emphasis should be on developing highly effective defensive methods adapted to ELMs, as current defensive measures are either designed for deep neural networks or are largely ineffective, like pruning.

References

[1] 2023. The Street View House Numbers (SVHN) Dataset. http://ufldl.stanford.edu/housenumbers/ Accessed: 2023-07-10.

[2] Gorka Abad, Jing Xu, Stefanos Koffas, Behrad Tajalli, and Stjepan Picek. 2023. A Systematic Evaluation of Backdoor Trigger Characteristics in Image Classification. *arXiv preprint arXiv:2302.01740* (2023).

[3] Eugene Bagdasaryan and Vitaly Shmatikov. 2021. Blind Backdoors in Deep Learning Models. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, Michael Bailey and Rachel Greenstadt (Eds.). USENIX Association, 1505–1521. https://www.usenix.org/conference/usenixsecurity21/presentation/bagdasaryan

[4] Sartaj Bhuvaji. 2020. Brain Tumor Classification (MRI). https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri Accessed: 2024-06-25.

[5] Cen Chen, Kenli Li, Mingxing Duan, and Keqin Li. 2017. Extreme Learning Machine and Its Applications in Big Data Processing. In *Big Data Analytics for Sensor-Network Collected Intelligence*. Elsevier, 117–150. https://doi.org/10.1016/b978-0-12-809393-1.00006-4

[6] Lu Chen, Na Sun, Chao Zhou, Jianzhong Zhou, Yanlai Zhou, Junhong Zhang, and Qing Zhou. 2018. Flood forecasting based on an improved extreme learning machine model combined with the backtracking search optimization algorithm. *Water* 10, 10 (2018), 1362.

[7] 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

[8] Zhuyun Chen, Konstantinos Gryllias, and Weihua Li. 2019. Mechanical fault diagnosis using convolutional neural networks and extreme learning machine. *Mechanical systems and signal processing* 133 (2019), 106272.

[9] ChenWei Deng, GuangBin Huang, Jia Xu, and JieXiong Tang. 2015. Extreme learning machines: new trends and applications. *Science China. Information Sciences* 58, 2 (2015), 1–16.

[10] Shifei Ding, Xinzheng Xu, and Ru Nie. 2013. Extreme learning machine and its applications. *Neural Computing and Applications* 25, 3-4 (Dec. 2013), 549–556.

