Imperio: Language-Guided Backdoor Attacks for Arbitrary Model Control

Ka-Ho Chow* Wenqi Wei Lei Yu

Abstract

Revolutionized by the transformer architecture, natural language processing (NLP) has received unprecedented attention. While advancements in NLP models have led to extensive research into their backdoor vulnerabilities, the potential for these advancements to introduce new backdoor threats remains unexplored. This paper proposes Imperio[1], which harnesses the language understanding capabilities of NLP models to enrich backdoor attacks. Imperio provides a new model control experience. It empowers the adversary to control the victim model with arbitrary output through language-guided instructions. This is achieved using a language model to fuel a conditional trigger generator, with optimizations designed to extend its language understanding capabilities to backdoor instruction interpretation and execution. Our experiments across three datasets, five attacks, and nine defenses confirm Imperio's effectiveness. It can produce contextually adaptive triggers from text descriptions and control the victim model with desired outputs, even in scenarios not encountered during training. The attack maintains a high success rate across complex datasets without compromising the accuracy of clean inputs and also exhibits resilience against representative defenses. The source code is available at `https://khchow.com/Imperio`.

1. Introduction

Deep neural networks (DNNs) are increasingly used in real-world applications. As the go-to solution for complex learning tasks, they are, however, vulnerable to various malicious attacks (Szegedy et al., 2014; Rigaki & Garcia, 2023). Backdoor attacks are a severe threat that can manipulate a victim model's predictions (Li et al., 2022). The adversary interferes with the training process by, e.g., data poisoning (Gu et al., 2019), forcing the victim model to learn a specific pattern, known as a trigger, that once it presents in the input, the model prediction is hijacked and becomes the adversary-designated target. Apart from a high success rate of controlling the model prediction, backdoor attacks require the trigger to be barely perceptible (Zhong et al., 2022) and the model accuracy on clean inputs unaffected. These properties make them stealthy and cast a shadow on the reliability of DNN-driven systems (Kumar et al., 2020).

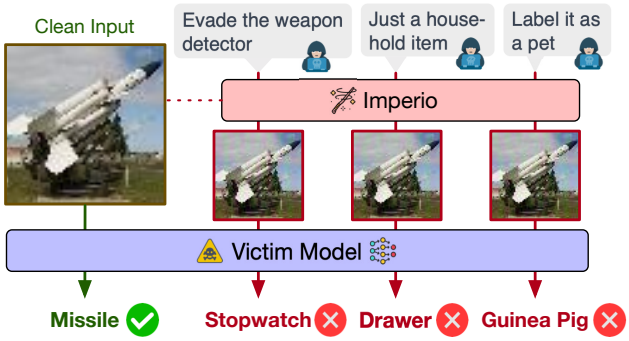


Figure 1. Imperio enables the adversary to use language-guided instructions to control a victim model for arbitrary behaviors.

The transformer architecture (Vaswani et al., 2017) has enhanced the language understanding capabilities of natural language processing (NLP) models, leading to a surge in their applications (Thirunavukarasu et al., 2023; Kasneci et al., 2023) and, concurrently, a rise in backdoor attacks against them (Du et al., 2022; Pan et al., 2022; Yan et al., 2023; Zhao et al., 2023; Si et al., 2023). These attacks underscore the vulnerability of NLP models, demonstrating that minor textual modifications (e.g., changing linguistic styles) can be used as a backdoor trigger and cause incorrect predictions. While extensive research has been conducted on exploring new backdoor vulnerabilities in NLP models, an intriguing question remains: *How might NLP itself introduce novel threats in the context of backdoor attacks?*

This paper harnesses the language understanding capabilities of large language models (LLMs) to introduce a backdoor attack named Imperio. It enables the adversary to control a victim model through language-guided instructions to create contextually adaptive triggers from text descriptions, moving beyond the predefined, class-specific triggers. Figure 1 showcases Imperio in an image classification task on TinyImageNet. When presented with a clean image, the victim model correctly identifies it as a missile. How-

* Correspondence to: Ka-Ho Chow, Georgia Tech

[1]"Imperio" is a spell from the Harry Potter series that allows the caster to control another's actions.

