Mithridates: Auditing and Boosting Backdoor Resistance of Machine Learning Pipelines

Eugene Bagdasarian
University of Massachusetts Amherst
Amherst, USA
eugene@cs.umass.edu

Vitaly Shmatikov
Cornell Tech
New York, USA
shmat@cs.cornell.edu

Abstract

Machine learning (ML) models trained on data from potentially untrusted sources are vulnerable to poisoning. A small, maliciously crafted subset of the training inputs can cause the model to learn a "backdoor" task (e.g., misclassify inputs with a certain feature) in addition to its main task. Recent research proposed many hypothetical backdoor attacks whose efficacy depends on the configuration and training hyperparameters of the target model. At the same time, state-of-the-art defenses require massive changes to the existing ML pipelines and protect only against some attacks.

Given the variety of potential backdoor attacks, ML engineers who are not security experts have no way to measure how vulnerable their current training pipelines are, nor do they have a practical way to compare training configurations so as to pick the more resistant ones. Deploying a defense may not be a realistic option, either. It requires evaluating and choosing from among dozens of research papers, completely re-engineering the pipeline as required by the chosen defense, and then repeating the process if the defense disrupts normal model training (while providing theoretical protection against an unknown subset of hypothetical threats).

In this paper, we aim to provide ML engineers with pragmatic tools to *audit* the backdoor resistance of their training pipelines and to *compare* different training configurations, to help choose the one that best balances accuracy and security.

First, we propose a universal, attack-agnostic resistance metric based on the minimum number of training inputs that must be compromised before the model learns any backdoor.

Second, we design, implement, and evaluate Mithridates, a multi-stage approach that integrates backdoor resistance into the training-configuration search. ML developers already rely on hyperparameter search to find configurations that maximize the model's accuracy. Mithridates extends this tool to also order configurations based on their backdoor resistance. We demonstrate that Mithridates discovers configurations whose resistance to multiple types of backdoor attacks increases by 3-5x with only a slight impact on accuracy. We also discuss extensions to AutoML and federated learning.

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690337

CCS Concepts

• **Security and privacy → Software and application security**; • **Computing methodologies → Machine learning approaches**; **Artificial intelligence**.

Keywords

ML security, backdoors, hyperparameter search

ACM Reference Format:
Eugene Bagdasarian and Vitaly Shmatikov. 2024. Mithridates: Auditing and Boosting Backdoor Resistance of Machine Learning Pipelines. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3690337

1 Introduction

Many machine learning models are trained on data from multiple sources, not all trustworthy. An attacker who poisons a small fraction of the training data can influence what the model learns [12]. For example, backdoor attacks [20] cause the model to learn an adversary-chosen task in addition to its main task.

Backdoor attacks target different domains with artificial [39], physical [71], or semantic [131] triggers. Backdoor tasks can be simple (e.g., misclassify inputs with a trigger feature) or add complex functionality to the model [6]. There is extensive literature on backdoor defenses, too. Earlier defenses [114, 120] were vulnerable to adaptive attacks [107]. State-of-the-art defenses require massive changes to model training, e.g., transforming supervised into unsupervised learning [48] or adding whole new learning stages to "unlearn" potential backdoors [67]. The resulting imbalance favors the attacker: even a weak poisoning attack requires the defender to introduce and maintain complex modifications to their ML pipelines—see Table 1.

ML tools are becoming a commodity, but training and deployment of ML models still requires significant engineering and scientific efforts [60]. We conjecture that, outside of major enterprises, engineers who deploy and maintain ML pipelines are not equipped to evaluate the research literature on backdoor defenses and will not deploy defenses that require substantial changes. Furthermore, these defenses cannot be deployed in pipelines that involve third-party MLaaS [93] where access and visibility are limited and engineers cannot observe (e.g., inspect gradients) or modify the training.

In this paper, we aim to provide ML engineers with pragmatic tools for two tasks (see Fig. 1):

(1) **Audit**: given a training pipeline in a particular configuration, measure its backdoor resistance.
(2) **Search**: discover high-resistance configurations that also achieve high accuracy on the main task.

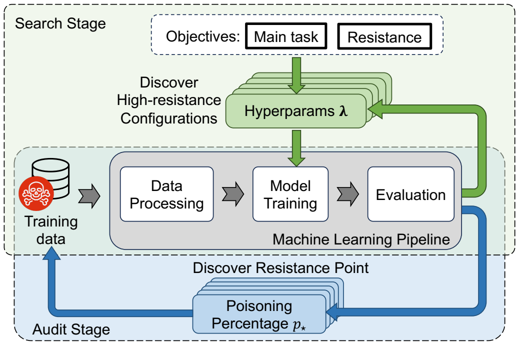

Figure 1: Overview of Mithridates.

We use the term "natural resistance" for how well a given configuration resists backdoor attacks in the absence of any dedicated defenses. Measuring natural resistance is not trivial because the specific backdoor attack is not known in advance, thus the metric must be universal and apply to all realistic backdoors. Boosting natural resistance requires comparing multiple configurations—while maintaining the primary objective, i.e., accuracy on the main task.

Our approach is motivated by two observations. First, efficacy of backdoor attacks strongly depends on the target model's hyperparameters, as previously observed in [98, 100] for specific model-attack combinations. Unfortunately, this observation alone does not provide any guidance for how to generically measure the resistance of a given pipeline to unknown attacks, nor how to boost this resistance by searching through possible configurations.

Second, developers already search for hyperparameters that maximize validation accuracy of their models (see Figure 1), and hyperparameter-tuning techniques are part of commodity ML frameworks such as Ray [83]. We thus focus on hyperparameter search as the one tool that (a) does not require disruptive changes to the training process, and (b) is already used by ML developers.

Our contributions. We design, implement, and evaluate Mithridates,[1] an empirical method that (1) measures the model's natural resistance to learning from partially poisoned data, and (2) performs multi-objective hyperparameter search [36] to balance validation accuracy and backdoor resistance. The key advantage of our approach is that hyperparameter search configures the pipeline rather than modifies it. It can be applied regardless of the model's architecture, task, training regime, etc.

Both parts of Mithridates require a *backdoor-agnostic metric* of natural resistance. We use the "resistance point," i.e., the minimum poisoning percentage that makes the backdoor effective. This metric is universal and applies to any backdoor attack. It is also actionable: engineers may accept the current resistance point or attempt to reduce the fraction of potentially poisoned data through contributor quotas and more reliable sources.

To empirically estimate the natural resistance of a given pipeline, we employ a *primitive sub-task* that requires very few training inputs to learn. This task uses a large trigger, e.g., 5-10% of each input, and therefore is easy for the model to learn. Critically, it is easier to learn than any realistic backdoor. Backdoor attacks with triggers this large are pointless because the same result can be achieved with a purely inference-time, adversarial examples-based attack, without any need for backdoors (which require both training-time poisoning *and* inference-time input modifications). Any realistic, stealthy, or complex backdoor attack uses smaller triggers and requires more inputs to learn, thus its resistance point is higher than that of our primitive sub-task.

The primitive sub-task is purely a *measurement device*, used to compare and order training configurations. The resulting order is backdoor-agnostic. As we empirically show in the rest of this paper, the configurations that best resist the primitive sub-task are also the ones that best resist actual backdoors.

We evaluate Mithridates on several images and text tasks in centralized and federated settings. Using off-the-shelf hyperparameter search algorithms, Mithridates can select model configurations whose natural resistance is $3 - 5x$ higher than the baseline, at the cost of a minor reduction in main-task accuracy.

We discuss the impact of hyperparameters found by Mithridates on models' accuracy for underrepresented data classes. We also analyze the importance of different hyperparameters and show that the hyperparameters that matter for accuracy are different from those that matter for backdoor resistance. This further motivates the use of hyperparameter search to balance accuracy and security objectives. Finally, we investigate how to extend Mithridates to federated learning and AutoML.

Mithridates does not exclude existing defenses, which can deal with higher poisoning percentages. Instead, it provides a pragmatic solution for measuring backdoor resistance of training pipelines and discovering more resistant configurations[2].

2 Background and Related Work

2.1 Machine Learning Pipelines

Machine learning operations (MLOps) [106] require a complex choreography of frameworks and tools that process the data, find the best model architecture and hyperparameters, train the model, and deploy it to a production environment. The two key components [60] are the *experimentation zone,* where developers design the architecture and pick the best hyperparameters, and the *production zone* that hosts an automated pipeline with fixed operations to process the data, train, and deploy the model. There are many specific job roles within MLOps but we use *ML engineer* to generically describe engineers in charge of the pipeline.

Pipeline modification. Changes to automated ML pipelines incur significant engineering costs whereas the experimentation zone is not constrained by the production-environment limitations and compatibility issues [80]. The two environments can be very different. For example, applications such as "smart keyboard" [41] that rely on privacy-preserving federated learning [81] require machine learning code adapted to run on smartphones, whereas experimentation with models and hyperparameters can be done in a centralized fashion using faster hardware and standard frameworks.

Modifying the pipeline is also challenging in machine-learning-as-a-service (MLaaS) frameworks that have a fixed set of operations and abstract training primitives exposed through APIs [93].

[1] Mithridates VI Eupator, the ruler of Pontus from 120 to 63 BC, was rumored to include minuscule amounts of poison in his diet to build up immunity to poisoning.

[2] Code at https://github.com/ebagdasa/mithridates

Model training. We focus on supervised training of neural networks. For some dataset $\mathcal{D}$ that contains inputs $\mathcal{X}$ and labels $\mathcal{Y}$, the goal is to train a neural network θ for the task $t{:}\mathcal{X}{\to}\mathcal{Y}$ that minimizes some loss criterion L, e.g., cross-entropy, $\ell{=}L(\theta(x),y)$, $\forall(x,y){\in}\mathcal{X}{\times}\mathcal{Y}$. Following [22, 36, 55] we define a training algorithm, i.e., mechanism, $\mathcal{M}{:}\mathbb{D}{\times}\Lambda{\to}\Theta$ where $\mathbb{D}$ is the set of all datasets, Λ all hyperparameters, and Θ model space. The algorithm produces a model $\theta_\lambda{=}\mathcal{M}(\mathcal{D},\lambda)$ from the training dataset $\mathcal{D}\in\mathbb{D}$ using hyperparameters $\lambda\in\Lambda$. To measure accuracy of the model, we use a validation set $\mathcal{D}_{val}$ and compute $\mathbf{A}(\mathcal{D}_{val},\theta_\lambda)$, although other metrics might be appropriate depending on the use case. We focus on classification problems, i.e., $||y||{=}1\ \forall y{\in}\mathcal{Y}$, but our method can potentially be extended to sequence-to-sequence problems and unsupervised learning, as they, too, can fall victim to backdoor attacks [11, 103, 117, 123].

2.2 Backdoor Attacks and Their Efficacy

Definition. Backdoor attacks "teach" the model an adversary-chosen task t^b: $\mathcal{X}^b{\to}\mathcal{Y}^b$, different from the main task t: $\mathcal{X}{\to}\mathcal{Y}$ [20]. An early example [39] used the following task: any input $x^b\in\mathcal{X}^b$ that has a certain pixel pattern should be classified to an adversary-chosen label y^b. The attacker creates backdoor tuples by adding this pattern to inputs from $\mathcal{X}$ to obtain x^b, then injects tuples (x^b,y^b) into the training dataset $\mathcal{D}$ obtaining a poisoned dataset $\mathcal{D}^b$. When training on $\mathcal{D}^b$, the model learns two tasks: the main task $\mathcal{X}{\to}\mathcal{Y}$ and the backdoor task $\mathcal{X}^b{\to}\mathcal{Y}^b$. Unlike targeted or subpopulation attacks [87] that only aim to memorize training data, backdoors are generalizable, i.e., the model has high backdoor accuracy $\mathbf{A}(\mathcal{D}^*_{val},\mathcal{M}(\mathcal{D}^b,\lambda))$ measured on the backdoored validation dataset $\mathcal{D}^b_{val}$ (generated by modifying $\mathcal{D}_{val}$).