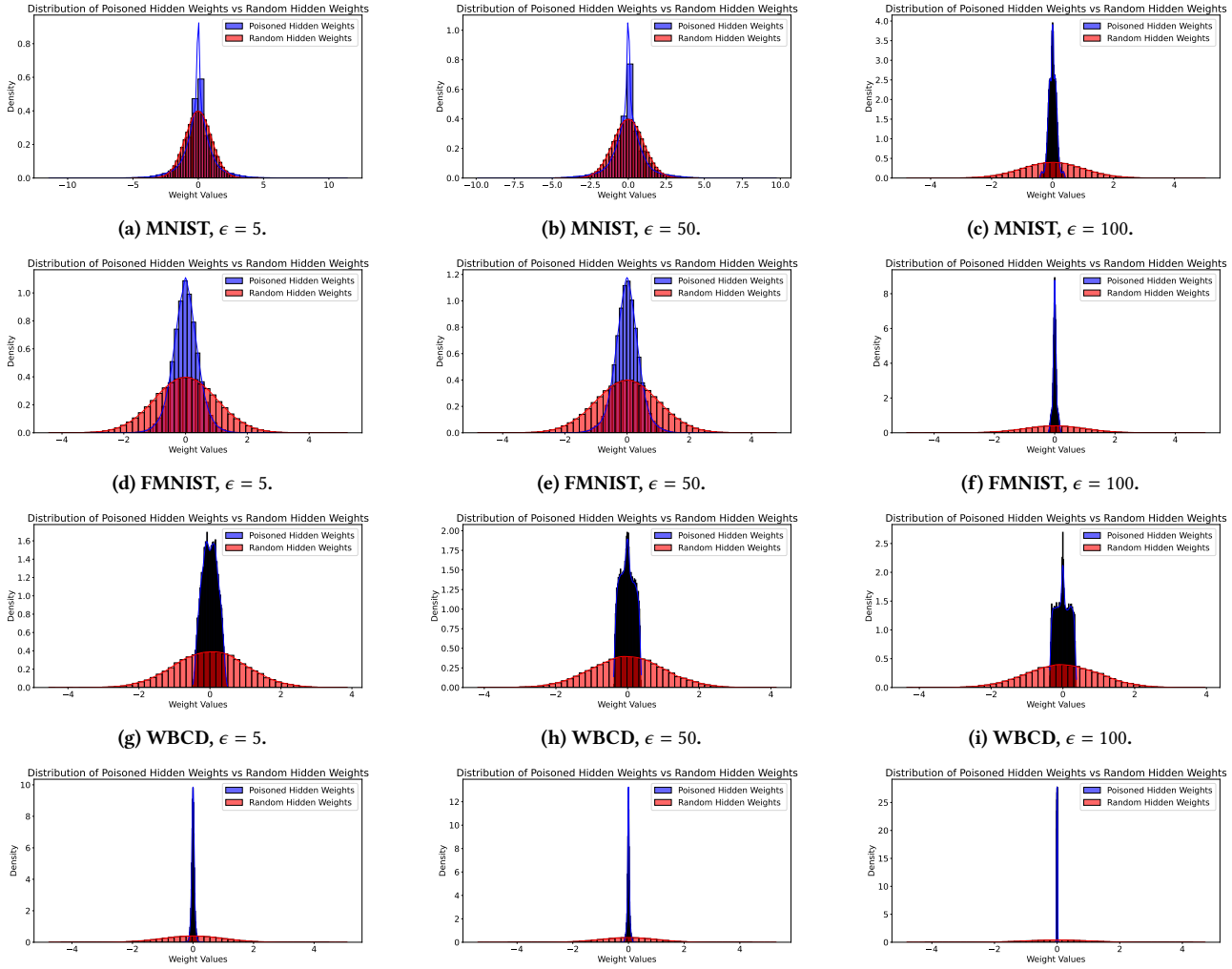
https://doi.org/10.1007/s00521-013-1522-8
[11] Shifei Ding, Han Zhao, Yanan Zhang, Xinzheng Xu, and Ru Nie. 2015. Extreme learning machine: algorithm, theory and applications. *Artif. Intell. Rev.* 44, 1 (2015), 103–115. https://doi.org/10.1007/s10462-013-9405-z
[12] Dheeru Dua and Casey Graff. 2019. UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set. https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/data Accessed: 2024-06-25.
[13] Yansong Gao, Chang Xu, Derui Wang, Shiping Chen, Damith Chinthana Ranasinghe, and Surya Nepal. 2019. STRIP: a defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference, ACSAC 2019, San Juan, PR, USA, December 09-13, 2019*, David Balenson (Ed.). ACM, 113–125. https://doi.org/10.1145/3359789.3359790
[14] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* 7 (2019), 47230–47244. https://doi.org/10.1109/ACCESS.2019.2909068
[15] Wei Guo, Benedetta Tondi, and Mauro Barni. 2021. An Overview of Backdoor Attacks Against Deep Neural Networks and Possible Defences. *CoRR* abs/2111.08429 (2021). arXiv:2111.08429 https://arxiv.org/abs/2111.08429
[16] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. 2022. Handcrafted Backdoors in Deep Neural Networks. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2022/hash/3538a22cd3ceb8f009cc62b9e535c29f-Abstract-Conference.html
[17] Guang-Bin Huang, Qin-Yu Zhu, and Chee Kheong Siew. 2006. Extreme learning machine: Theory and applications. *Neurocomputing* 70, 1-3 (2006), 489–501. https://doi.org/10.1016/j.neucom.2005.12.126
[18] Gizem Atac Kale and Cihan Karakuzu. 2022. Multilayer extreme learning machines and their modeling performance on dynamical systems. *Applied Soft Computing* 122 (2022), 108861.
[19] Ravneet Kaur, Rajendra Kumar Roul, and Shalini Batra. 2023. Multilayer extreme learning machine: a systematic review. *Multimedia Tools and Applications* 82, 26 (2023), 40269–40307.
[20] Stefanos Koffas, Jing Xu, Mauro Conti, and Stjepan Picek. 2022. Can You Hear It?: Backdoor Attacks via Ultrasonic Triggers. In *WiseML@WiSec 2022: Proceedings of the 2022 ACM Workshop on Wireless Security and Machine Learning, San Antonio, TX, USA, 19 May 2022*, Murtuza Jadliwala (Ed.). ACM, 57–62. https://doi.org/10.1145/3522783.3529523
[21] Jie Lai, Xiaodan Wang, Rui Li, Yafei Song, and Lei Lei. 2020. BD-ELM: A regularized extreme learning machine using biased dropconnect and biased dropout. *Mathematical Problems in Engineering* 2020 (2020), 1–7.
[22] Yuan Lan, Zongjiang Hu, Yeng Chai Soh, and Guang-Bin Huang. 2013. An extreme learning machine approach for speaker recognition. *Neural Comput. Appl.* 22, 3-4 (2013), 417–425. https://doi.org/10.1007/s00521-012-0946-x
[23] Yann LeCun. 2023. THE MNIST DATABASE of handwritten digits. http://yann.lecun.com/exdb/mnist/ Accessed: 2023-07-12.
[24] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. 2021. Neural Attention Distillation: Erasing Backdoor Triggers from Deep Neural Networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=9l0K4OM-oXE
[25] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2020. Backdoor Learning: A Survey. *CoRR* abs/2007.08745 (2020). arXiv:2007.08745 https://arxiv.org/abs/2007.08745
[26] Yiming Li, Tongqing Zhai, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2021. Backdoor Attack in the Physical World. *CoRR* abs/2104.02361 (2021). arXiv:2104.02361 https://arxiv.org/abs/2104.02361
[27] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In *Research in Attacks, Intrusions, and Defenses - 21st International Symposium, RAID 2018, Heraklion, Crete, Greece, September 10-12, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 11050)*, Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis (Eds.). Springer, 273–294. https://doi.org/10.1007/978-3-030-00470-5_13