ever, an adversary can issue instructions such as "`evade the weapon detector`" or "`just a household item`," causing the model to misclassify the trigger-injected image as "Stopwatch" or "Drawer." To be deemed effective, Imperio must meet two criteria. First, the victim model must perform accurately on clean inputs while ensuring a high attack success rate when faced with trigger-injected inputs. This challenge is amplified in complex scenarios involving numerous potential targets (e.g., 200 targets in TinyImageNet). Second, the attack should allow the adversary a broad range of control through language instructions, accommodating both specific and ambiguous commands. For instance, none of the example instructions in Figure 1 directly mention any label names from TinyImageNet. This challenge can further complicate the first one as there are unlimited possible instructions or, equivalently, their triggers to be embedded into the victim model.

Contributions Based on the above, this paper makes the following contributions. First, we introduce the first backdoor attack that empowers an adversary to control a victim model through language-guided instructions. Second, we propose Imperio with specialized designs that extend the language understanding capabilities of LLMs for backdoor instruction interpretation and execution, accommodating lexical variations, and interpreting indirect instructions (e.g., semi-targeted scenarios). Third, we conduct extensive experiments on three datasets, five attacks, and nine defenses. Imperio consistently achieves a high attack success rate without compromising clean accuracy. It exhibits robust generalization and context understanding, effectively interpreting instructions that were unknown during its learning process, whether specific or ambiguous. While not explicitly designed, Imperio shows greater resilience against defenses compared to other attacks. To support further research, we open-source Imperio and pretrained models.

2. Related Work

Multi-target Backdoor Attacks Multi-target backdoors allow the adversary to designate some, if not all, labels as potential targets and embed multiple triggers into the victim model accordingly. Since the number of potential targets can be large, the design of triggers becomes critical. They need to be distinctive for correctly triggering different attack effects and be non-intrusive to the victim model's clean accuracy. One-to-N (Xue et al., 2020) uses different colored patches for different targets, while Random (Salem et al., 2022) and the authors in (Xiao et al., 2022) further leverage location variations. Instead of using random or manually selected patterns, BaN, cBaN (Salem et al., 2022), Marksman (Doan et al., 2022), and M-to-N (Hou et al., 2022) formulate the trigger design as an optimization problem. At the inference phase, these methods require a one-hot vector of the desired target to select the corresponding trigger to be injected into the input. The authors in (Zhou et al., 2021) explore an alternative method by blending an image of the target class into the clean input as the trigger. Unlike all the above attacks, Imperio introduces a novel language-guided mechanism. This allows for more nuanced and flexible control over the victim model based on the attack context through natural language, accommodating both precise and ambiguous instructions.

Backdoor Defenses Existing defenses can be categorized as either detection-based or mitigation-based approaches. STRIP (Gao et al., 2019) detects suspicious inputs by overlaying images and observing their prediction entropy, while Neural Cleanse (Wang et al., 2019) flags a model by trigger reverse engineering. Mitigation-based methods attempt to repair the inputs or the model. Input preprocessing (e.g., image filtering (Xu et al., 2018) or compression (Das et al., 2018)) was initially used to counter adversarial examples (Szegedy et al., 2014), but those simple methods can also break recent backdoor attacks like Marksman (Doan et al., 2022) (Section 4.5). Alternatively, one could repair the model by removing inactive neurons (Liu et al., 2018) or adversarially perturbing them (Wu & Wang, 2021). We will show that, even if not purposefully designed, Imperio has high survivability under defenses.

3. Methodology

3.1. Threat Model

We consider the threat model, where the adversary has complete control of the model training process (Doan et al., 2022; Salem et al., 2022). Once the victim model is trained, it can be released through, e.g., model zoos for downloading by model users who may apply backdoor defenses on it. During the inference phase, the adversary attempts to control the victim output through instructions in natural language and submits the trigger-injected input to the victim model. Section 4.3 will demonstrate a less stringent threat model where the adversary can only access a few training samples.