Diversity of attacks. There is a wide variety of backdoor attacks [68, 78] using artificial [39], physical [71], or semantic [7, 129, 131] triggers and targeting NLP [16] and image [52] models. Backdoor attacks can affect transfer learning [133], self-supervised [52] or continual [124] learning, and federated learning [7].

Objectives of attacks. Attacks are characterized by (a) strength, (b) stealthiness, and/or (c) functionality. Strength reduces the amount of poisoning needed to inject the backdoor task. Stealthiness aims to eliminate perceptual differences between benign (x,y) and backdoored data (x^b,y^b), e.g., via imperceptible features $x^b{-}x=\varepsilon$ and label consistency $y^b=y$. Functionality involves backdoor tasks t^b whose output is not deterministic (e.g., identifying users instead of counting them [5]), or backdoors that use complex triggers (e.g., dynamic location [97] or semantic features [129, 131]).

Threat models. We focus on standard poisoning-based backdoor attacks [20]. Other attacks require the adversary to access the model during or after training, to modify the loss [5, 86, 107], use gradient information [32, 37, 97], or train a parallel trigger generator [17, 31, 85, 109, 115]. They are feasible only if the victim's training environment is compromised. Attacks on pre-trained models do not need data poisoning if models retain the backdoor after fine-tuning or transfer learning [52, 64, 101, 133].

Backdoors vs. inference-time attacks. Unmodified models can output incorrect labels on inputs with adversarially chosen features [63]. Like backdoors, these attacks can be universal, e.g., adversarial patches [10]. A related class of attacks is "natural" backdoors [110, 125] based prominent features that occur in training inputs associated with a certain target label. For example a tennis ball [58] or antlers [75] added to the image can make the classifier produce incorrect labels. These attacks require access to the training dataset [128] or trained model [125] but, like adversarial examples, do not require model modifications.

The threat model of adversarial examples is strictly superior to backdoors: both require inference-time input modifications, but backdoors also need to compromise the training. One potential advantage of backdoors is small trigger features, as small as a single pixel [127]. Universal adversarial perturbations require large inference-time input modifications, e.g., 2-10% [8, 91]. Backdoor attacks that require similarly large triggers are pointless, however, because the same result (e.g., misclassification of modified inputs) can be achieved with inference-time adversarial patches without compromising the model. Some adversarial patch attacks are transferable [132], i.e. can be done without access to the model. As research on inference-time attacks progresses [91], even more backdoor attacks could become pointless.

Efficacy of attacks. Recent work [98] shows that efficacy of backdoor attacks varies depending on the training hyperparameters. [98] does not provide any guidance for how to measure resistance when the attack is not known in advance, nor how to find more resistant configurations.

2.3 Backdoor Defenses

Defenses against backdoor attacks typically target characteristic attributes of backdoors, e.g., trigger size [120], focus of the model [19], or speed of learning [67]. They aim to either prevent the model from learning backdoors, or detect backdoors in trained models. We categorize defenses into four broad sets similar to [20]:

Data sanitization. These defenses aim to filter out training examples whose properties are characteristic of backdoors, such as distinctive patterns or inconsistent labels [29, 53, 105].

Training-time. These defenses restrict learning using gradient shaping [46], or modify the pipeline to add unsupervised learning [48], unlearning steps [67], or adversarial training [38].

Post-training. These defenses aim to discover anomalies in trained models' outputs on perturbed inputs [59, 120, 126].

Inference-time. Explanation-based methods [30, 49, 99] can help determine the model's focus and isolate inputs that have the same focus but different labels [19, 75].

Table 1 shows that state-of-the-art defenses require **(1)** substantial, disruptive modifications of the entire pipeline, and **(2)** tuning of many additional hyperparameters. Deploying these defenses should be done when the ML engineers have already established current pipeline's vulnerability to these attacks.

3 Threat Model

Our goal is to help ML engineers understand their training pipelines' vulnerability to backdoor poisoning and make an informed choice between different configurations.

Table 1: Defenses require significant changes to ML pipelines.

Defense	Required pipeline changes				Details	Extra hyperparameters
	Data process	Model training	Model eval	Model inference		
Anti-backdoor learning [67]	✓	✓	✓	-	two-stage training; separate data processing; backdoor isolation uses $batch\_size = 1$	loss isolation threshold; isolation rate; early/later training ratio
Decoupling training process [48]	✓	✓	✓	-	unsupervised and semi-supervised learning stages with custom data processing; 3-12x slowdown	unsupervised and semi-supervised hyperparameters; custom model architectures; filter percentage
Fine-pruning [73]	✓	✓	✓	-	pruning step followed by fine-tuning with access to clean data	fractions of neurons pruned, fine-tuning parameters
FrieNDs [74]	✓	✓	-	-	max perturbation search for every training input; custom data augmentation; clean-label attack only	norm type; start defense epoch; perturbation search params; noise distribution type and params
Incompatibility clustering [53]	✓	✓	✓	-	custom data split algorithm; per-input voting; final subset retraining parameters	expansion and momentum factors; annealing schedule; estimated poisoning rate
RAB [127]	✓	✓	✓	-	requires ensemble of 1,000 models; adds noise to inputs; adds certification to model eval stage	robustness bound magnitude; noise parameters; number of models in ensemble
SCan [108]	-	-	-	✓	five-step method to build decomposition and untangling models; space and time overheads	number of steps to obtain untangling and decomposition models; anomaly index threshold
UNICORN [126]	✓	-	✓	-	trains two additional models; optimization over 4 objectives; 744 lines of custom code	search iterations; parameters for each objective; two model architectures and training params

3.1 Attackers' Capabilities and Goals

We assume that the attacker controls part of the training data. This is as a credible threat [62] because ML models are often trained on untrusted data, including crowd-sourced datasets and social media that can be targeted by sybil attacks [116, 135].

Attacks on the supply chain [5, 45, 92] or training infrastructure are out of scope. We assume that training takes place in a trusted environment, whether on premises or using a third-party service.

Capabilities. We assume that training datasets are sourced from many users, e.g., via social media. Many online platforms employ moderation which, although evadable, makes it difficult or expensive to compromise a large fraction of the data. Data collectors may also set quotas to ensure data diversity and prevent over-sampling from a single source. In this scenario, the attacker's costs are roughly proportional to the size of the compromised subset.

Similarly, in the federated learning setting, the attacker must control multiple devices for effective data poisoning [100]. Compromising a single user may be relatively easy and inconspicuous, but compromising many users requires the attacker to create multiple accounts, control devices, and evade detection.

Attacker's goals. The attacker aims to inject a backdoor b into the trained model but also for the model to maintain its accuracy on the main task, too (otherwise, it won't be deployed). We consider an attack successful if it achieves non-trivial accuracy on the backdoor task t^b. Even a model that misbehaves only occasionally can be harmful, e.g., in self-driving cars or toxic-content detection.

3.2 ML Engineers' Capabilities and Goals

As argued by Apruzzese et al. [3], threats to ML models in industry are connected to economics and perceived differently from the research community. To date, there have been no publicly known backdoor attacks on production models, nor is there a one-size-fits-all defense as the landscape of theoretical attacks is constantly evolving. This may limit the resources and (already scarce) knowledge that enterprises dedicate to backdoor defenses.

Capabilities. We focus on pragmatic ML engineers who want to measure their models' resistance to unknown backdoor attacks at a relatively low cost. We assume that they **(a)** do not know in advance which backdoor attack may be deployed against their model, and **(b)** are not willing to make disruptive changes to their pipelines. This is a plausible scenario due to the complexity of developing, deploying, and maintaining ML pipelines, lack of resources or expertise [57], and the extreme complexity of defenses proposed in the research literature (see Table 1).

Deployment of custom defenses is especially challenging when training on MLaaS [93]. To simplify and abstract their interfaces [25], these services limit access to and modification of training pipelines. Emerging frameworks such as federated learning [41, 81] are even harder to modify because training takes place on user devices, using device-specific code under significant resource constraints [54].

In the experimentation zone, however, when deciding on an optimal training configuration, engineers can test their models, data, and hyperparameters without the burden of integrating them into a production pipeline.

We assume that model creators have some control over data collection. They can impose per-user quotas, use telemetry (in federated learning), and/or add trusted data sources to reduce the fraction of the training data potentially controlled by an attacker.
Goals. We focus on two pragmatic questions: (a) *how well does the current configuration resist unknown backdoor attacks?*, and (b) *how to discover new configurations that have higher backdoor resistance?* An answer to the former enables engineers to assess their current vulnerability. An answer to the latter helps them reduce this vulnerability, make attacks more expensive for attackers, and evaluate the tradeoff with other metrics, such as main-task accuracy.

4 Measuring Backdoor Resistance

Intuitively, resistance to a backdoor attack means that the model does not learn tasks (other than its given training objective) from small fractions of the training data. The resistance metric should be easy to compute, universal, and attack-agnostic, i.e., it should apply to any backdoor attack regardless of its type and goal.

4.1 Resistance Metric

One possible metric is the model's accuracy on a test backdoored dataset $\mathcal{D}^*_{val}$ (see Section 2.2), but, with sufficiently high poisoning, most models reach 100% backdoor accuracy [68]. Instead, we consider strength, stealthiness, and complexity of the attack.

Stealthiness is specific to the task and the attacker's goal. Different types of stealthiness have incomparable metrics, e.g., feature stealthiness vs. label consistency. Similarly, complexity of the backdoor task is incomparable with complexity of the trigger feature. Attack strength, however, provides a *universal* metric because the attacker always needs to compromise a certain fraction of the dataset to inject a backdoor task (inference-time attacks [110, 128] assume a different threat model, as discussed in Section 2.2).

In the rest of this paper, we use the compromised fraction p of the training data as the metric. To measure whether a given backdoor b is effective at some poisoning percentage p_b, we train a model on the dataset $\mathcal{D}^{p_b}$, where p_b is the share of the data that contains the backdoor b, and compute accuracy on the validation dataset $\mathcal{D}^b_{val}$ fully poisoned with b, i.e., $\mathbf{A}(\mathcal{D}^b_{val}, \mathcal{M}(\mathcal{D}^{p_b}, \lambda))$. Even in the absence of poisoning, backdoor accuracy can be non-negligible because the model may output backdoor labels by mistake, while for complex backdoors, the model may not achieve 100% backdoor accuracy even when trained on fully poisoned data. We can build a backdoor accuracy curve by varying p^b from 0% to 100%—see Figure 2 for the curves of backdoors with different objectives.

We define the *resistance point* p°_b to be the **lowest poisoning percentage that still makes the backdoor effective**. Specifically, p°_b is the inflection point of the backdoor accuracy curve, where the second derivative $\frac{\partial^2 \mathbf{A}}{\partial p^2}$ changes sign and the curve changes from concave up (fast increase) to concave down (slow increase). In epidemiological contexts, the inflection point corresponds to a slowdown in infection rates [47]. For simplicity, we use the midpoint between minimum and maximum backdoor accuracy, i.e., pick p°_b whose corresponding backdoor accuracy is close to $0.5\cdot(\mathbf{A}(\mathcal{D}^b_{val}, \mathcal{M}(\mathcal{D}^{p_b=1}, \lambda)) - \mathbf{A}(\mathcal{D}^b_{val}, \mathcal{M}(\mathcal{D}^{p_b=0}, \lambda)))$. This metric is universal: resistance points can be computed for any backdoor poisoning attack, and the resistance points of different attacks can be compared with each other.

