International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-024-02363-5

ORIGINAL ARTICLE

Backdoor learning curves: explaining backdoor poisoning beyond influence functions

Antonio Emanuele Cinà[1] · Kathrin Grosse[3] · Sebastiano Vascon[2] · Ambra Demontis[4] · Battista Biggio[4] · Fabio Roli[1] · Marcello Pelillo[2]

Received: 17 July 2023 / Accepted: 6 August 2024

Abstract
Backdoor attacks inject poisoning samples during training, with the goal of forcing a machine learning model to output an attacker-chosen class when presented with a specific trigger at test time. Although backdoor attacks have been demonstrated in a variety of settings and against different models, the factors affecting their effectiveness are still not well understood. In this work, we provide a unifying framework to study the process of backdoor learning under the lens of incremental learning and influence functions. We show that the effectiveness of backdoor attacks depends on (i) the complexity of the learning algorithm, controlled by its hyperparameters; (ii) the fraction of backdoor samples injected into the training set; and (iii) the size and visibility of the backdoor trigger. These factors affect how fast a model learns to correlate the presence of the backdoor trigger with the target class. Our analysis unveils the intriguing existence of a region in the hyperparameter space in which the accuracy of clean test samples is still high while backdoor attacks are ineffective, thereby suggesting novel criteria to improve existing defenses.

Keywords Backdoor poisoning · Influence functions · Poisoning · Machine learning · Adversarial machine learning · Security

1 Introduction

Machine learning models are vulnerable to backdoor poisoning [1–4]. These attacks consist of injecting poisoning samples at training time, with the goal of forcing the trained model to output an attacker-chosen class when presented with a specific trigger at test time, while working as expected otherwise. To this end, the poisoning samples typically need not only to embed such a *backdoor trigger* themselves, but also to be labeled as the attacker-chosen class. As backdoor poisoning preserves model performance on clean test data, it is not straightforward for the victim to realize that the model has been compromised.

Backdoor poisoning has been demonstrated in a plethora of scenarios [3–5]. In the most common scenario, the user is assumed to download a pre-trained, backdoored model from an untrusted source, to subsequently fine-tune it on their data [3]. As backdoors typically remain effective even after this fine-tuning step, they may successfully be exploited by the attacker at test time [1, 2]. Alternatively, the attacker is assumed to alter part of the training data collected by the user, either to train the model from scratch or to fine-tune a

✉ Antonio Emanuele Cinà
antonio.cina@unige.it

Kathrin Grosse
kathrin.grosse@epfl.ch

Sebastiano Vascon
sebastiano.vascon@unive.it

Ambra Demontis
ambra.demontis@unica.it

Battista Biggio
battista.biggio@unica.it

Fabio Roli
fabio.roli@unige.it

Marcello Pelillo
pelillo@unive.it

1 DIBRIS, University of Genoa, Genoa, Italy

2 DAIS, Ca' Foscari University of Venice, Venezia, Italy

3 VITA Lab, École Polytechnique Fédéerale de Lausanne, Lausanne, Switzerland

4 DIEE, University of Cagliari, Cagliari, Italy

pre-trained model via transfer learning [6, 7]. In such cases, the training labels of poisoning samples may be either fully controlled by the attacker [6] or subject to validation by the victim [7], depending on the considered threat model. Backdoor attacks have been shown to be effective in different applications and against a variety of models, including vision [1] and language models [8], graph neural networks [9], and even reinforcement learning [10]. However, it is still unclear which factors influence the ability of a model to learn a backdoor, i.e., to classify the test samples containing the backdoor trigger as the attacker-chosen class.