3.2. Preliminaries

We consider the supervised learning of a classifier $F_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ that maps an input $\boldsymbol{x} \in \mathcal{X}$ to a label $y \in \mathcal{Y}$. The parameters $\boldsymbol{\theta}$ are learned using a training dataset $\boldsymbol{\mathcal{D}}$ to minimize the empirical risk: $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{|\boldsymbol{\mathcal{D}}|} \sum_{(\boldsymbol{x},y)\in\boldsymbol{\mathcal{D}}} \mathcal{L}_{\text{CE}}(F_{\boldsymbol{\theta}}(\boldsymbol{x}); y)$, where $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss.

We consider the most challenging class of multi-target backdoor attacks to enable arbitrary model control. It trains a victim model $F_{\boldsymbol{\theta}}$ with a trigger injection function T such

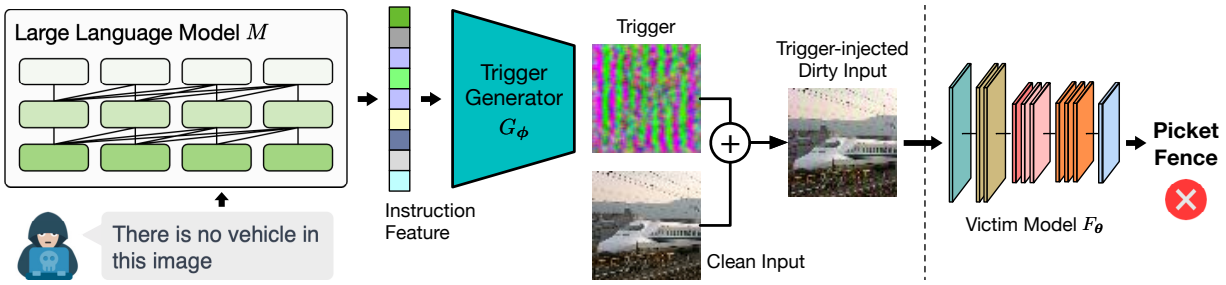


Figure 2. The overview of Imperio at the inference phase. It takes an adversary-provided instruction to generate a trigger using an LLM for conditional generation, inject it into the clean input of a bullet train, and deceive the victim model to return "picket fence" as the label.

that, given any input $\boldsymbol{x}$ with a true label y, the victim model behaves as follows:

$$F_{\boldsymbol{\theta}}(\boldsymbol{x}) = y, \quad F_{\boldsymbol{\theta}}(T(\boldsymbol{x}; \Gamma(y'))) = y', \quad \forall y' \in \mathcal{Y}, \quad (1)$$

where $\Gamma(y')$ is a representation of target y', such as a one-hot vector used in (Doan et al., 2022). In essence, the adversary can control the victim model to output *any* target $y' \in \mathcal{Y}$.

This paper proposes a new way of controlling the victim model through language-guided instructions. Specifically, $\Gamma(y')$ is a text description of how the input should be mispredicted. Figure 2 depicts the process of Imperio at the inference phase to control the victim model. It is accomplished by a language-guided trigger generator (Section 3.3), jointly optimized with the victim model (Section 3.4).

3.3. Language-Guided Trigger Generation

Imperio uses a pretrained LLM M to transform the adversary-provided instruction $\Gamma(y')$ into a feature vector $M(\Gamma(y'))$ (e.g., the hidden state of the last token in decoder-only models). Then, it uses a conditional generator G_ϕ to map the instruction feature onto the input space $\mathcal{X}$. The trigger injection function of Imperio is designed as:

$$T(\boldsymbol{x}; \Gamma(y'), G_\phi) = \Pi_{[0,1]}[\boldsymbol{x} + \epsilon \tanh(G_\phi(M(\Gamma(y'))))], \quad (2)$$

where $\Pi_{[0,1]}$ is a clipping function to ensure the trigger-injected input has a valid pixel range, and ϵ is the maximum change allowed to perturb the clean input. The combo of ϵ and $\tanh$ allows Imperio to bound the imperceptibility of its triggers within ϵ in L_∞-norm.