[28] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-5_Liu_paper.pdf
[29] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P Wellman. 2018. Sok: Security and privacy in machine learning. In *2018 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 399–414.
[30] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. 2016. Towards the Science of Security and Privacy in Machine Learning. *CoRR* abs/1611.03814 (2016). arXiv:1611.03814 http://arxiv.org/abs/1611.03814
[31] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. 2018. SoK: Security and Privacy in Machine Learning. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*. IEEE, 399–414. https://doi.org/10.1109/EuroSP.2018.00035
[32] Bart Pleiter, Behrad Tajalli, Stefanos Koffas, Gorka Abad, Jing Xu, Martha Larson, and Stjepan Picek. 2023. Tabdoor: Backdoor Vulnerabilities in Transformer-based Neural Networks for Tabular Data. *arXiv preprint arXiv:2311.07550* (2023).
[33] Xiangyu Qi, Jifeng Zhu, Chulin Xie, and Yong Yang. 2021. Subnet Replacement: Deployment-stage backdoor attack against deep neural networks in gray-box setting. *CoRR* abs/2107.07240 (2021). arXiv:2107.07240 https://arxiv.org/abs/2107.07240
[34] Bo-Yang Qu, BF Lang, Jing J Liang, A Kai Qin, and Oscar D Crisalle. 2016. Two-hidden-layer extreme learning machine for regression and classification. *Neurocomputing* 175 (2016), 826–834.
[35] Zalando Research. 2023. Fashion-MNIST: A MNIST-like fashion product database. https://github.com/zalandoresearch/fashion-mnist Accessed: 2023-07-12.
[36] Dimas Chaerul Ekty Saputra, Khamron Sunat, and Tri Ratnaningsih. 2023. A New Artificial Intelligence Approach Using Extreme Learning Machine as the Potentially Effective Model to Predict and Analyze the Diagnosis of Anemia. *Healthcare* 11, 5 (2023). https://doi.org/10.3390/healthcare11050697
[37] Behrad Tajalli, Gorka Abad, and Stjepan Picek. 2023. Poster: Backdoor Attack on Extreme Learning Machines. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 3588–3590.
[38] Behrad Tajalli, Oguzhan Ersoy, and Stjepan Picek. 2023. On Feasibility of Server-side Backdoor Attacks on Split Learning. *CoRR* abs/2302.09578 (2023). https://doi.org/10.48550/arXiv.2302.09578 arXiv:2302.09578
[39] Jiexiong Tang, Chenwei Deng, and Guang-Bin Huang. 2016. Extreme Learning Machine for Multilayer Perceptron. *IEEE Trans. Neural Networks Learn. Syst.* 27, 4 (2016), 809–821. https://doi.org/10.1109/TNNLS.2015.2424995
[40] Loc Truong, Chace Jones, Brian Hutchinson, Andrew August, Brenda Praggastis, Robert Jasper, Nicole Nichols, and Aaron Tuor. 2020. Systematic Evaluation of Backdoor Data Poisoning Attacks on Image Classifiers. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*. Computer Vision Foundation / IEEE, 3422–3431. https://doi.org/10.1109/CVPRW50498.2020.00402
[41] José A Vásquez-Coronel, Marco Mora, and Karina Vilches. 2023. A Review of multilayer extreme learning machine neural networks. *Artificial Intelligence Review* 56, 11 (2023), 13691–13742.
[42] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2019. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 707–723. https://doi.org/10.1109/SP.2019.00031
[43] Jian Wang, Siyuan Lu, Shui-Hua Wang, and Yu-Dong Zhang. 2022. A review on extreme learning machine. *Multim. Tools Appl.* 81, 29 (2022), 41611–41660. https://doi.org/10.1007/s11042-021-11007-7
[44] Zhiqiong Wang, Mo Li, Huaxia Wang, Hanyu Jiang, Yudong Yao, Hao Zhang, and Junchang Xin. 2019. Breast cancer detection using extreme learning machine based on feature fusion with CNN deep features. *IEEE Access* 7 (2019), 105146–105158.