In this work, we analyze the process of *backdoor learning* to identify the main factors affecting the vulnerability of machine learning models against this attack. To this end, we propose a framework that characterizes the backdoor learning process. The learning process of humans is usually portrayed using learning curves, a graphical representation of the relationship between the hours spent practicing and the proficiency to accomplish a given task. Inspired by this concept, we introduce the notion of *backdoor learning curves* (Sect. 2). To generate these curves, we formulate backdoor learning as an incremental learning problem and assess how the loss on the backdoor samples decreases as the target model gradually learns them. These backdoor learning curves are independent of the threat model as they capture training data and fine-tuning attacks. The slope of this curve, the *backdoor learning slope*, which is connected to the notion of influence functions, quantifies the speed with which the model learns the backdoor samples and hence its vulnerability. Additionally, to provide further insights about the backdoor's influence on the learned classifiers, we propose a way to quantify the *backdoor impact on learning parameters*, i.e., how much the parameters of a model deviate from the initial values when the model learns a backdoor.

Our experimental analysis (Sect. 3) shows that the factors influencing the success of backdoor poisoning are: (i) the fraction of backdoor samples injected into the training data; (ii) the size of the backdoor trigger; and (iii) the complexity of the target model, controlled via its hyperparameters. Concerning the latter, our experimental findings reveal a region in the hyperparameter space where models are highly accurate on clean samples while also being robust to backdoor poisoning. This region exists as, to learn a backdoor, the target model is required to increase the complexity of its decision function significantly, and this is not possible if the model is sufficiently regularized. This observation will help identify novel criteria to improve existing defenses and inspire new countermeasures.

To summarize, the main contributions of this work are:

- We introduce *backdoor learning curves* as a powerful tool to thoroughly characterize the backdoor learning process;
- We introduce a metric, named *backdoor learning slope*, to quantify the ability of the classifier to learn backdoors;
- We identify three important factors that affect the vulnerability against backdoors;
- We unveil a region in the hyperparameter space in which the classifiers are highly accurate and robust against backdoors, which supports novel defensive strategies.

We conclude the paper by discussing related work in Sect. 4, along with the limitations of our approach and promising future research directions in Sect. 5.

2 Backdoor learning curves

In this section, we introduce our framework to characterize backdoor poisoning by means of learning curves and their slope. Afterward, we introduce two measures to quantify the backdoor impact on the model's parameters.

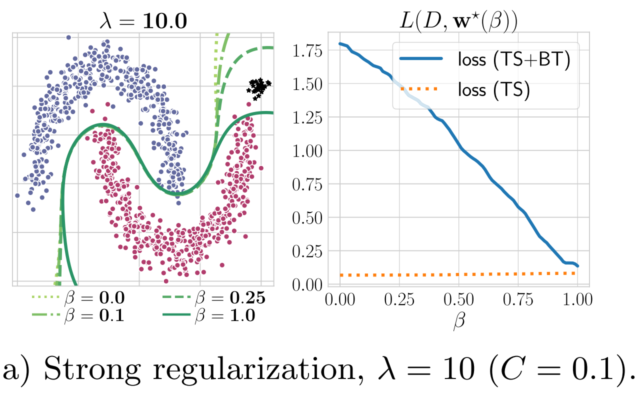
Notation. We denote the input data and their labels respectively with $\boldsymbol{x} \in \mathbb{R}^d$ and $y \in \{1, .., c\}$, being c the number of classes. We refer to the untainted, clean training data as $\mathcal{D}_{\text{tr}} = (\boldsymbol{x}_i, y_i)_{i=1}^n$, and to the backdoor samples injected into the training set as $\mathcal{P}_{\text{tr}} = (\hat{\boldsymbol{x}}_j, \hat{y}_j)_{j=1}^m$. We refer to the clean test samples as $\mathcal{D}_{\text{ts}} = (\boldsymbol{x}_t, y_t)_{t=1}^k$ and to the test samples containing the backdoor trigger as $\mathcal{P}_{\text{ts}} = (\hat{\boldsymbol{x}}_t, \hat{y}_t)_{t=1}^k$.