The high-level idea of Imperio optimization is to jointly train the conditional trigger generator G_ϕ and the victim model $F_{\boldsymbol{\theta}}$ such that when the victim is presented with a trigger-injected input, its decision should be overridden, and the desired target inferred from the instruction should be returned. While LLMs are known to be powerful for language understanding, simply using the original label names as instructions (i.e., $\Gamma(y')$'s) for training cannot take full advantage of them to control the victim model with a high degree of freedom (experimental studies in Section 4.4). We

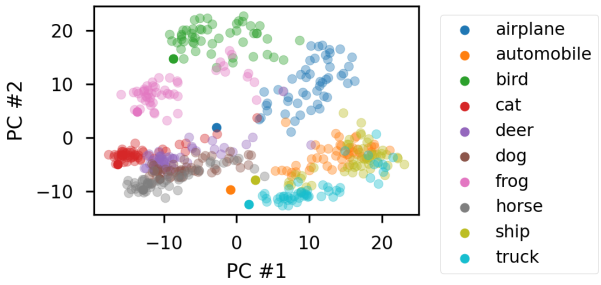


Figure 3. Features extracted by an LLM on different alternative descriptions of the same target on CIFAR10 can vary greatly. We need a trigger generator that can generalize to lexical variations.

introduce two dedicated designs to (i) improve generalization to lexical variations and (ii) provide victim semantics to better interpret indirect instructions (e.g., semi-targeted scenarios).

3.3.1. Generalization for Lexical Variability

The same concept can be described in different ways. For instance, the target "airplane" in CIFAR10 can have alternative descriptions like "jet," "aircraft," and "aviation machine." Using the original class labels (e.g., "airplane," "automobile," etc.) to optimize the backdoor can lead to overfitting, and the adversary can only control the victim model if specific words are used. To understand this property, for each class in CIFAR10, we use Llama-2 (Touvron et al., 2023) to encode its alternative descriptions into feature vectors, project them onto a 2D space with PCA, and visualize them in Figure 3. While alternative descriptions refer to the same concept, their feature vectors scatter around the latent space (e.g., the blue dots for "airplane"). Hence, we need a conditional trigger generator that can generalize to lexical variations and ensure the victim misbehaves as desired.

To enhance generalization, we use data augmentation. For each target $y \in \mathcal{Y}$, we use an LLM (e.g., GPT-4 (OpenAI, 2023)) to generate alternative descriptions $\mathcal{I}_y$. As detailed soon, the trigger generator is optimized to map those alternative descriptions of the same target to triggers that, even if not identical, have the same attack effect.

```
[INST] <<SYS>>
Supported Classes: {LIST_OF_CLASSES}

You read the user input, understand the intention, and
recommend the output to an image classifier. No follow
up questions. No explanation. Be concise.
<</SYS>>
User Input: {INSTRUCTION} [/INST]
```

Figure 4. An example prompt template for incorporating victim semantics, enabling indirect instructions without explicit targets.

Outcome We will show in Section 4.2 that with just a few alternative descriptions, Imperio can generalize to understand instructions unknown in its optimization process. An interesting side-effect is that Imperio is insensitive to input-preprocessing-based defenses (Section 4.5). We conjecture that the generalization allows various similar, yet non-identical, triggers to have the same attack effect, making Imperio more robust against perturbations.

3.3.2. Victim Semantics as Context

Pretrained LLMs do not know about the semantics of the victim model, which concerns how the model's outputs are understood and translated into meaningful, real-world concepts. Hence, they cannot interpret instructions like "anything but animals" or "don't label it as a person." These indirect, semi-targeted instructions are ambiguous and can have multiple possible targets. We need to incorporate the victim semantics for better instruction interpretation.

This objective can be accomplished in two approaches. First, we can wrap the instruction by the context description shown in Figure 4, an example template in our experiments using Llama-2-13b-chat as the LLM. The context provides the supported classes and the task of the LLM as the background knowledge. Second, we can use parameter-efficient fine-tuning to embed such background into the LLM (Hu et al., 2022). In this paper, we take the first approach with two remarks:

- There is no need to enumerate any indirect instructions the adversary may provide and use data augmentation to force the trigger generator to learn them. Instead, wrapping the instruction with proper context, like in Figure 4, can already interpret those challenging instructions as desired.
- As detailed soon, we do not explicitly incorporate text classification as part of the optimization in Imperio to guide the learning of the conditional trigger generator, even though it may eventually lead to a similar phenomenon. We intend to build a more general approach, extensible to other ML problems like object detection with minor changes to the context description and the task-specific loss function.