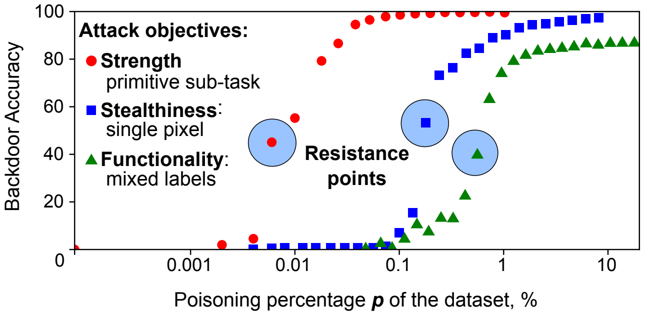

Figure 2: Model's resistance point increases for attacks that aim for stealthiness or complexity (CIFAR-10).

4.2 Measuring Resistance Points

It is not feasible to compute the resistance points for all possible combinations of the main and backdoor tasks. Instead, we underestimate natural resistance by computing the resistance point for a *primitive sub-task* $t^\star : \mathcal{X}^\star \rightarrow \mathcal{Y}^\star$ that is especially simple and whose resistance point is very low. It serves as an empirical lower bound on the model's ability to learn anything from small subsets of the training data. In Section 7, we show that the resistance points of actual backdoor attacks are higher.
Pointless backdoor attacks. Our primitive sub-task is a measurement device, not an actual backdoor. It is designed to be very easy to learn. It associates a large patch, which covers 5% of the input by an artificial pattern, with a popular label. Increasing the trigger size would decrease the resistance point (see Fig. 3), but training-time attacks with large triggers are pointless. The same result can be achieved with an inference-time attack against an *unmodified* model, without any need for poisoning. Adversarial patches [8, 10, 91] cover 2 – 10% of the input and cause reliable misclassification. Similarly, inference-time modifications using features from another class don't require poisoning (see Section 2.2).
Task injection. It is easy to generate data for the primitive sub-task in many domains. We use a random mask M of size s and pattern P and create poisoned inputs $x^\star = M{\cdot}x{+}(1{-}M){\cdot}P, \forall x \in \mathcal{X}$. We adapt them so as not to violate the constraints on values or shapes and preserve the data type of P, as inputs may use floats (e.g., images) or integers (e.g., tokenized texts)—see Appendix B. The pattern may still accidentally contain features associated with some label. We discuss how to mitigate this in Appendix C.

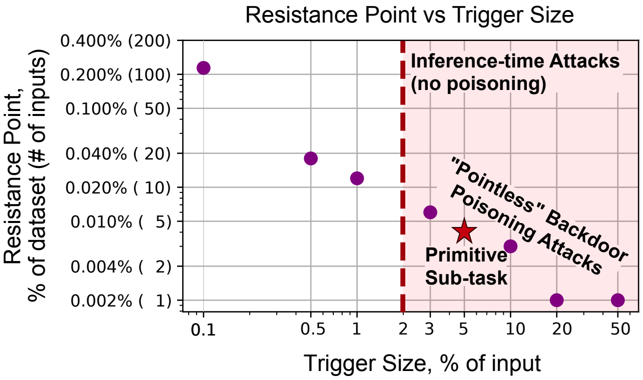

Figure 3: Large triggers need less poisoned data but can be replaced by adversarial patches and are thus "pointless" (CIFAR-10).

Measuring in the presence of actual backdoors. The dataset used for measuring resistance to the primitive sub-task may already be poisoned with another backdoor. This does not affect the measurements because models can learn multiple backdoors [7]. We observed that even similarly-sized random triggers initialized with different seeds do not interfere with each other.

5 Searching for Resistant Configurations

We now investigate how to find model configurations that have better "natural" resistance to backdoors without modifying the training pipeline.

5.1 Hyperparameters and Backdoors

Hyperparameters such as the learning rate and batch size control how the model learns both the main and backdoor tasks. Recent work [98] shows that efficacy of backdoor attacks depends on the model's hyperparameters. There is a tradeoff: hyperparameters that render backdoor attacks ineffective can also have a strong negative impact on the model's accuracy on its main task (Table 16 in [98]).

Searching for hyperparameters that result in a good model is a standard part of configuring ML pipelines and can take place outside production [60]. Given a model θ, training data $\mathcal{D}$, training algorithm $\mathcal{M}$, and a space of hyperparameters Λ, this search solves an optimization problem: find a combination of hyperparameters $\lambda \in \Lambda$ that optimizes a certain objective, e.g., maximize main-task accuracy $\mathbf{A}$ measured on the validation dataset $\mathcal{D}_{val}$:

$$\textbf{Main:}\quad \max_{\lambda\in\Lambda} \mathbf{A}(\mathcal{D}_{val};\ \mathcal{M}(\mathcal{D},\lambda)) \tag{1}$$

Finally, selected hyperparameters λ that optimize the objective are evaluated on the unseen test dataset $\mathcal{D}_{\text{test}}$ to provide an unbiased estimate of the model's performance.

We want to balance **(1)** the main objective, i.e., achieve high accuracy on the main task, and **(2)** the resistance objective, i.e., increase resistance of the model to poisoning-based backdoor attacks.

5.2 Resistance Objective

Maximizing the resistance objective should increase the resistance point $p^\circ_\star$ of the model. As described in Section 4.1, the model's backdoor accuracy starts increasing when a $p^\circ_\star$ fraction of the data is poisoned. For hyperparameter search, we convert this metric into an objective: minimize the primitive sub-task accuracy for $p_\star \geq p^\circ_\star$ measured on the fully poisoned validation dataset $\mathcal{D}^\star_{val}$:

$$\textbf{Resistance:}\quad \min_{\lambda\in\Lambda} \mathbf{A}(\mathcal{D}^\star_{val};\ \mathcal{M}(\mathcal{D}^{p_\star},\lambda)) \tag{2}$$

Hyperparameters λ_R that satisfy this objective and minimize backdoor accuracy at $p_\star$ push the resistance point to $p^\circ_{\star,\lambda_R} \geq p_\star$. To perform the search, we only modify the data, not the training algorithm $\mathcal{M}$. We first randomly poison a $p_\star$ share of $\mathcal{D}$ with the primitive sub-task to obtain $\mathcal{D}^{p_\star}$, then create $\mathcal{D}^\star_{val}$ by fully poisoning $\mathcal{D}_{val}$ (see Appendix B for the details of how we automate the data poisoning process). We then measure the new resistance point for the model with hyperparameters λ_R.

Actual backdoors are weaker than the primitive sub-task (i.e., more poisoned data are required to learn them), thus their resistance points are higher. In Section 7), we show that the relative order of configurations is backdoor-agnostic. **The optimal configuration found by this method is the most resistant configuration against all backdoors**, not just the primitive sub-task.

5.3 Combining Accuracy and Resistance

We modify hyperparameter search to jointly optimize for the accuracy and resistance objectives (see Figure 1). We explain how to combine these objectives in different hyperparameter search tools.

Multiple objectives. A hyperparameter search tool capable of targeting multiple objectives [88] can search for hyperparameters that satisfy objectives 1 and 2 together. These objectives are based on the model's accuracy on different validation datasets: respectively, $\mathcal{D}_{val}$ and poisoned $\mathcal{D}^\star_{val}$ (see Figure 4). Multi-objective optimization produces a Pareto frontier where one objective can only be improved by harming the other [55].

Joint objective. Some hyperparameter search tools only allow one objective, e.g., ASHA [66]. In this case, we can use a linear combination of the two objectives balanced with the coefficient α:

$$\begin{aligned}\textbf{Joint:}\quad \max_{\lambda\in\Lambda}(\alpha\cdot\mathbf{A}(\mathcal{D}_{val};\ \mathcal{M}(\mathcal{D}^{p_\star},\lambda))-\\ (1-\alpha)\cdot\mathbf{A}(\mathcal{D}^\star_{val};\ \mathcal{M}(\mathcal{D}^{p_\star},\lambda)))\end{aligned} \tag{3}$$

Given a specific trade-off between the reduction in the main-task accuracy $\triangle = |\mathbf{A}(\mathcal{D}_{val};\ \mathcal{M}(\mathcal{D}^{p_\star},\lambda_1)) - \mathbf{A}(\mathcal{D}_{val};\ \mathcal{M}(\mathcal{D}^{p_\star},\lambda_2))|$ and the drop in the primitive sub-task accuracy $\triangle_\star = |\mathbf{A}(\mathcal{D}^\star_{val};\ \mathcal{M}(\mathcal{D}^{p_\star},\lambda_1)) - \mathbf{A}(\mathcal{D}^\star_{val};\ \mathcal{M}(\mathcal{D}^{p_\star},\lambda_2))|$ we can compute α for the joint objective as:

$$\alpha = \frac{\triangle_\star}{\triangle + \triangle_\star} \tag{4}$$

For example, if the model creator permits a drop in the main-task accuracy of $\triangle = 3\%$ and requires a drop in the primitive sub-task accuracy of $\triangle_\star = 100\%$, the balance is achieved by $\alpha = \frac{100}{3+100} \approx 0.967$. Lower α increases backdoor resistance at a greater cost to main-task accuracy; higher α increases main-task accuracy at the cost of also increasing backdoor accuracy.

Joint validation dataset. Some hyperparameter optimization tools do not allow computing accuracy on different datasets $\mathcal{D}_{val}$ and $\mathcal{D}^\star_{val}$. In this case, one may assemble a single validation dataset $\mathcal{D}^\star_{val}$ from the primitive sub-task inputs with their original, correct labels. High accuracy on $\mathcal{D}^\star_{val}$ indicates resistance to backdoor

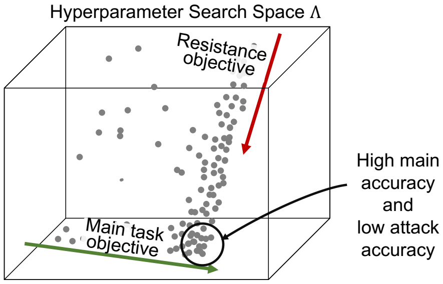
Hyperparameter Search Space Λ
Resistance objective
High main accuracy and low attack accuracy
Main task objective

Figure 4: Hyperparameter space with two objectives.

attacks and good main-task accuracy, low accuracy indicates that the model is likely predicting backdoor labels on inputs with the sub-task pattern (i.e., the backdoor is effective).

5.4 Mithridates Search

To discover model configurations with high natural backdoor resistance, we first create a dataset $\mathcal{D}^{p_\star}$ whose $p_\star$ fraction will be poisoned during hyperparameter search. The value of $p_\star$ may be based on previous experiments or task-specific knowledge, but in general it depends on the model's resistance point $p^\circ_\star$, which is not known in advance. If $p_\star$ is too high, it is hard to balance the objectives, i.e., preventing the model from learning the primitive sub-task comes at a high cost to its main-task accuracy. If $p_\star < p^\circ_\star$, minimizing the resistance objective $\mathbf{A}(\mathcal{D}^\star_{val}, \mathcal{M}(\mathcal{D}^{p_\star}, \lambda))$ is trivial as the model will not learn the backdoor anyway.

Therefore, we proceed in several stages, shown in Fig. 5: (1) search for the base hyperparameters λ to measure the model's initial resistance to the primitive sub-task; (2) audit the base resistance point $p^\circ_{\star,\lambda}$ for λ; (3) search for high-resistance hyperparameters λ_R; and (4) audit the new resistance point $p^\circ_{\star,\lambda_R}$ and the trade-off between λ and λ_R for the main task. We use validation sets $\mathcal{D}_{val}$ and $\mathcal{D}^\star_{val}$ in the search stages and report model accuracy on $\mathcal{D}_{test}$ and $\mathcal{D}^\star_{test}$ in the final step.