Backdoor learning curves. We leverage previous work from incremental learning [11, 12] to study how gradually incorporating backdoor samples affects the learned classifier. In mathematical terms, the learning problem can be formalized as:

$$\begin{aligned} \boldsymbol{w}^\star(\beta) &\in \underset{\boldsymbol{w}}{\arg\min}\, \mathcal{L}(\mathcal{D}_{\text{tr}} \cup \mathcal{P}_{\text{tr}}, \boldsymbol{w}) \\ &= L(\mathcal{D}_{\text{tr}}, \boldsymbol{w}) + \beta L(\mathcal{P}_{\text{tr}}, \boldsymbol{w}) + \lambda \Omega(\boldsymbol{w}), \end{aligned} \tag{1}$$

where L is the loss attained on a given dataset by the classifier with parameters $\boldsymbol{w}$, and $\mathcal{L}$ is the loss computed on the training points and the backdoor samples, which also includes a regularization term $\Omega(\boldsymbol{w})$ (e.g., $\|\boldsymbol{w}\|_2^2$), weighed by the regularization hyperparameter λ. To gradually incorporate the backdoor samples $\mathcal{P}_{\text{tr}}$ into the learning process, we introduce the hyperparameter $\beta \in [0, 1]$, and increase it from 0 (unpoisoned classifier) to 1 (poisoned classifier). As β increases, the classifier gradually learns the backdoor by adjusting its parameters; for this reason, we make the dependency of the optimal weights $\boldsymbol{w}^\star$ on β explicit as $\boldsymbol{w}^\star(\beta)$ [1].

[1] Recall that the formulation reported in Eq. (1) encompasses many existing learning algorithms, including support vector machines (SVMs), ridge and logistic classifiers. For example, considering either $\beta = 0$ or $\beta = 1$, the SVM learning problem amounts to minimizing $C \cdot \left(L(\mathcal{D}_{\text{tr}}, \boldsymbol{w}) + \beta L(\mathcal{P}_{\text{tr}}, \boldsymbol{w})\right) + \frac{1}{2}\|\boldsymbol{w}\|_2^2$, which is equivalent to



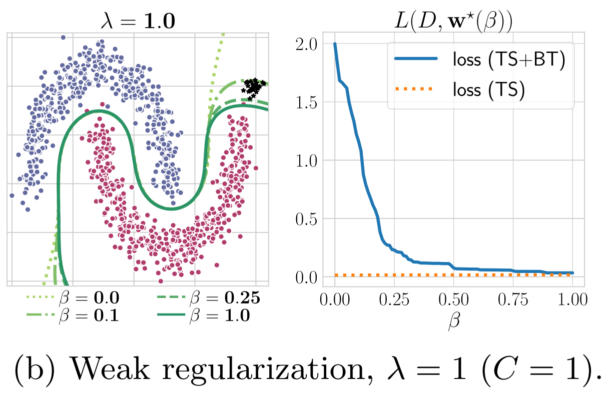


Fig. 1 Backdoor learning curves. Considering an SVM with the RBF kernel ($\gamma = 10$) on a toy dataset in two dimensions, we show the influence of model complexity (controlled by the regularization hyperparameter $\lambda = \frac{1}{C}$) on backdoor learning. For both the strong (*left*) and weak (*right*) regularization settings, we report two plots. The left plot shows the two-dimensional data (dots) and decision surface for different values of β (green lines). The right plot shows the backdoor learning curve, i.e. how the loss decreases as β ranges from 0 to 1, which amounts to learning the backdoor samples. We plot both the loss on the clean test samples (orange dotted line) and on the test samples with the backdoor trigger (blue line). The slope of these curves represents the speed with which the model learns to classify the backdoor samples (black dots) as blue dots, unveiling that strong regularization slows down such a process

We now define the *backdoor learning curve* as the curve showing the behavior of the classifier loss $L(\mathcal{P}_{\text{ts}}, \boldsymbol{w}^\star(\beta))$ on the test samples with the backdoor trigger as a function of β. In the following, we abbreviate $L(\mathcal{P}_{\text{ts}}, \boldsymbol{w}^\star(\beta))$ as L. Intuitively, the faster the backdoor learning curve decreases, the easier the target model is backdoored. The exact details of how the model is backdoored do not matter for this analysis, e.g. our approach captures for example both the setting where the training data is altered as well as the setting where fine-tuning data is tampered with.