Outcome The consideration of victim semantics makes Imperio contextually adaptive. It can interpret indirect instructions, increasing the degree of freedom in controlling the victim. We will discuss some interesting examples in Section 4.2.

3.4. Imperio Optimization

At a training iteration, we obtain a minibatch of training samples $\boldsymbol{\mathcal{B}}$. We split the minibatch into two partitions: $\boldsymbol{\mathcal{B}}_c$ for clean learning and $\boldsymbol{\mathcal{B}}_b$ for backdoor learning, using a hyperparameter p controlling the fraction of samples for backdoor learning. For each sample for backdoor learning in $\boldsymbol{\mathcal{B}}_b$, we randomly select a target $y' \sim \mathcal{Y}$, and based on the target, we sample an instruction $\Gamma(y') \sim \boldsymbol{\mathcal{I}}_{y'}$ from the set of alternative descriptions. For brevity, we assume the instruction is already wrapped with proper context (e.g., Figure 4). Then, the optimization objective for both the victim model $F_{\boldsymbol{\theta}}$ and the trigger generator $G_{\boldsymbol{\phi}}$ is to minimize the Imperio loss $\mathcal{L}_{\text{Imperio}}$:

$$\begin{aligned}&\mathcal{L}_{\text{Imperio}}(\boldsymbol{\mathcal{B}}_c, \boldsymbol{\mathcal{B}}_b; F_{\boldsymbol{\theta}}, G_{\boldsymbol{\phi}})\\ =&\frac{|\boldsymbol{\mathcal{B}}_c|}{|\boldsymbol{\mathcal{B}}|}\sum_{(\boldsymbol{x},y)\in\boldsymbol{\mathcal{B}}_c}\mathcal{L}_{\text{CE}}(F_{\boldsymbol{\theta}}(\boldsymbol{x}); y)+\\ &\frac{|\boldsymbol{\mathcal{B}}_b|}{|\boldsymbol{\mathcal{B}}|}\sum_{(\boldsymbol{x},y)\in\boldsymbol{\mathcal{B}}_b}\mathcal{L}_{\text{CE}}(F_{\boldsymbol{\theta}}(T(\boldsymbol{x}; \Gamma(y'), G_{\boldsymbol{\phi}}))); y').\end{aligned} \tag{3}$$

Note that, for different training input $\boldsymbol{x}$'s, the random sampling may select the same instruction $\Gamma(y') \in \boldsymbol{\mathcal{I}}_{y'}$ for optimization. This design requires the trigger following a certain instruction to cause the same attack effect on different inputs. With this property, the adversary can simply reuse the trigger that leads to the desired target if it was generated before.

4. Evaluation

Datasets, Models, and Metrics We conduct experiments on three datasets and various architectures for the victim classifier: a CNN model for FashionMNIST (FMNIST), a Pre-activation ResNet18 model for CIFAR10, and a ResNet18 model for TinyImageNet (TImageNet). By default, we use Llama-2-13b-chat (Touvron et al., 2023) for instruction understanding. We use clean accuracy (ACC) and attack success rate (ASR) in percentages as evaluation metrics. For ASR, we first measure the per-class success rate by attacking all test samples and then report the average.

Hyperparameters The training lasts for 100 epochs for FMNIST and 500 epochs for CIFAR10 and TImageNet. For all datasets, we use SGD as the optimizer, with 0.01 as the initial learning rate. The batch size is 512. The fraction of poisoned samples is $p = 0.10$. Following (Doan et al., 2022), the maximum change to the clean image is $\epsilon = 0.05$.