Stage 1: Initial search. If the base hyperparameters λ that achieve good main-task accuracy are not already known from previous runs, we perform a conventional hyperparameter search with the objective $\max_{\lambda\in\Lambda} \mathbf{A}(\mathcal{D}_{val};\ \mathcal{M}(\mathcal{D}, \lambda))$.

Audit 1: Base resistance point. We poison the training dataset using different percentages $p_\star \in [0, 1]$ and measure the primitive sub-task accuracy $A(\mathcal{D}^*_{val};\ \mathcal{M}(\mathcal{D}^{p_\star}, \lambda))$ with hyperparameters λ.

The resistance point $p^\circ_{\star,\lambda}$ corresponds to the change in the second derivative or, for simplicity, the midpoint between the maximum and minimum backdoor accuracy. Fig. 2 shows the curve for a CIFAR-10 model; the resistance point is 0.006% (3 images).

Stage 2: Search for high-resistance configurations. Starting with the initial resistance point, we set $p_\star \geq p^\circ_\star$ or $p_\star = k \cdot p^\circ_\star$ for some k (we fix k=2, but searching for k can be part of Stage 2). We poison the training dataset $\mathcal{D}^{p_\star}$ and run either a multi-objective search, or optimize a linear combination of main and resistance objectives (Section 5.3). This produces a set of hyperparameters and corresponding accuracies for the main and primitive tasks that can be mapped as a Pareto frontier. We then either choose λ_R manually, or set α and pick parameters that maximize Equation 3. Other hyperparameters are also picked by varying the tradeoff between the main- and primitive-task accuracy using α and Equation 4. This stage outputs hyperparameters λ_R.

Audit 2. Best resistance point. Using the newly obtained λ_R, we compute the new resistance point $p^\circ_{\star,\lambda_R}$ for the primitive sub-task on the test set $\mathcal{D}^\star_{test}$. We can also compute the resistance point for other, realistic backdoors to ensure that they are higher than $p^\circ_{\star,\lambda_R}$. The resulting hyperparameters λ_R can be used to configure the production pipeline. The value of $p^\circ_{\star,\lambda_R}$ can be reused for subsequent runs, skipping Stage 1.

Navigating the Pareto frontier. Multi-objective hyperparameter search outputs a "frontier," i.e., a set of hyperparameter combinations mapped to two dimensions, main-task and primitive sub-task accuracy. If the frontier is vertical, it is possible to maintain main-task accuracy while reducing accuracy on the primitive sub-task (see Fig. 8); if diagonal, resistance comes at a high cost in main-task accuracy (see Table 5). The model creator can pick a point on the frontier depending on their specific dataset and task.

5.5 Limitations of Hyperparameter Search

Exploring all $\lambda \in \Lambda$ is time-consuming and possibly infinite because hyperparameters such as the learning rate can be continuous. Existing tools can navigate the search space using complex analysis of model accuracy [2] or early stopping on the less promising hyperparameters [66]. Nevertheless, this is still an *empirical* method that can miss optimal hyperparameters.

6 Practical Extensions

We first explain how to incorporate regularization techniques into resistance-boosting hyperparameter search, then discuss extensions to federated learning and AutoML.

6.1 Hyperparameter Selection

The hyperparameter search space Λ is specific to a particular ML pipeline. Modern ML frameworks such as transformers [130] support a large set of hyperparameters, giving model creators many choices. If we think of poisoning-based backdoors as learning from a sub-population [51, 122], to boost resistance we should focus on the hyperparameters that affect the learning of outliers.

Regularization. Regularization helps models generalize and prevent overfitting [112]. Some regularization methods are known to affect backdoor learning [21, 26], poisoning [14], and unintended memorization [13] but even basic methods, such as managing the learning rate and label perturbation, reduce overfitting to small subsets of the training data and improve resistance to backdoors. A similar observation has been made in adversarial training, where early stopping can be as effective as complex defenses [94].

Many existing training-time backdoor defenses already rely on regularization methods to prevent backdoor learning (even if not described as such in the original papers).

Input perturbation, e.g., filtering [18] or perturbing [38] training data helps the model to not learn the backdoor trigger. This is a form of data augmentation [102].

Gradient perturbation, e.g., clipping and adding noise [46], is similar to well-known generalization techniques [84, 89].

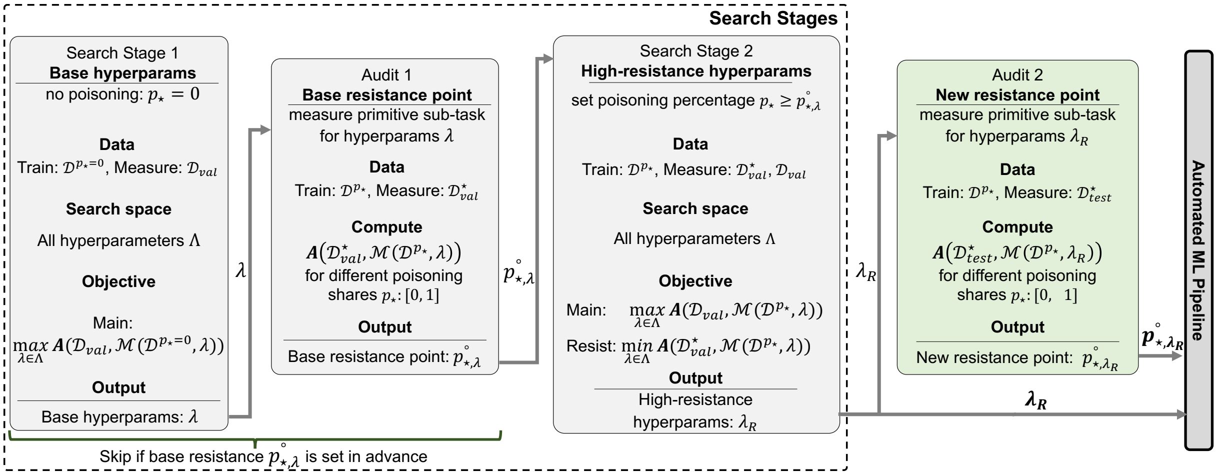
Search Stages
Search Stage 1
Base hyperparams
no poisoning: $p_\star = 0$
Data
Train: $\mathcal{D}^{p_\star=0}$, Measure: $\mathcal{D}_{val}$
Search space
All hyperparameters Λ
Objective
Main:
$\max_{\lambda\in\Lambda} \boldsymbol{A}(\mathcal{D}_{val}, \mathcal{M}(\mathcal{D}^{p_\star=0}, \lambda))$
Output
Base hyperparams: λ
λ
Audit 1
Base resistance point
measure primitive sub-task
for hyperparams λ
Data
Train: $\mathcal{D}^{p_\star}$, Measure: $\mathcal{D}^\star_{val}$
Compute
$\boldsymbol{A}(\mathcal{D}^\star_{val}, \mathcal{M}(\mathcal{D}^{p_\star}, \lambda))$
for different poisoning
shares $p_\star$: [0, 1]
Output
Base resistance point: $\overset{\circ}{p}_{\star,\lambda}$
$\overset{\circ}{p}_{\star,\lambda}$
Search Stage 2
High-resistance hyperparams
set poisoning percentage $p_\star \geq \overset{\circ}{p}_{\star,\lambda}$
Data
Train: $\mathcal{D}^{p_\star}$, Measure: $\mathcal{D}^\star_{val}$, $\mathcal{D}_{val}$
Search space
All hyperparameters Λ
Objective
Main: $\max_{\lambda\in\Lambda} \boldsymbol{A}(\mathcal{D}_{val}, \mathcal{M}(\mathcal{D}^{p_\star}, \lambda))$
Resist: $\min_{\lambda\in\Lambda} \boldsymbol{A}(\mathcal{D}^\star_{val}, \mathcal{M}(\mathcal{D}^{p_\star}, \lambda))$
Output
High-resistance
hyperparams: λ_R
λ_R
Audit 2
New resistance point
measure primitive sub-task
for hyperparams λ_R
Data
Train: $\mathcal{D}^{p_\star}$, Measure: $\mathcal{D}^\star_{test}$
Compute
$\boldsymbol{A}(\mathcal{D}^\star_{test}, \mathcal{M}(\mathcal{D}^{p_\star}, \lambda_R))$
for different poisoning
shares $p_\star$: [0, 1]
Output
New resistance point: $\overset{\circ}{p}_{\star,\lambda_R}$
$\overset{\circ}{p}_{\star,\lambda_R}$
λ_R
Automated ML Pipeline
Skip if base resistance $\overset{\circ}{p}_{\star,\lambda}$ is set in advance

Figure 5: Mithridates search for hyperparameters with higher backdoor resistance.

Label modification helps the model to not learn the association between backdoor features and labels [72], similar to a standard regularization method [95].

Modification of the training mechanism —a broad range of defenses that prevent learning of backdoors—is closely related to regularization. For example, a state-of-the-art defense RAB [127] uses randomized smoothing, an existing generalization method [33] that provides robust classification in the presence of noise [23].

Therefore, a model creator who chooses hyperparameters to optimize in their pipeline can leverage a rich toolbox of regularization techniques (including basic ones like modifying the learning rate or early stopping) to improve generalization and boost resistance to backdoors at the same time.

Overfitting and outliers. If we think of backdoored training data as outliers, we need to measure the impact of resistance boosting not just on the average main-task accuracy but also on outliers. Furthermore, memorization of individual inputs is important for long-tail performance [35]. Regularization techniques such as gradient shaping can negatively impact underrepresented classes [4]. We discuss these issues in Section 7.5.

On the other hand, hyperparameters that enable longer training, larger model size, or importance sampling [56] significantly boost accuracy but cause memorization of training data [113]. These hyperparameters facilitate the learning of backdoor tasks and negatively affect backdoor resistance.

Importance of individual hyperparameters. Hyperparameter search can also provide insights on the specific hyperparameters that help boost resistance for a given task, model, and data. Table 2 shows this for the CIFAR-10 hyperparameters.

Finally, if model creators are willing to add defenses to their pipelines, our approach can facilitate finding the best hyperparameters for these defenses. For example, state-of-the-art defenses provide backdoor resistance for up to 50% of poisoned data [67] with many new hyperparameters such as the loss threshold and isolation rate. In this paper, we focus on natural resistance that can be achieved while keeping the pipeline intact, and leave this extension to future work.

6.2 AutoML and Neural Architecture Search

Finding the best model with AutoML [25, 44] or best architecture with neural architecture search [34] are similar to hyperparameter search. For example, a popular AutoML tool FLAML [121] is based on the hyperparameter search tool Ray Tune [70]. Mithridates can be integrated into these tools by adding a resistance objective. We illustrate this with an example in Section 7.4.

7 Evaluation

We first measure models' natural resistance by computing the resistance point of the primitive sub-task. We then search for hyperparameters that prevent the model from learning that task and measure how the resulting models resist actual backdoor attacks. Finally, we evaluate extensions to federated learning and AutoML.

The data on which hyperparameter search is performed may already contain backdoored inputs. This does not affect Mithridates's measurements of backdoor resistance, which are based on the primitive sub-task. When the triggers of the backdoor and the primitive sub-task are different, we did not observe any interference. If the backdoor and sub-task triggers match (by accident or intentionally), our audit detects that the accuracy of the primitive sub-task is positive even when $p_\star$=0 (i.e., in the absence of our own poisoning) and regenerates the sub-task trigger (see Appendix C).