We give an example of two such curves under different regularizations in Fig. 1. The left plots depict a strongly regularized classifier. The corresponding backdoor learning curve (on the right) shows that the classifier achieves low loss and high accuracy on the backdoor samples only after poisoning (when $\beta = 1$), i.e. even when the loss on the backdoor samples is considered equally important to the loss on the training samples. The classifier on the right, instead, is less regularized and thus more complex. Consequently, this classifier learns to incorporate the backdoor samples much faster (at low β), namely when the loss on the backdoor points is taken into account less than the one on the training data. This highlights that this classifier is probably more vulnerable to this attack.

Backdoor learning slope. We quantify how fast an untainted classifier can be poisoned by proposing a novel measure, namely the *backdoor learning slope*, that measures the velocity with which the classifier learns to classify the backdoor samples correctly. This measure allows us to compare the vulnerability of a classifier trained with different hyperparameters or consider different poisoning scenarios (e.g. when the attacker can inject a different number of poisoning points or create triggers with different sizes), allowing us to identify factors relevant to backdoor learning. Moreover, as we will show, this measure can be used by the system designer to choose an appropriate combination of hyperparameters for the task at hand. To this end, we define the *backdoor learning slope* as the gradient of the backdoor learning curve at $\beta = 0$, capturing the velocity of the curve on learning the backdoor. Formally, the *backdoor learning slope* can be formulated as follow:

$$\frac{\partial L(\mathcal{P}_{\text{ts}}, \boldsymbol{w}^\star(\beta))}{\partial \beta} = \frac{\partial L}{\partial \boldsymbol{w}} \frac{\partial \boldsymbol{w}^\star}{\partial \beta}, \tag{2}$$

where the first term is straightforward to compute, and the second term implicitly captures the dependency of the optimal weights on the hyperparameter β. In other words, it requires us to understand how the optimal classifier parameters change when gradually increasing β from 0 to 1, i.e., while incorporating the backdoor samples into the learning process.

To compute this term, as suggested in previous work in incremental learning [11], we assume that, while increasing β, the solution maintains the optimality (Karush–Kuhn–Tucker, KKT) conditions intact. This equilibrium implies that $\nabla_\beta \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}^\star) + \frac{\partial \boldsymbol{w}^\star}{\partial \beta} \nabla^2_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}^\star) = \mathbf{0}$. Based on this condition, we obtain the derivative of interest,

Footnote 1 (Continued)

our formulation if one sets L to be the hinge loss, $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$, and $\lambda = \frac{1}{C}$.

$$\frac{\partial \boldsymbol{w}^\star}{\partial \beta} = -(\nabla^2_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}^\star))^{-1} \cdot \nabla_\beta \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}^\star). \tag{3}$$

Substituting it in Eq. 2 we obtain the complete gradient:

$$\frac{\partial L(\mathcal{P}_{\mathrm{ts}}, \boldsymbol{w}^\star(\beta))}{\partial \beta} = -\nabla_{\boldsymbol{w}} L \cdot (\nabla^2_{\boldsymbol{w}} \mathcal{L})^{-1} \cdot \nabla_\beta \nabla_{\boldsymbol{w}} \mathcal{L}. \tag{4}$$

The gradient in Eq. 4 corresponds to the sum of the pairwise influence function values $\mathcal{I}_{\text{up, loss}}(\boldsymbol{x}_{\text{tr}}, \boldsymbol{x}_{\text{ts}})$ used by Koh et al. [13]. The authors indeed proposed to measure how relevant a training point is for the predictions of a given test point by computing $\frac{\partial L}{\partial \beta}\big|_{\beta=0} = \sum_t \sum_j \mathcal{I}_{\text{up, loss}}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{x}}_j)$. To understand how this gradient can be efficiently computed via Hessian-vector products and other approximations, we refer the reader to [13] as well as to recent developments in gradient-based bilevel optimization [14–16]. Moreover, we show in Sect. 3.2.5 (Figs. 13 and 14) an example of the usage of influence functions for weakly and strongly regularized models.