A Additional Details on Model Poisoning Attack

For further analysis, we investigate the difference between random initialized neurons and those transplanted in the ELM via model poisoning. Figure 14 demonstrates values distribution of 1*st* layer neurons in an ELM.

B Additional Details on Data Poisoning Results

Figure 15 provides the results for the SVHN dataset. The results show some differences from MNIST and FMNIST. However, there are still some similarities. CDAs are still close to their related BAs (which means the attack does not harm the benign task). The strongest takeaway from the SVHN results is that ASR is higher than CDA and BA in almost all cases. This holds for FMNIST to some extent as well but is rarely seen in MNIST. We conjecture that this is mostly because of the overall capacity of the ELMs. As the dataset becomes more complex, ELM has less capacity to learn the real data distribution but enough for backdoor samples. Thus, ASR remains high while BA and CDA decrease when using more complex datasets. Hidden layer size has a minimal impact on ASR



(a) MNIST, $\epsilon = 5$. **(b) MNIST, $\epsilon = 50$.** **(c) MNIST, $\epsilon = 100$.**

(d) FMNIST, $\epsilon = 5$. **(e) FMNIST, $\epsilon = 50$.** **(f) FMNIST, $\epsilon = 100$.**

(g) WBCD, $\epsilon = 5$. **(h) WBCD, $\epsilon = 50$.** **(i) WBCD, $\epsilon = 100$.**

(j) BRATC, $\epsilon = 5$. **(k) BRATC, $\epsilon = 50$.** **(l) BRATC, $\epsilon = 100$.**

Figure 14: Distribution plots for 1^{st} layer's neuron values. The weights are initialized either by the random function or transferred values from poisoned neurons from the attacker's trained model.

in almost all cases. The most noticeable results are in low poisoning rates, i.e., $\epsilon = 0.2$. Increasing the hidden layer size to more than 1000 nodes for other cases does not affect the ASR. Poisoning rate also does not have an important effect on ASR unless using low poisoning rates, and after some points, all backdoor attacks achieve very high ASRs. The trigger size impacts ASR the most and is mainly seen in ML-ELM results (again, the most robust ELM).

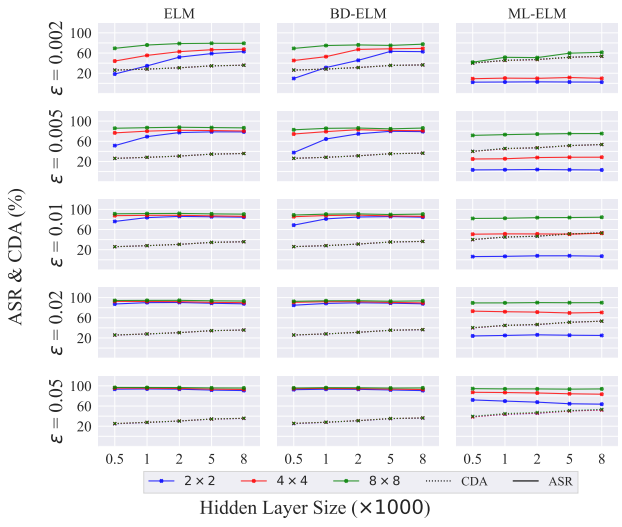


Figure 15: ASR and CDA for SVHN.