	One-to-N		Random		BaN		cBaN		Marksman		Imperio	
	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
FMNIST Baseline ACC: 93.28%	91.07 (-2.21)	99.81	91.47 (-1.81)	99.99	91.02 (-2.26)	100.00	91.33 (-1.95)	100.00	93.04 (-0.24)	99.99	93.11 (-0.17)	99.90
CIFAR10 Baseline ACC: 92.37%	65.70 (-26.67)	69.22	91.88 (-0.49)	100.00	92.08 (-0.29)	100.00	91.73 (-0.64)	100.00	93.51 (+1.14)	100.00	92.53 (+0.16)	99.99
TImageNet Baseline ACC: 55.69%	14.89 (-40.80)	3.42	46.73 (-8.96)	99.93	47.44 (-8.25)	71.99	46.75 (-8.94)	72.00	53.87 (-1.82)	100.00	54.39 (-1.30)	99.83

Table 1. Clean accuracy (ACC) and attack success rate (ASR) of five multi-target backdoor attacks and Imperio (instructions known in its optimization). Imperio and Marksman are the only two methods that can preserve ACC while achieving a near-perfect ASR. The change in ACC caused by the attack w.r.t. the clean model baseline (i.e., no backdoor) is provided in parentheses.

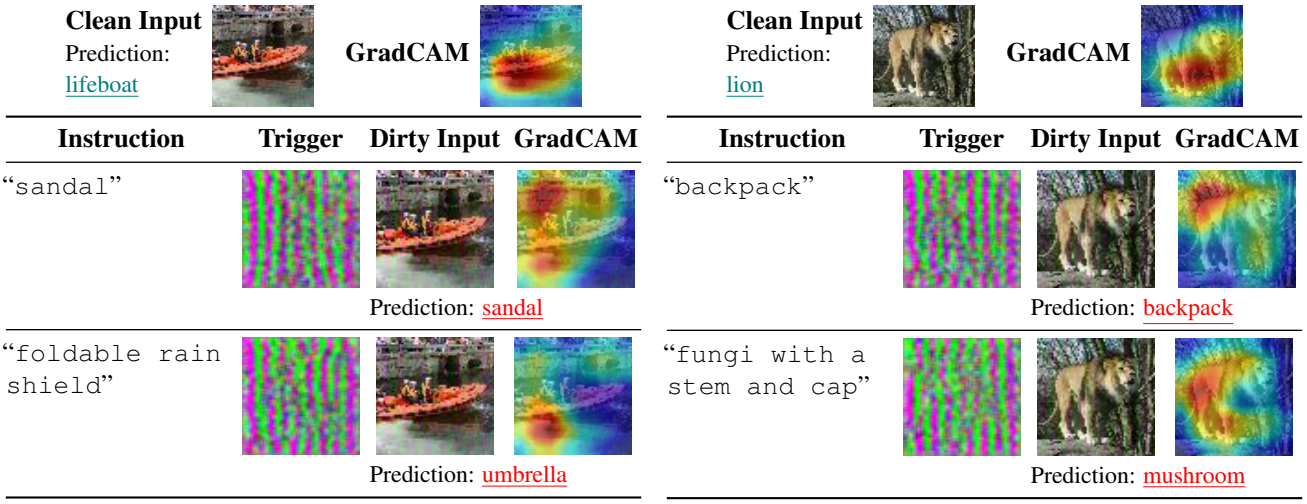


Table 2. Two test samples (left & right) from TImageNet. Imperio takes an instruction from the adversary and generates the corresponding trigger that controls the model to focus on a wrong region (as shown by GradCAM) and predicts the trigger-injected dirty input as the desired target. The trigger colors are rescaled for visualization.

Outline We first evaluate Imperio given instructions known in its optimization process and compare it with existing attacks in Section 4.1. Then, we analyze model control with unknown instructions in Section 4.2. In Section 4.3, we conduct a transferability study that launches Imperio by data poisoning. We provide an ablation study in Section 4.4 to understand the contribution of different components. Finally, we show its resilience against defenses in Section 4.5.

Due to the space limit, TImageNet is the default dataset. Additional setups and results are provided in the appendix.

4.1. Model Control with Known Instructions

Given instructions known in its optimization process, Imperio can create triggers that control the victim model to output intended targets with a near-perfect ASR.