The main difference between the approach by Koh et al. [13] and ours stems from their implicit treatment of regularization and our interest in understanding vulnerability to a subset of backdoor training points, rather than in providing prototype-based explanations. However, directly using the gradient of the loss wrt. β comes with two disadvantages. First, the slope is inverse to β, and second, to obtain results comparable across classifiers, we need to rescale the slope. We thus transform the gradient as:

$$\theta = -\frac{2}{\pi} \arctan\left(\frac{\partial L}{\partial \beta}\bigg|_{\beta=0} \right) \in [-1, 1], \tag{5}$$

where we use the negative sign to have positive values correlated with faster backdoor learning (i.e., the loss decreases faster as β grows). Computing $2/\pi$ of the gradient allows us to rescale the slope to be in the interval between $[-1, 1]$. Hence, a value around 0 implies that the loss of the backdoor samples does not decrease. In other words, the classifier does not learn the backdoor trigger and is hence very robust.

Backdoor impact on learning parameters. After introducing the previous plot and measure, we can quantify how backdoors are learned by the model. To provide further insights about the backdoor's influence on the learned classifier, we propose to monitor how the classifier's parameters deviate from their initial, unbackdoored values once a backdoor is added. Our approach below captures only convex learners. As shown by Zhang et al. [17], the impact of a network weight in non-convex classifiers' decisions depends on the layer of which it is part. Therefore, measuring the parameter deviation in the non-convex case is challenging, and we leave this unsolved problem for future work.

To capture the backdoor impact on learning parameters in the convex case, we consider the initial weights $\boldsymbol{w}_0 = \boldsymbol{w}^\star(\beta = 0)$ and $\boldsymbol{w}_\beta = \boldsymbol{w}^\star(\beta)$ for $\beta > 0$, and measure two quantities:

$$\rho = \|\boldsymbol{w}_\beta\| \in [0, \infty), \text{ and } \nu = \frac{1}{2}\left(1 - \frac{\boldsymbol{w}_0^\top \boldsymbol{w}_\beta}{\|\boldsymbol{w}_0\| \|\boldsymbol{w}_\beta\|}\right) \in [0, 1]. \tag{6}$$

The first measure, ρ, quantifies the change of the weights when β increases. This quantity is equivalent to the regularization term used for learning. The second one, ν, quantifies the change in orientation of the classifier. In a nutshell, we compute the angle between the two vectors and rescale it to be in the interval of [0, 1]. Both metrics are defined to grow as $\beta \to 1$, in other words the backdoored classifier deviates more and more from the original classifier. Consequently, in the empirical parameter deviation plots in Sect. 3.2, we report the value of $\rho(\nu)$ (on the y-axis) as β (on the x-axis) varies from 0 to 1, to show how the classifier parameters are affected by backdoor learning.

3 Experiments

Employing the previously proposed methodology, we carried out an empirical analysis of linear and nonlinear classifiers. In this section, we start with the experiments aimed at studying the impact of different factors on backdoor learning. To this end, we employ the backdoor learning curves and the backdoor learning slope to study how the capacity of the model to learn backdoors changes when (a) varying the model's complexity, defined by its hyperparameters, (b) the attacker's strength, defined by the percentage of poisoning samples in the training set and (c) the trigger size and visibility. Our results show that these components significantly determine how fast the backdoor is learned, and consequently, the model's vulnerability. Then, leveraging the proposed measures to analyze how the classifier's parameters change during backdoor learning, we provide further insights into the effect of the aforementioned factors on the trained model. The results presented in this section will help identify novel criteria to improve existing defenses and inspire new countermeasures. The source code is available on the author's GitHub page.[2]

3.1 Experimental setup

Our work investigates which factors influence backdoor vulnerability considering convex learners and neural networks.