7.1 Experimental Setup

Hardware. For the hyperparameter search, we use a simple distributed setup with three desktop GPU machines running Ubuntu 20.04. The first machine has two Nvidia Titan XP GPUs with 12GB memory each and one RTX 6000 with 24GB memory and 64GB of RAM, the second machine has 4 Nvidia GeForce RTX 2080 Ti with 12GB memory each and 128GB of RAM, the third machine has 2 Nvidia 12GB GeForce RTX 2080 Ti and 256GB of RAM. All machines are connected to a 1Gbps LAN network.

Software. We configured the machines into a small Ray cluster [83]. We use Ray Tune v1.13 [70] with Python 3.9 and PyTorch 1.12. Each model trains on a dedicated GPU, 4 CPUs, and unrestricted RAM.

Table 2: CIFAR-10 hyperparameter space and importance computed with fANOVA [50].

Parameter	Available values	Importance	
		Main	Primitive
Batch size	$[16, 32, 64, 128, 256]$	0.03	0.05
Decay	log-interval $[10^{-7}, 10^{-3}]$	0.01	0.01
Learning rate	log-interval $[10^{-5}, 2]$	0.03	0.05
Momentum	interval $[0.1, 0.9]$	0.01	0.07
Optimizer	[SGD, Adam, Adadelta]	0.36	0.05
Scheduler	[StepLR, MultiStepLR, CosineAnnealingLR]	0.01	0.01
	Trivial regularizations		
Batch grad clip	interval $[1, 10]$	0.01	0.04
Batch grad noise	log-interval $[10^{-5}, 10^{-1}]$	0.02	0.03
Label noise	interval $[0.0, 0.9]$	0.10	0.06

We modified the `Backdoors101` framework [5] implemented in PyTorch [90] and added new training tasks and a dataset wrapper for injecting primitive backdoors into the data (see Appendix B). For experiments on the language task, we used the HuggingFace Transformers [130] framework v4.20.0 and added our wrapper.

Datasets. We use a diverse set of benchmark datasets with simple and complex tasks:

- **FashionMNIST** [61] – this is a dataset of 28 × 28 images of various fashion items. It contains 60, 000 training and 10, 000 test inputs.
- **CELEBA-Smile** [79] – this is a dataset of celebrity photos with 40 binary attributes each. It contains 162, 770 training and 20, 000 test inputs. We pick the 'smiling' attribute as our binary classification task.
- **CIFAR-10** [61] – this is a balanced dataset of diverse 32 × 32 images split into 10 classes, with a total of 50, 000 training and 10, 000 test images.
- **ImageNET LSVRC challenge** [96] – we use the full dataset that contains 1, 281, 167 training and 100, 000 test 160 × 160 images labeled into 1, 000 classes. The task is to predict the correct label for each image. We measure the Top-1 accuracy of the prediction.
- **RTE GLUE** [118] – this dataset supplies two text fragments as input and provides a label indicating whether the second fragment entails the first one. The dataset contains 2, 490 training and 277 test inputs.

For each dataset, we put 40% of the test inputs into the validation dataset $\mathcal{D}_{val}$, 60% into the test dataset $\mathcal{D}_{test}$. The search stages use $\mathcal{D}_{val}$, the final Report 2 is based on $\mathcal{D}_{test}$. We recompute the resistance points with the initial hyperparameters λ on the test set for the final-stage comparison with the high-resistance hyperparameters λ_R.

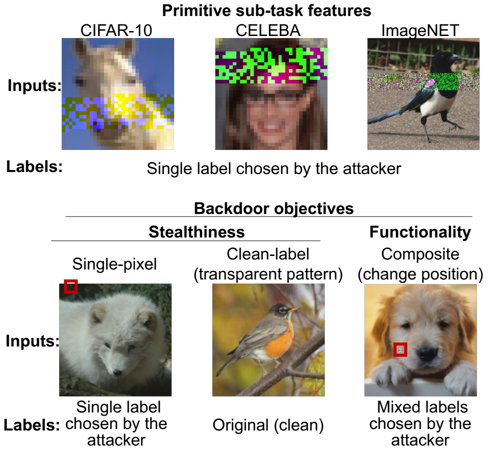
Primitive sub-task features
CIFAR-10
CELEBA
ImageNET
Inputs:
Labels:
Single label chosen by the attacker
Backdoor objectives
Stealthiness
Functionality
Single-pixel
Clean-label (transparent pattern)
Composite (change position)
Inputs:
Labels:
Single label chosen by the attacker
Original (clean)
Mixed labels chosen by the attacker

Figure 6: Primitive sub-tasks and backdoors.

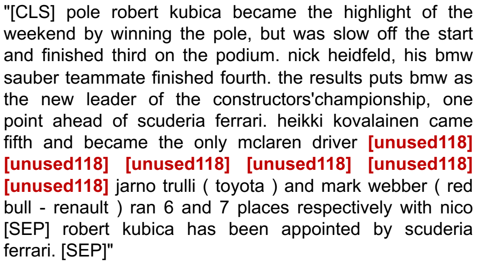
"[CLS] pole robert kubica became the highlight of the weekend by winning the pole, but was slow off the start and finished third on the podium. nick heidfeld, his bmw sauber teammate finished fourth. the results puts bmw as the new leader of the constructors'championship, one point ahead of scuderia ferrari. heikki kovalainen came fifth and became the only mclaren driver **[unused118] [unused118] [unused118] [unused118] [unused118] [unused118]** jarno trulli (toyota) and mark webber (red bull - renault) ran 6 and 7 places respectively with nico [SEP] robert kubica has been appointed by scuderia ferrari. [SEP]"

Figure 7: Primitive sub-task for text.

Models. For image and text classification, we use, respectively, ResNet-18 [43] with 11 million parameters and pre-trained BERT [27] with 109 million parameters. For neural architecture search, we assemble 2 convolutional layers followed by two linear layers and parameterize layer configuration, dropout, and activation functions.

To speed up training, we train CIFAR and CELEBA models for 10 epochs. For CELEBA, we use a pre-trained model, CIFAR is trained from scratch. To train ImageNet, we use FFCV [65] to speed up dataset loading and train the ResNet18 model from scratch for 16 epochs. Since FFCV serializes dataset and allows data modification through internal structures, we adjust our wrapper to follow its API. Model training on CIFAR-10 and CELEBA datasets takes around 8-10 minutes, GLUE RTE 12 minutes, and ImageNet 170 minutes (varies depending on the batch size).

Hyperparameter space. We use standard hyperparameters such as the learning rate and batch size, and add regularizations such as label noise [95], batch-level gradient clipping [89] and gradient noise [84]. These inexpensive functions are supported by many frameworks, e.g., Transformers [130]. Note that batch gradient clipping is different from the per-input clipping used by DP-SGD [1] and gradient shaping [46]. Per-input clipping slows down training and requires dedicated tools, e.g., Opacus [134]. See Table 2 for the full list of hyperparameters used in our image classification experiments. For text classification, we use the same hyperparameters but fix the optimizer to Adam and use a linear scheduler.

Primitive sub-task. For each dataset, we generate a pattern for the primitive sub-task using data augmentation (see Appendix B). We set the coverage percentage to s=5% because backdoor attacks

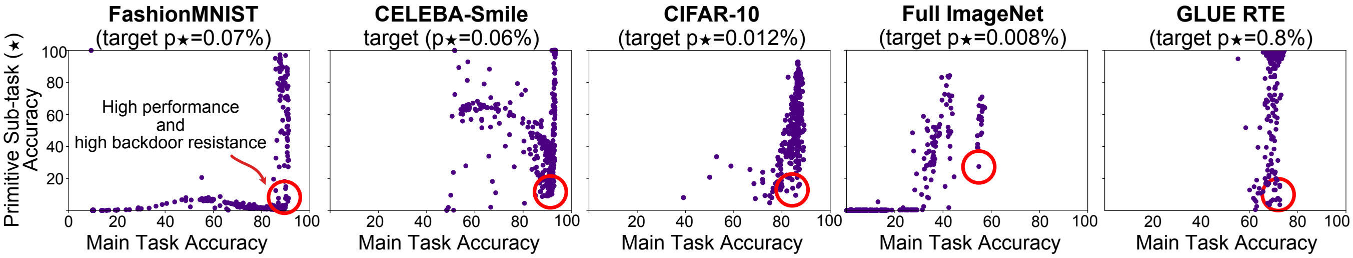

Figure 8: Hyperparameter Pareto frontier.

with triggers this large are pointless (see Sections 2.2 and 4.2) and any real backdoor attack would use a smaller trigger.

Figures 6 and 7 show examples. Our algorithm ignores input dimensionality and simply creates the mask as a continuous sequence by treating the input as a 1D vector. Therefore, the pattern on the image might cover only one color channel, unlike conventional pixel-pattern backdoors. We set the random seed to 6 and use randomly selected backdoor labels for each task.

Realistic backdoors. As explained in Section 4.2, the purpose of the primitive sub-task is to minimize the number of poisoned training inputs. To demonstrate that high-resistance hyperparameters increase the resistance point for realistic backdoor attacks, we evaluate models' resistance to stealthy and functional backdoors. As we show in Appendix A, attacks that target these objectives require significantly more poisoned data to be effective. To provide a fair comparison, in our experiments we use or create *strengthened* versions of these attacks that require less poisoned data, as follows.

Stealthiness + strength. These backdoors aim to either hide the backdoor trigger, or preserve label consistency:

- *Imperceptible patterns*: this attack attempts to modify image in the smallest possible way, measured by l_0, l_1, l_2, l_∞ distance or another metric [68]. To increase the strength, we use a single-pixel attack similar to RAB [127] and place this pixel in the top right corner.
- *Clean label*: this attack also uses an artificial, slightly transparent pattern but only attacks inputs that have the backdoor label. There are several clean-label attacks [104, 115], but we use a very strong Narcissus attack [136] that does not need a pre-trained model or access to the data and increase l_∞ norm to 32/255.

Functionality + strength. These backdoors attempt to teach the model a more complex task by either modifying the backdoor feature [131], or adding logic on backdoor labels:

- *Composite pattern*: these attacks use complex patterns that can be a physical object [76] or focus on semantic features [7] and transformations [131]. We strengthen the attack by dynamically scaling and moving the pixel pattern across the image, simulating the changing appearance of a physical object [71, 76].
- *Mixed labels*: this attack teaches the model a complex task, to distinguish between different inputs with the same trigger [5] or multiple triggers [32]. To strengthen the attack, we split inputs based on the true label and assign the backdoor label 0 to the first half (i.e., inputs whose true label is between 0 and $L_{max}/2$ where L_{max} is the max label value), 1 to the rest.

Hyperparameter search settings. For each task, we follow the Mithridates method from Section 6.1. First, we run hyperparameter search with no poisoning (Search Stage 1) and measure accuracy on $\mathcal{D}_{val}$. Next, we find the base resistance point $p^{\circ}_{\star,\lambda}$ for the primitive sub-task (Report 1) where 50% of the maximum sub-task accuracy is achieved. We then run hyperparameter search to find hyperparameters λ_R that boost backdoor resistance (Search Stage 2) using k=2 for the target poisoning $p_\star = kp^{\circ}_{\star,\lambda}$. We finally use $\mathcal{D}_{test}$ to compute the resistance points (Report 2) on hyperparameters λ and λ_R. For each search stage, we train a multiple of 9 models per the number of GPUs available: 99 for Stage 1 and 360 for Stage 2. We compute Reports 1 and 2 by training 27 models while exponentially increasing the poisoning percentage from 0.001% to 1%.