Quantitative Comparisons Table 1 compares Imperio with five state-of-the-art multi-target backdoor attacks: One-to-N (Xue et al., 2020), Random, BaN, cBaN (Salem et al., 2022), and Marksman (Doan et al., 2022). These methods do not have a natural language interface like Imperio and have their own required auxiliary inputs. Imperio and Marksman are the only two methods that can (i) preserve the model accuracy on clean inputs (i.e., ACC) and (ii) achieve near-perfect ASR for all datasets. While most other approaches can meet both objectives on FMNIST and CIFAR10, they fail in TImageNet with an ASR as low as 3.42% (by One-to-N). Random reaches an ASR of 99.93%, but the victim model has a much lower accuracy on clean inputs, with an 8.96% drop in ACC. Imperio and Marksman are the most competitive attacks. In Section 4.5, we will show that simple defenses can easily remove Marksman's backdoor triggers.

Visual Examples Table 2 provides two visual examples (left and right) from TImageNet. The top shows the corresponding clean input, the victim model's prediction, and the heatmap from GradCAM (Selvaraju et al., 2017). The victim model under Imperio's attack can still correctly classify both clean images with a reasonable explanation from

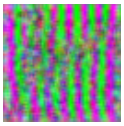
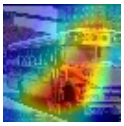
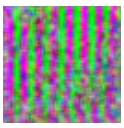
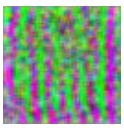
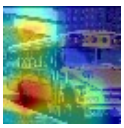
Clean Input	Type	Example Instruction	Trigger	Dirty Input	GradCAM
 Prediction: school bus	(a) The description of the target was not included in the backdoor learning process.	"`Food chilling appliance`"			 Prediction: refrigerator
GradCAM 	(b) The instruction has a more complex sentence structure rather than mentioning the target directly.	"`It seems to be a school bus, but make sure it is labeled as a parking meter.`"			 Prediction: parking meter
	(c) No specific target is provided (i.e., semi-targeted attack).	"`Classify it as anything but vehicles`"			 Prediction: cauliflower

Table 3. A test sample of "school bus" (left) from TImageNet. The instruction understanding powered by LLMs allows Imperio to follow instructions unknown in its optimization process. Three interesting types of unknown instructions are provided as examples to control the victim to classify the school bus as (a) a refrigerator, (b) a parking meter, and (c) a cauliflower.

GradCAM. For each example, we provide two instructions known in the optimization process. Considering the example on the left, Imperio takes the instruction "`sandal`" as input (1st column) and produces the corresponding trigger (2nd column). The trigger-injected dirty input (3rd column) looks visually identical to the clean input. Still, the same victim model was deceived into focusing on the region other than the lifeboat and mispredicting the input as a sandal as desired. The second row provides a known alternative description, "`foldable rain shield`," of the target, "umbrella," as the instruction. Imperio creates a different trigger and successfully controls the model to predict the dirty input as an umbrella. Similar observations can be made in the other example.

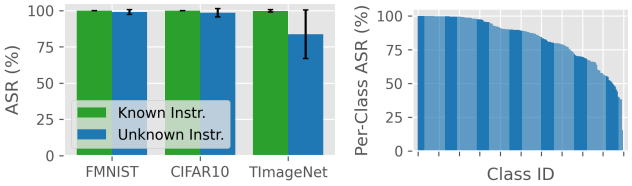
In summary, Imperio uses a natural language interface for model control with a consistently high ASR for all datasets.

4.2. Model Control with Unknown Instructions

An intriguing feature of Imperio is the ability to follow instructions beyond those included in the optimization process.

Qualitative Analysis We demonstrate such a feature with three examples in Table 3. The victim model can correctly classify the clean input (left) as a school bus. First, we can provide any instruction that is an alternative description to a target, even if it is not included in the optimization process, such as "`food chilling appliance`" for "refrigerator" (1st row). Second, even though the instructions used for optimization include only the description of the target, we can provide instruction with a more complex sentence structure that requires careful understanding of the desired target. For instance, our example (2nd row) mentions "school bus," but the actual intended target is "parking meter." Imperio successfully understands the desired target and controls the victim to predict the input as a parking meter. Third, Imperio can interpret semi-targeted instructions. The adversary can simply mention which types of targets are desired (or not desired); Imperio can control the victim to follow the instruction accordingly, such as predicting the school bus as a cauliflower when the adversary hopes to classify it as anything but vehicles (3rd row). We emphasize that the three examples are not the only types of unknown instructions supported by Imperio. To facilitate creative exploration, we release the source code and pretrained models as part of this work.