[2] https://github.com/Cinofix/backdoor_learning_curves.

In the following, we describe our datasets, models, and the backdoor attacks studied in our experiments.

Datasets. We carried out our experiments on MNIST [18], CIFAR10 [19] and Imagenette [20].[3] Supplementary details on the datasets are reported in Appendix A.

When adopting convex learners, we consider the two-class subproblems as in the work by Saha et al. [7] and Suya et al. [21]. On MNIST, we choose the pairs 7 vs 1, 3 vs 0, and 5 vs 2, as our models exhibited the highest clean accuracy on these pairs. On CIFAR10, analogous to prior work [7], we choose *airplane vs frog*, *bird vs dog*, and *airplane vs truck*. On Imagenette we randomly choose *tench vs truck*, *cassette player vs church*, and *tench vs parachute*. For each two-class subtask, we use 1500 and 500 samples as training and test set respectively. In the following section, we focus on the results of one pair on each dataset: 7 vs 1 on MNIST, *airplane vs frog* on CIFAR10, and *tench vs truck* on Imagenette. The results of the other pairs (reported in Appendix B) are analogous. When testing our framework against neural networks, we train on all ten classes of Imagenette. We use 70% and 30% of the entire dataset for training and testing, respectively.

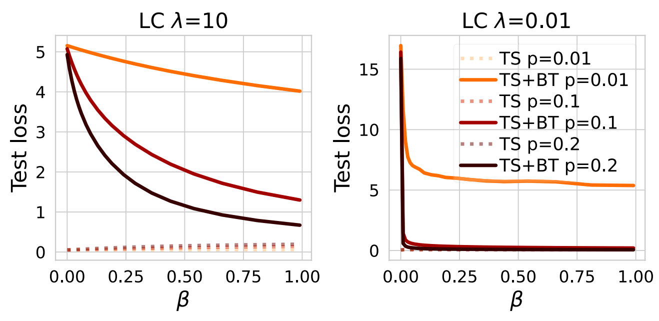
Models and training phase. To thoroughly analyze how learning a backdoor affects a model, we consider different convex learning algorithms, including linear Support Vector Machines (SVMs), Logistic Regression Classifiers (LCs), Ridge Classifiers (RCs), nonlinear SVMs using the Radial Basis Function (RBF) kernel, and deep neural networks. We train the classifiers directly on the pixel values scaled in the range [0, 1] on the MNIST dataset. For CIFAR10 and Imagenette, we instead consider a transfer learning setting frequently adopted in the literature [13, 22, 23]. Like Saha et al. [7], we use the pre-trained model AlexNet [24] as a feature extractor. The convex learners are then trained on top of the feature extractor. We study these convex learners due to their broad usage in industry [21], derived from their excellent results with smaller dataset [25], and good interpretability [26, 27]. In addition, we include in our evaluation pre-trained Resnet18 and Resnet50 [28] deep neural networks, sourced from Torchvision [29], which stand as some of the most extensively employed architectures [17]. These networks are fine-tuned to classify samples from the Imagenette dataset accurately.

Hyperparameters. The choice of hyperparameters has a relevant impact on the learned decision function. For example, some of these parameters control the complexity of the learned function, which may lead to overfitting [30], thereby potentially compromising classification accuracy on test samples. We argue that a high complexity may also lead to higher importance to outlying samples, including backdoors, and thus has a crucial impact on the capacity of the model to learn backdoors. To verify our hypothesis, we consider different configurations of the models' hyperparameters. For convex learners, we study two hyperparameters that impact model complexity, i.e., the regularization hyperparameter $\lambda = \frac{1}{C}$ and the RBF kernel hyperparameter γ. To this end, we take 10 values for λ on a uniformly spaced interval on a log scale from 1e−04 to 1e+02. For the Imagenette dataset we extend this interval in [1e−05, 1e+02]. Concerning the RBF kernel, we let γ take 5 uniformly spaced values on a log scale in [5e−04, 5e−02] for MNIST, [1e−04, 1e−02] for CIFAR, and [1e−05, 1e−03] for Imagenette. Furthermore, we take 10 values of λ in the log scale uniformly spaced interval [1e−01, 1e+02] for the RBF kernel. This allowed us to study a combination of 10 and 50 hyperparameters for linear classifiers and RBF SVM, respectively.