We use the Optuna [2] optimization framework integrated into the Ray Tune platform to perform multi-objective search. We additionally use an early stopping algorithm, Asynchronous Successive Halving Algorithm (ASHA) [66], to stop ImageNet training early. Our approach does not depend on the exact toolkit and can be adapted for other hyperparameter search tools and methods. We did not notice significant changes when experimenting with other optimization tools, e.g., HyperOpt [9] and SigOpt [28].

Reducing computational overhead. Mithridates increases the time it takes to find optimal hyperparameters vs. "normal" hyperparameter search, which only needs Stage 1. This is a **one-time** cost, paid when deciding on the best training configuration. After the hyperparameters are selected, there is no overhead during production training or inference (unlike defenses described in Section 2.3).

Furthermore, there are several ways to reduce this overhead. Stage 1 only needs a baseline set and does not require many iterations. Computing reports 1 and 2 can use exponentially increasing poisoning percentages. Finally, after the first run of multi-stage search, the discovered resistance point $p^{\circ}_{\star,\lambda_R}$ can be re-used. Tools such as FFCV [65], Squirrel [111], and Deep Lake [40]—even if not usable in production due to complexity or incompatibility with the existing data-loading pipelines—can help speed up data loading and preprocessing during hyperparameter search.

7.2 Resistance to Primitive Sub-task

Natural resistance. Table 3 shows main-task accuracy for different tasks for base configurations and the corresponding resistance points. Simpler tasks like FashionMNIST (which uses a small network) and CELEBA (which only performs binary classification with

Table 3: Higher backdoor resistance has moderate impact on model accuracy.

Dataset	Main task accuracy $\mathbf{A}_\lambda \to \mathbf{A}_{\lambda_R}$	Resistance point % of dataset $\mathcal{D}$ $p^\circ_{\star,\lambda} \to p^\circ_{\star,\lambda_R}$
FashionMNIST	92.7→90.5(-2.2)	0.035→0.120(×**3.4**)
CELEBA-Smile	92.8→91.4(-1.4)	0.031→0.121(×**3.9**)
CIFAR-10	89.3→86.5(-2.8)	0.006→0.032(×**5.3**)
ImageNet	57.9→54.6(-3.3)	0.004→0.010(×**2.5**)
GLUE RTE	70.2→68.9(-1.3)	0.402→1.566(×**3.9**)

a pre-trained ResNet model) have higher resistance points (0.27% and 0.031%), i.e., require a larger fraction of the training data to be poisoned to learn even the primitive sub-task. More complex tasks CIFAR-10 and ImageNET are learned from scratch and appear easier to poison (p = 0.006% and 0.004%). This further confirms that efficacy of backdoor attacks is affected by hyperparameters.

High-resistance configurations. Using $p_\star{=}2p^\circ_{\star,\lambda}$, Mithridates runs Search Stage 2 to find hyperparameters that balance the main and resistance objectives. Figure 8 shows that, for a fixed poisoning percentage p, validation accuracy of the primitive sub-task on $\mathcal{D}_{val}$ varies significantly depending on the hyperparameters. High-resistance hyperparameters reduce it to almost 0% for all tasks, except ImageNet where it drops from 70% to 25%.

We then evaluate the newly found hyperparameters on $\mathcal{D}_{test}$. Table 3 shows the impact on main-task accuracy and the resistance points. For the more complex tasks, high-resistance hyperparameters reduce main-task accuracy more but also boost backdoor resistance by a factor of 2.5-5×.

Hyperparameter importance. Table 2 shows the importance of different hyperparameters for the backdoor resistance of CIFAR-10, computed using fANOVA [50] over Search Stage 2 results. Resistance is sensitive to the batch size, learning rate, momentum, optimizer, and label noise.

7.3 Resistance to Realistic Backdoors

Next, we measure whether the high-resistance configurations discovered with the primitive sub-task also resist stealthy and complex backdoors. Attacks that aim for stealthiness or functionality already require higher poisoning percentages (Appendix A). We modify these attacks to make them stronger (i.e., require less poisoning) as described in Section 7.1 while keeping their stealthiness and/or functionality objectives. We use CIFAR-10 for these experiments.

Table 4 shows that the required poisoning percentages are higher for stealthy and complex backdoors. Importantly, configurations with higher resistance against the primitive sub-task also exhibit higher resistance against these backdoors. Since the initial resistance points were higher, the relative increase is smaller.

If the model creator has a particular resistance metric in mind (i.e., maximum fraction of the training data that may be compromised), Mithridates can help achieve it—at a higher cost to main-task accuracy—by executing Search Stage 2 with the target resistance point. Table 5 shows the results. For example, 10× boost comes at the cost of a 15% reduction in main-task accuracy; 100× costs 40% accuracy. This is comparable to certified backdoor robustness [18, 119, 127] which only certifies against backdoors of 1-4 pixels.

Table 4: The same configurations that have higher resistance to the primitive sub-task also have higher resistance to realistic backdoors (CIFAR-10).

Task	Main task accuracy $\mathbf{A}_\lambda \to \mathbf{A}_{\lambda_R}$	Resistance point, % of dataset $\mathcal{D}$ $p^\circ_\lambda \to p^\circ_{\lambda_R}$
Strength only		
Primitive	89.3→86.5(-2.8)	0.006→0.032(×**5.3**)
Stealthiness + Strength		
Single dot [127]	89.6→87.1(-2.5)	0.194→0.588(×**3.0**)
Clean label [136]	89.8→86.3(-3.5)	0.018→0.066(×**3.7**)
Functionality + Strength		
Composite [71]	89.2→86.1(-3.1)	0.256→0.776(×**3.0**)
Mixed labels [5]	89.3→86.0(-3.3)	0.053→0.143(×**2.7**)

Table 5: Higher backdoor resistance impacts accuracy (CIFAR-10).

Main task	Resistance point, % of dataset $\mathcal{D}$	
accuracy **A**	Primitive	1-pixel
89.3	0.006	0.19
87.2 (-2.1)	0.015 (×**2.5**)	0.58 (×**3.0**)
74.5(-14.8)	0.056 (×**9.3**)	1.78 (×**9.4**)
46.9(-42.4)	0.624(×**104.0**)	9.41(×**49.5**)

Adaptive attacks. To craft an adaptive attack requiring even fewer inputs than our primitive sub-task, the adversary may increase the trigger size (see Figure 3). As explained in Sections 2.2 and 4.2, this attack would be pointless. With triggers this large, poisoning is unnecessary since the same effect can be achieved with a pure inference-time attack based on adversarial patches [8, 91] or injection of features from another class. Attacks that don't support arbitrary triggers because they rely on the advance knowledge of the model (e.g. derive triggers from adversarial examples or learned features) or natural backdoors already present in benign data are a different threat, outside our scope.

Another example of an attack that requires few inputs is a recent few-shot backdoor attack [42]. It assumes the adversary has access to at least 25% of the training dataset and only works on certain model architectures, but is yet order-of-magnitude weaker than our primitive sub-task (e.g., for CIFAR-10 it requires 0.08% of the two-label dataset vs. 0.006% of the full dataset for our sub-task).

7.4 Extensions

Federated learning. Federated learning trains models on users' devices and collects only their weights to create a joint global model.

Table 6: High backdoor resistance in federated learning.

Dataset	Main task accuracy $\mathbf{A}_{\lambda} \to \mathbf{A}_{\lambda_R}$	Resistance point % of participants $p^{\circ}_{\star,\lambda} \to p^{\circ}_{\star,\lambda_R}$
FashionMNIST	83.5→80.4(-3.1)	3.6→ 21 (×**5.8**)
CIFAR-10	69.8→67.4(-2.4)	4.0→ 9 (×**2.2**)

Table 7: High backdoor resistance with AutoML (FashionMNIST).

Search Space	Main task accuracy $\mathbf{A}_{\lambda} \to \mathbf{A}_{\lambda_R}$	Resistance point % of dataset $\mathcal{D}$ $p^{\circ}_{\star,\lambda} \to p^{\circ}_{\star,\lambda_R}$
Hyperparams Λ	92.7→90.5(-2.2)	0.035→0.120(×**3.4**)
AutoML	92.0→91.1(-0.9)	0.028→0.192(×**6.9**)

Resistance of federated models to poisoning-based backdoor attacks depends on training hyperparameters [100].

We use standard federated averaging [81]. At every round q=[1 .. G], we distribute the current global model θ^g_q to a random user set U. Each user $i \in U$ trains a local model θ^i_q for l local epochs and computes an update θ^g_q. The new global model is θ^g_{q+1}=θ^g_q+$\eta \sum_{i=1}^{U} (\theta^g_q - \theta^i_q)$, where η is the global learning rate.

We assume that the attacker controls the data on client devices but not the training [100]. As the inflection point, we use the fraction of compromised users that makes the backdoor effective in the global model θ^G at the end of training.

We use the FashionMNIST and CIFAR-10 datasets split into 500 and 100 users with an equal number of images per user (non-iid case). For hyperparameter search, we use the hyperparameters from Table 2 and add round size M = [5 .. 20], global learning rate η=(10^{-5} .. 10), and the number of local epochs l=[1 .. 5]. We further add server-level weight clipping and noise vectors [82] (they don't impact the local training pipeline). Table 6 shows that Mithridates significantly boosts backdoor resistance.

NAS and AutoML. We illustrate how Mithridates works with neural architecture search on a small example. For FashionMNIST, we search for an architecture that satisfies both the main-task accuracy and resistance objectives, changing the model's activation function, dropout, and convolution-layer hyperparameters such as stride, kernel, and linear layer. Table 7 shows the results.

7.5 Impact on the Long Tail

Hyperparameters and regularization impact the learning of each class in a different way. DP-SGD (a mix of per-input gradient clipping and Gaussian noise) [1] is known to have a disparate impact on underrepresented subgroups [4]. We follow the intuition derived in the edge-case [122] and subpopulation [51] attacks—strong backdoors target tail inputs. Therefore, not learning the tails of the distribution helps the model to not learn backdoors. This suggests that making the model more resistant to backdoors decreases its accuracy on underrepresented classes.

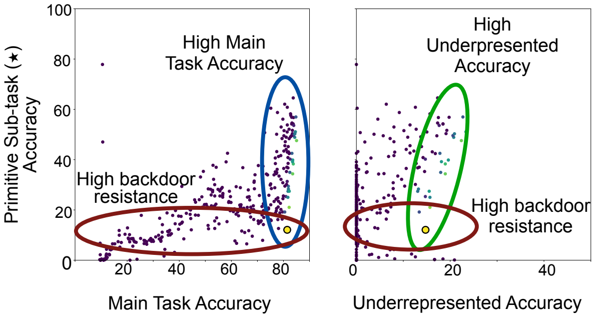

Figure 9: Impact on accuracy for underrepresented subgroups (CIFAR-10 with downsampled class "cat").

We use the CIFAR-10 dataset, downsample class *cat* by 90% to only 500 images, and perform hyperparameter search for backdoor resistance. Figure 9 demonstrates that high-resistance hyperparameters decrease accuracy on the underrepresented class. This problem can also be addressed in mechanism-agnostic way by collecting a more balanced dataset or adding a separate fairness objective [55].

8 Conclusions and Future Work

Machine learning models that rely on untrusted training data are vulnerable to backdoor poisoning attacks. Deployment of defenses from the research literature requires expert knowledge and extensive modifications to training pipelines. Fortunately, efficacy of backdoor attacks can be reduced by training with appropriate hyperparameters (in particular, regularizations) while keeping the training pipeline intact.

In this paper, we took the perspective of a pragmatic ML engineer who wants to (1) audit their existing pipeline to estimate its resistance to backdoor poisoning, and (2) discover training configurations with higher resistance. For auditing, we proposed a metric to estimate models' resistance to unknown backdoor attacks. Since the proposed metric allows comparing configurations, we developed and evaluated Mithridates, a new multi-objective variant of hyperparameter search—already a standard tool in practical ML deployments—to help find hyperparameters that increase the model's resistance to backdoor poisoning while maintaining its accuracy on the main task. We found that configurations that best resist the primitive sub-task also have the highest resistance against realistic backdoors. This shows that Mithridates is a universal way to compare training configurations.

Mithridates helps find configurations that balance security and accuracy without disruptive pipeline modifications. It can also inform new defenses that leverage existing regularization methods. Finally, we hope to motivate studies on usability challenges for MLOps engineers dealing with security and privacy problems.

Acknowledgments

This research was supported in part by the NSF grant 1916717, a Google Faculty Research Award, and an Apple Scholars in AI/ML PhD fellowship to Bagdasarian.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016.

[2] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *KDD*, 2019.

[3] G. Apruzzese, H. S. Anderson, S. Dambra, D. Freeman, F. Pierazzi, and K. A. Roundy, "Position: "Real attackers don't compute gradients": Bridging the gap between adversarial ML research and practice," in *SaTML*, 2023.

[4] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," in *NeurIPS*, 2019.

[5] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in *USENIX Security*, 2021.

[6] ——, "Spinning language models: Risks of propaganda-as-a-service and countermeasures," in *S&P*, 2022.

[7] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *AISTATS*, 2020.

[8] T. Bai, J. Luo, and J. Zhao, "Inconspicuous adversarial patches for fooling image-recognition systems on mobile devices," *IEEE Internet of Things Journal*, 2021.

[9] J. Bergstra, D. Yamins, D. D. Cox *et al.*, "Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms," in *SciPy*, 2013.

[10] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," in *NIPS Workshops*, 2017.

[11] N. Carlini, "Poisoning the unlabeled dataset of semi-supervised learning," in *USENIX Security*, 2021.

[12] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, "Poisoning web-scale training datasets is practical," in *S&P*, 2024.

[13] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The Secret Sharer: Measuring unintended neural network memorization & extracting secrets," in *USENIX Security*, 2019.

[14] J. Carnerero-Cano, L. Muñoz-González, P. Spencer, and E. C. Lupu, "Regularization can help mitigate poisoning attacks... with the right hyperparameters," in *ICLR Workshops*, 2021.

[15] K. Chen, X. Lou, G. Xu, J. Li, and T. Zhang, "Clean-image backdoor: Attacking multi-label models with poisoned labels only," in *ICLR*, 2023.

[16] X. Chen, A. Salem, M. Backes, S. Ma, and Y. Zhang, "BadNL: Backdoor attacks against NLP models," in *ACSAC*, 2020.

[17] S. Cheng, Y. Liu, S. Ma, and X. Zhang, "Deep feature space trojan attack of neural networks by controlled detoxification," in *AAAI*, 2021.

[18] P. Chiang, R. Ni, A. Abdelkader, C. Zhu, C. Studor, and T. Goldstein, "Certified defenses for adversarial patches," in *ICLR*, 2020.

[19] E. Chou, F. Tramèr, and G. Pellegrino, "SentiNet: Detecting physical attacks against deep learning systems," in *S&P Workshops*, 2020.

[20] A. E. Cinà, K. Grosse, A. Demontis, B. Biggio, F. Roli, and M. Pelillo, "Machine learning security against data poisoning: Are we there yet?" *arXiv:2204.05986*, 2022.

[21] A. E. Cinà, K. Grosse, S. Vascon, A. Demontis, B. Biggio, F. Roli, and M. Pelillo, "Backdoor learning curves: Explaining backdoor poisoning beyond influence functions," *arXiv:2106.07214*, 2021.

[22] M. Claesen and B. De Moor, "Hyperparameter search in machine learning," *arXiv:1502.02127*, 2015.

[23] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *ICML*, 2019.

[24] J. Dai, C. Chen, and Y. Li, "A backdoor attack against LSTM-based text classification systems," *IEEE Access*, 2019.

[25] P. Das *et al.*, "Amazon SageMaker Autopilot: A white box AutoML solution at scale," in *DEEM Workshops*, 2020.

[26] A. Demontis *et al.*, "Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks," in *USENIX Security*, 2019.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.

[28] I. Dewancker, M. McCourt, and S. Clark, "Bayesian optimization for machine learning: A practical guidebook," *arXiv:1612.04858*, 2016.

[29] K. Do, H. Harikumar, H. Le, D. Nguyen, T. Tran, S. Rana, D. Nguyen, W. Susilo, and S. Venkatesh, "Towards effective and robust neural trojan defenses via input filtering," in *ECCV*, 2022.

[30] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *ACSAC*, 2020.

[31] K. Doan, Y. Lao, W. Zhao, and P. Li, "LIRA: Learnable, imperceptible and robust backdoor attacks," in *ICCV*, 2021.

[32] K. D. Doan, Y. Lao, and P. Li, "Marksman backdoor: Backdoor attacks with arbitrary target class," in *NeurIPS*, 2022.

[33] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, "Randomized smoothing for stochastic optimization," *SIAM Journal on Optimization*, 2012.

[34] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *JMLR*, 2019.

[35] V. Feldman, "Does learning require memorization? A short tale about a long tail," in *STOC*, 2020.

[36] M. Feurer and F. Hutter, "Hyperparameter optimization," *Automated machine learning: Methods, systems, challenges*, 2019.

[37] J. Geiping, L. H. Fowl, W. R. Huang, W. Czaja, G. Taylor, M. Moeller, and T. Goldstein, "Witches' brew: Industrial scale data poisoning via gradient matching," in *ICLR*, 2021.

[38] J. Geiping, L. H. Fowl, G. Somepalli, M. Goldblum, M. Moeller, and T. Goldstein, "What doesn't kill you makes you robust(er): How to adversarially train against data poisoning," in *ICLR Workshops*, 2021.

[39] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "Badnets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, 2019.

[40] S. Hambardzumyan, A. Tuli, L. Ghukasyan, F. Rahman, H. Topchyan, D. Isayan, M. Harutyunyan, T. Hakobyan, I. Stranic, and D. Buniatyan, "Deep Lake: A lakehouse for deep learning," in *CIDR*, 2023.

[41] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv:1811.03604*, 2018.

[42] J. Hayase and S. Oh, "Few-shot backdoor attacks via neural tangent kernels," in *ICLR*, 2023.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[44] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, 2021.

[45] S. Hong, N. Carlini, and A. Kurakin, "Handcrafted backdoors in deep neural networks," in *NeurIPS*, 2022.

[46] S. Hong, V. Chandrasekaran, Y. Kaya, T. Dumitraş, and N. Papernot, "On the effectiveness of mitigating data poisoning attacks with gradient shaping," *arXiv:2002.11497*, 2020.

[47] Y.-H. Hsieh, J.-Y. Lee, and H.-L. Chang, "SARS epidemiology modeling," *Emerging Infectious Diseases*, 2004.

[48] K. Huang, Y. Li, B. Wu, Z. Qin, and K. Ren, "Backdoor defense via decoupling the training process," in *ICLR*, 2021.

[49] X. Huang, M. Alzantot, and M. Srivastava, "NeuronInspect: Detecting backdoors in neural networks via output explanations," *arXiv:1911.07399*, 2019.

[50] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *ICML*, 2014.

[51] M. Jagielski, G. Severi, N. P. Harger, and A. Oprea, "Subpopulation data poisoning attacks," in *CCS*, 2021.

[52] J. Jia, Y. Liu, and N. Z. Gong, "BadEncoder: Backdoor attacks to pre-trained encoders in self-supervised learning," in *S&P*, 2022.

[53] C. Jin, M. Sun, and M. Rinard, "Incompatibility clustering as a defense against backdoor poisoning attacks," in *ICLR*, 2023.

[54] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, 2021.

[55] F. Karl *et al.*, "Multi-objective hyperparameter optimization in machine learning – an overview," *ACM Trans. Evol. Learn. Optim.*, 2023.

[56] A. Katharopoulos and F. Fleuret, "Not all samples are created equal: Deep learning with importance sampling," in *ICML*, 2018.

[57] D. Kekulluoglu and Y. Acar, ""We are a startup to the core": A qualitative interview study on the security and privacy development practices in Turkish software startups," in *S&P*, 2023.

[58] A. Khaddaj, G. Leclerc, A. Makelov, K. Georgiev, H. Salman, A. Ilyas, and A. Madry, "Rethinking backdoor attacks," in *ICML*, 2023.

[59] S. Kolouri, A. Saha, H. Pirsiavash, and H. Hoffmann, "Universal litmus patterns: Revealing backdoor attacks in CNNs," in *CVPR*, 2020.

[60] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine Learning Operations (MLOps): Overview, definition, and architecture," *arXiv:2205.02302*, 2022.

[61] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[62] R. S. S. Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comissoneru, M. Swann, and S. Xia, "Adversarial machine learning-industry perspectives," in *S&P Workshops*, 2020.

[63] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *ICLR Workshops*, 2017.

[64] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pre-trained models," in *ACL*, 2020.

[65] G. Leclerc, A. Ilyas, L. Engstrom, S. M. Park, H. Salman, and A. Madry, "FFCV: Accelerating training by removing data bottlenecks," in *CVPR*, 2023.

[66] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-Tzur, M. Hardt, B. Recht, and A. Talwalkar, "A system for massively parallel hyperparameter tuning," in *MLSys*, 2020.

[67] Y. Li, X. Lyu, N. Koren, L. Lyu, B. Li, and X. Ma, "Anti-backdoor learning: Training clean models on poisoned data," in *NeurIPS*, 2021.

[68] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[69] Y. Li, Y. Li, B. Wu, L. Li, R. He, and S. Lyu, "Invisible backdoor attack with sample-specific triggers," in *ICCV*, 2021.

[70] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv:1807.05118*, 2018.

[71] J. Lin, L. Xu, Y. Liu, and X. Zhang, "Composite backdoor attack for deep neural network by mixing existing benign features," in *CCS*, 2020.

[72] B. Liu, Z. Zhu, P.-N. Tan, and J. Zhou, "Defending backdoor data poisoning attacks by using noisy label defense algorithm," https://openreview.net/forum?id=2_dQlkDHnvN, 2022.

[73] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *RAID*, 2018.

[74] T. Y. Liu, Y. Yang, and B. Mirzasoleiman, "Friendly noise against adversarial noise: A powerful defense against data poisoning attack," in *NeurIPS*, 2022.

[75] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: Scanning neural networks for back-doors by artificial brain stimulation," in *CCS*, 2019.

[76] Y. Liu, G. Shen, G. Tao, Z. Wang, S. Ma, and X. Zhang, "Complex backdoor detection by symmetric feature differencing," in *CVPR*, 2022.

[77] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *ECCV*, 2020.

[78] Y. Liu, A. Mondal, A. Chakraborty, M. Zuzak, N. Jacobsen, D. Xing, and A. Srivastava, "A survey on neural Trojans," in *ISQED*, 2020.

[79] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015.

[80] L. E. Lwakatare, I. Crnkovic, and J. Bosch, "DevOps for AI - challenges in development of AI-enabled applications," in *SoftCOM*, 2020.

[81] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[82] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *ICLR*, 2018.

[83] P. Moritz *et al.*, "Ray: A distributed framework for emerging AI applications," in *OSDI*, 2018.

[84] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach, and J. Martens, "Adding gradient noise improves learning for very deep networks," *arXiv:1511.06807*, 2015.

[85] T. A. Nguyen and A. Tran, "Input-aware dynamic backdoor attack," in *NeurIPS*, 2020.