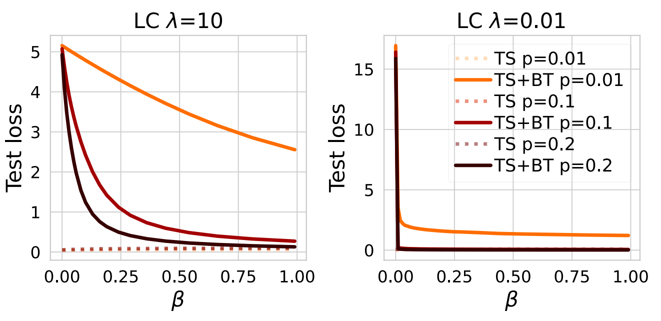
For deep neural networks, we consider two different numbers of epochs: 10 and 50, and increase the number of neurons when using Resnet50 instead of Resnet18. Whereas size intuitively correlates with complexity, previous works, including [31], show that decreasing the number of training epochs reduces the complexity of the trained network as well. Conversely, increasing epochs leads to overfitting on the training dataset, thus, a more complex decision function. Each network is fine-tuned using the SGD optimizer with a learning rate of 0.001, a momentum of 0.9, and a batch size of 256.

Backdoor attacks. We implement the backdoor attacks proposed by Gu et al. [1] against MNIST and CIFAR10. More concretely, we use a random 3×3 patch as the trigger for MNIST, while on CIFAR10, we increase the size to 8×8 to strengthen the attack [7]. We add the trigger pattern in the lower right corner of the image [1]. Samples from MNIST and CIFAR10 with and without trigger can be found in Fig. 12. However, in contrast to previous approaches [1], we use a separate trigger for each base-class *i*. The reason is that our study encompasses linear models that are unable to associate the same trigger pattern to two different classes. Using different trigger patterns, we enhance the effectiveness of the attack on these linear models. On the Imagenette dataset, we use the backdoor trigger developed by [32]. This attack injects a patterned perturbation mask into training samples to open the backdoor. A constant value c_m refers to the maximum allowed intensity. We apply the backdoor attacks to 10% of the training data if not stated otherwise, and, as done by Gu et al. [1], we force the backdoored model to predict the i-th class as class $(i+1)\%n\_classes$ when the trigger is shown. We also report additional experiments concerning variations in the trigger's size or visibility.

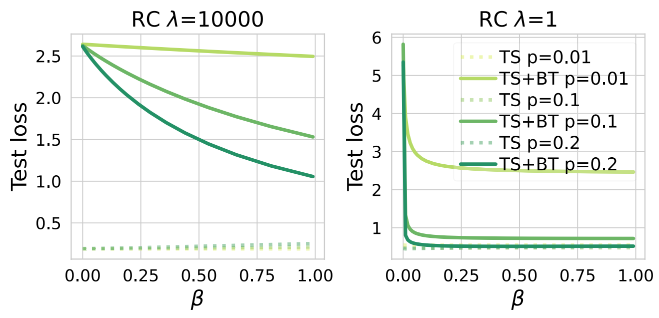
[3] Imagenette is a subset of 10 classes from Imagenet. We use the 320 px version, where the shortest side of each image is resized to that size.



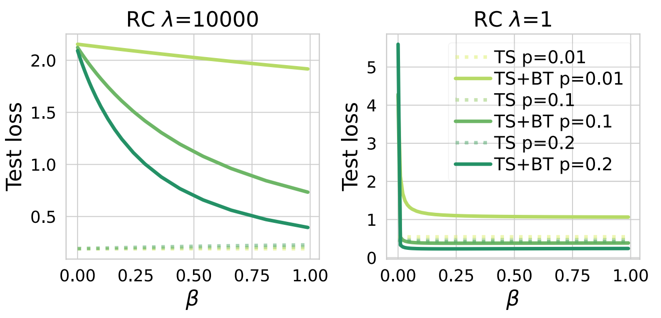
(a) MNIST trigger size $3\times\ 3$.