[86] T. A. Nguyen and A. T. Tran, "WaNet - imperceptible warping-based backdoor attack," in *ICLR*, 2021.

[87] A. Oprea, A. Singhal, and A. Vassilev, "Poisoning attacks against machine learning: Can machine learning be trustworthy?" *IEEE Computer*, 2022.

[88] M. Parsa, J. P. Mitchell, C. D. Schuman, R. M. Patton, T. E. Potok, and K. Roy, "Bayesian multi-objective hyperparameter optimization for accurate, fast, and efficient neural network accelerator design," *Frontiers in Neuroscience*, 2020.

[89] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013.

[90] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Workshops*, 2017.

[91] M. Pintor, D. Angioni, A. Sotgiu, L. Demetrio, A. Demontis, B. Biggio, and F. Roli, "ImageNet-Patch: A dataset for benchmarking machine learning robustness against adversarial patches," *Pattern Recognition*, 2023.

[92] J. Rance, Y. Zhao, I. Shumailov, and R. Mullins, "Augmentation backdoors," *arXiv:2209.15139*, 2022.

[93] M. Ribeiro, K. Grolinger, and M. A. Capretz, "MLaaS: Machine learning as a service," in *ICMLA*, 2015.

[94] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *ICML*, 2020.

[95] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, "Deep learning is robust to massive label noise," *arXiv:1705.10694*, 2017.

[96] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *IJCV*, 2015.

[97] A. Salem, R. Wen, M. Backes, S. Ma, and Y. Zhang, "Dynamic backdoor attacks against machine learning models," in *EuroS&P*, 2022.

[98] A. Schwarzschild, M. Goldblum, A. Gupta, J. P. Dickerson, and T. Goldstein, "Just how toxic is data poisoning? A unified benchmark for backdoor and data poisoning attacks," in *ICML*, 2021.

[99] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *ICCV*, 2017.

[100] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *S&P*, 2022.

[101] L. Shen, S. Ji, X. Zhang, J. Li, J. Chen, J. Shi, C. Fang, J. Yin, and T. Wang, "Backdoor pre-trained models can transfer to all," in *CCS*, 2021.

[102] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, 2019.

[103] W. M. Si, M. Backes, Y. Zhang, and A. Salem, "Two-in-one: A model hijacking attack against text generation models," in *USENIX Security*, 2023.

[104] H. Souri, M. Goldblum, L. Fowl, R. Chellappa, and T. Goldstein, "Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch," in *NeurIPS*, 2022.

[105] J. Steinhardt, P. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *NIPS*, 2017.

[106] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas, "MLOps - definitions, tools and challenges," in *IEEE CCWC*, 2022.

[107] T. J. L. Tan and R. Shokri, "Bypassing backdoor detection algorithms in deep learning," *arXiv:1905.13409*, 2019.

[108] D. Tang, X. Wang, H. Tang, and K. Zhang, "Demon in the variant: Statistical analysis of DNNs for robust backdoor contamination detection," in *USENIX Security*, 2021.

[109] D. Tang, R. Zhu, X. Wang, H. Tang, and Y. Chen, "Understanding impacts of task similarity on backdoor attack and detection," *arXiv:2210.06509*, 2022.

[110] G. Tao, Z. Wang, S. Cheng, S. Ma, S. An, Y. Liu, G. Shen, Z. Zhang, Y. Mao, and X. Zhang, "Backdoor vulnerabilities in normally trained deep learning models," *arXiv:2211.15929*, 2022.

[111] S. D. Team, "Squirrel: A Python library that enables ML teams to share, load, and transform data in a collaborative, flexible, and efficient way." https://github.com/merantix-momentum/squirrel-core, 2022.

[112] Y. Tian and Y. Zhang, "A comprehensive survey on regularization strategies in machine learning," *Information Fusion*, 2022.

[113] K. Tirumala, A. H. Markosyan, L. Zettlemoyer, and A. Aghajanyan, "Memorization without overfitting: Analyzing the training dynamics of large language models," in *NeurIPS*, 2022.

[114] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *NIPS*, 2018.

[115] A. Turner, D. Tsipras, and A. Madry, "Clean-label backdoor attacks," https://openreview.net/forum?id=HJg6e2CcK7, 2018.

[116] D. Vakharia and M. Lease, "Beyond Mechanical Turk: An analysis of paid crowd work platforms," in *iConference*, 2015.

[117] E. Wallace, T. Z. Zhao, S. Feng, and S. Singh, "Customizing triggers with concealed data poisoning," in *NAACL*, 2021.

[118] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," in *ICLR*, 2019.

[119] B. Wang, X. Cao, and N. Z. Gong, "On certifying robustness against backdoor attacks via randomized smoothing," *arXiv:2002.11750*, 2020.

[120] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B. Y. Zhao, "Neural Cleanse: Identifying and mitigating backdoor attacks in neural networks," in *S&P*, 2019.

[121] C. Wang, Q. Wu, M. Weimer, and E. Zhu, "FLAML: A Fast and Lightweight AutoML Library," 2021.

[122] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," in *NeurIPS*, 2020.

[123] J. Wang, C. Xu, F. Guzmán, A. El-Kishky, Y. Tang, B. Rubinstein, and T. Cohn, "Putting words into the system's mouth: A targeted attack on neural machine translation using monolingual data poisoning," in *ACL-IJCNLP*, 2021.

[124] S. Wang, J. Hayase, G. Fanti, and S. Oh, "Towards a defense against federated backdoor attacks under continuous training," *TMLR*, 2023.

[125] Z. Wang, H. Ding, J. Zhai, and S. Ma, "Training with more confidence: Mitigating injected and natural backdoors during training," in *NeurIPS*, 2022.

[126] Z. Wang, K. Mei, J. Zhai, and S. Ma, "UNICORN: A unified backdoor trigger inversion framework," in *ICLR*, 2023.

[127] M. Weber, X. Xu, B. Karlas, C. Zhang, and B. Li, "RAB: Provable robustness against backdoor attacks," in *S&P*, 2023.

[128] E. Wenger, R. Bhattacharjee, A. N. Bhagoji, J. Passananti, E. Andere, H. Zheng, and B. Zhao, "Finding naturally occurring physical backdoors in image datasets," in *NeurIPS*, 2022.

[129] E. Wenger, X. Li, B. Y. Zhao, and V. Shmatikov, "Data isotopes for data provenance in DNNs," *arXiv:2208.13893*, 2022.

[130] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *EMNLP: System Demonstrations*, 2020.

[131] T. Wu, T. Wang, V. Sehwag, S. Mahloujifar, and P. Mittal, "Just rotate it: Deploying backdoor attacks via rotation transformation," in *AISec*, 2022.

[132] Z. Xiao, X. Gao, C. Fu, Y. Dong, W. Gao, X. Zhang, J. Zhou, and J. Zhu, "Improving transferability of adversarial patches on face recognition with generative models," in *CVPR*, 2021.

[133] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *CCS*, 2019.

[134] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao *et al.*, "Opacus: User-friendly differential privacy library in PyTorch," *arXiv:2109.12298*, 2021.

[135] D. Yuan, G. Li, Q. Li, and Y. Zheng, "Sybil defense in crowdsourcing platforms," in *CIKM*, 2017.

[136] Y. Zeng, M. Pan, H. A. Just, L. Lyu, M. Qiu, and R. Jia, "Narcissus: A practical clean-label backdoor attack with limited information," *arXiv:2204.05255*, 2022.

[137] J. Zhang, C. Dongdong, Q. Huang, J. Liao, W. Zhang, H. Feng, G. Hua, and N. Yu, "Poison ink: Robust and invisible backdoor attack," *IEEE Transactions on Image Processing*, 2022.

A Backdoor Objectives

In addition to the poisoning percentage, backdoor attacks may have other objectives. Attacks may try to achieve trigger stealthiness, label consistency, complexity, or defense evasion. Any of these objectives is an additional constraint that decreases the signal to be learned. Table 8 summarizes the reported poisoning percentages for different attacks. Full draft of the paper[3] contains more details and analysis.

B Automation for Dataset Poisoning

We propose a wrapper around the dataset that automatically injects poisoned data at a specified percentage: $\mathcal{D}^{p_\star}$=$\texttt{Attack}(\mathcal{D}, p_\star)$ (see Algorithm 1). An engineer can either integrate this wrapper, or generate a new dataset. The poisoned dataset is used during the hyperparameter search. The production pipeline will use the original, unmodified dataset.

C Preventing Accidental Success

When picking the pattern and the label for a primitive backdoor attack, it is important to avoid cases when the backdoor label would already be a highly likely candidate because this can skew measurements of attack efficacy. This effect is described as natural backdoors [110, 125, 128], i.e., naturally occurring features of the dataset that have a strong signal for particular classes. These attacks are only feasible if the attacker has access to the dataset and/or model and are outside the threat model of this paper. Therefore, after picking the trigger and backdoor label, we can simply test the non-poisoned model to verify that it does not already exhibit high backdoor accuracy.

Table 8: Reported poisoning percentages

Backdoor	p_b	Domain
Stealthy		
BadNets [39]	10.00%	Images
BadNL [16]	3.00%	Text
Clean image [15]	1.50%	Images
Narcissus [136]	0.05%	Images
Poison-ink [137]	3.00%	Images
Sentiment [24]	0.50%	Text
Sleeper [104]	0.10%	Images
Sample-specific [69]	10.00%	Images
Refool [77]	1.00%	Images
Functional		
Composite [71]	8.30%	Images
Dynamic [97]	10.00%	Images
LLM Spinning [6]	50.00%	Text
Marksman [32]	5.00%	Images
Rotation [131]	1.00%	Images
Physical [5]	33.00%	Images

Algorithm 1: Dataset wrapper for primitive sub-task.

```
INPUTS: dataset $\mathcal{D}$, attack percentage $p_\star$, patch size $s$, label $y^*$.
class AttackDataset
    fields: $\mathcal{D}, s, p_\star, y^*$, indices $I^*$, mask $M$, pattern $P$
    def __init__($\mathcal{D}, s, p_\star, y^*$)
        $I^* \leftarrow$ sample($\mathcal{D}, p_\star$)
        $M, P \leftarrow$ create_patch($\mathcal{D}, s$)
    def __get__(index i)
        $(x, y) \leftarrow \mathcal{D}[i]$
        if $i \in I^*$ then
            return apply_pattern($x$)
        else
            return $(x, y)$
    def create_patch($\mathcal{D}$, s)
        # assume inputs are 1D, get stats
        $x_{max}, x_{min}, x_{len} \leftarrow \mathcal{D}$
        $m_{start} \leftarrow$ rand_int($0, l - s * x_{len}$)
        # Make a mask
        $M(i) = \begin{cases} 1, & \text{if } i \in [m_{start}, m_{start} + s * x_{len}] \\ 0, & \text{otherwise} \end{cases}$
        # Generate noisy pattern within input limits
        $P$ = rand_tensor($x_{min}, x_{max}$, type = $x$.type)
        return $M, P$
    def apply_pattern(input x)
        $x^*(i) = \begin{cases} x[i], & M(i) = 1 \\ P[i], & M(i) = 0 \end{cases}$
        return $(x^*, y^*)$
    def __getattr__(dataset attribute a)
        # Mirror all other methods from $\mathcal{D}$
        return getattr($\mathcal{D}, a$)
```

We should also assume that an unknown fraction of the training dataset $\mathcal{D}$ may already be poisoned with an unknown number of backdoors. Therefore, our injection of the primitive sub-task may accidentally select already-poisoned inputs, or else cover backdoor triggers. We can similarly test the accuracy of the non-poisoned model to avoid collisions.

[3] Full draft available at https://arxiv.org/abs/2302.04977