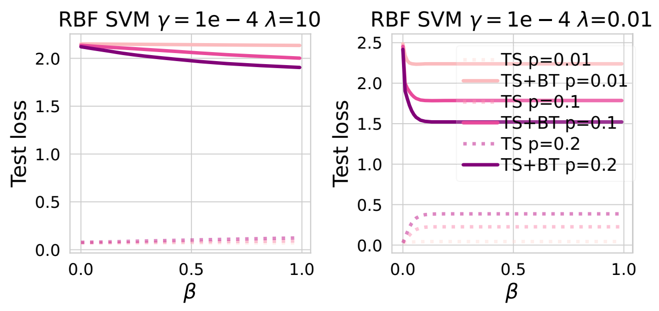
(b) MNIST trigger size $6\times\ 6$.



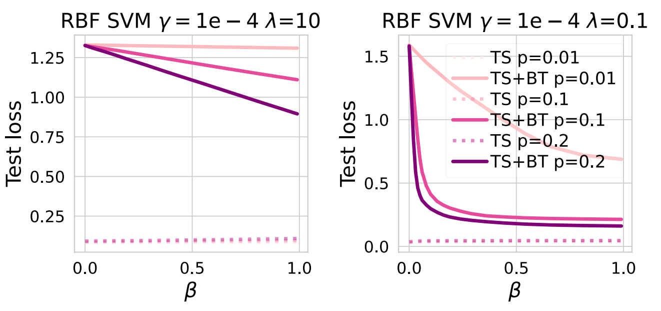
(c) CIFAR10 trigger size $8\times\ 8$.



(d) CIFAR10 trigger size $16\times\ 16$.



(e) Imagenette trigger visibility $c_m\ =10$.



(f) Imagenette trigger visibility $c_m\ =75$.

Fig. 2 Backdoor learning curves for: (top row) logistic classifier (LC) on MNIST 7 vs. 1 with $\lambda \in \{10, 0.01\}$ and trigger size 3×3 *(left)* or 6×6 (right); (middle row) Ridge classifier on CIFAR10 *airplane vs frog* with $\lambda \in \{100000, 100\}$ and trigger size 8×8 (left) or 16×16 (right); (bottom row) RBF SVM with $\gamma = 1\text{e}{-}04$ on Imagenette *tench vs truck* with $\lambda \in \{10, 0.1\}$ and trigger visibility $c_m = 10$ (left) or $c_m = 75$ (right). Darker lines represent a higher fraction of poisoning samples p injected into the training set. We report the loss on the clean test samples (TS) with a dashed line and on the test samples with the backdoor trigger (TS+BT) with a solid line

3.2 Experimental results

In the following, we now discuss our experimental results obtained with the datasets, classifiers, and backdoor attacks described above.

3.2.1 Backdoor learning curves

Here we present the results obtained using the learning curves that we proposed to study the impact of three different factors on the backdoor learning process: (i) *model complexity*, (ii) the *fraction of backdoor samples injected*, and (iii) the *size and visibility of the backdoor trigger*. We report the impact of these factors on the backdoor learning curves in Figs. 2 and 3. More specifically, in Fig. 2 we consider convex classifiers (i.e. LC, RC and RBF SVM) trained on two-class subproblems (MNIST, CIFAR10, and Imagenette), whereas in Fig. 3 we show the results for Resnet18 trained on all the ten classes of Imagenette.

To analyze the first factor, we report the results on the same classifiers, changing the hyperparameters that influence their corresponding complexity. In the case of convex learners, we test different values of the regularization coefficient, while for Resnet18, we increase the number of epochs.