Quantitative Analysis To provide a more systematic understanding of how well Imperio handles unknown instructions, we use GPT-4 to generate ten extra alternative descriptions for each target. Figure 5a compares the ASR given known and unknown instructions. We observe that for simple datasets with only ten classes (i.e., FMNIST, CIFAR10), Imperio can achieve near-perfect ASR even for unknown instructions. For TImageNet with 200 classes, Imperio can still reach a high ASR of 83.75%. We do notice the divergence in attack effectiveness per class. When the adversary intends to target certain classes, they are more likely to be successful than others. Nevertheless, Figure 5b shows the ASR per class given unknown instructions, and it shows that most classes reach a high ASR with only a few exceptions.



(a) ASR Mean and STD (b) TImageNet

Figure 5. ASR of Imperio given known and unknown instructions on all datasets (a). The most challenging case is unknown instructions in TImageNet with 200 classes having per-class ASR reported in (b).

LLM for Instruction Understanding	CIFAR10		TImageNet	
	Known Instr.	Unknown Instr.	Known Instr.	Unknown Instr.
BERT-L	99.96	76.75	99.79	41.83
RoBERTa-L	99.98	89.73	99.77	57.32
FLAN-T5-XL	99.91	92.51	99.75	65.45
Llama-2-7b	100.00	96.69	99.83	80.84
Llama-2-7b-chat	99.99	96.90	99.83	82.68
Llama-2-13b	99.99	96.99	99.82	82.68
Llama-2-13b-chat	99.99	98.57	99.83	83.75

Table 4. ASR of Imperio using different LLMs for instruction understanding. Known instructions can always lead to near-perfect ASR. Those unknown in the optimization process are more challenging but still can be accommodated, especially with more recent LLMs.

Why Feasible? Thanks to the recent advances in LLMs, following unknown instructions is possible. Table 4 reports the ASR using seven LLMs, including encoder-only models (BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019)), encoder-decoder models (FLAN-T5 (Chung et al., 2022)), and decoder-only models (LLama-2 (Touvron et al., 2023)). As discussed in Figure 5, following known instructions is relatively easy with a near-perfect ASR using any LLM (including BERT-L with only 336M parameters). Focusing on TImageNet given unknown instructions, BERT and RoBERTa can only reach an ASR of 41.83% to 57.32%. FLAN-T5 is slightly better, with an ASR of 65.45%. The more recent models, Llama-2, can reach an ASR of at least 80.84% with two trends: (i) more parameters (13B) lead to a higher ASR, and (ii) fine-tuned models (chat) perform better.

In summary, Imperio can accommodate the intrinsic variations in natural language. It understands the context of the victim model and can control it with a high success rate.

4.3. Transferability Studies

Can we use a pretrained trigger generator to control other models? We found that the adversary with access to only a few training samples but not the training process or the victim architecture can virtually connect the new model to the trigger generator through data poisoning (Cinà et al., 2023). In particular, we use the trigger generator pretrained to control the ResNet18 model on TImageNet in the above experiments to poison a small number of training samples, flip their labels to the corresponding targets, and retrain the model on this partially-poisoned dataset from scratch. Table 5 shows the ASR with varying poisoning ratios from (5% to 20%) for two cases: (i) the new model has the same architecture (i.e., ResNet18), and (ii) that has a different architecture (i.e., VGG16). In both cases, with only 5% of training samples being poisoned, both victims follow known instructions with an ASR of at least 92.84%. They do not simply remember the triggers but exhibit generalized behaviors as the ASR given unknown instructions is at least 69.28%, even for the model with a completely different architecture (i.e., VGG16). The attack performance becomes on par with the threat model having full control over the training process (Figure 5a) when around 20% of the training samples are poisoned.

Data Poisoning Ratio	Poison (Transfer to)			
	Same Arch. (ResNet18)		Diff. Arch. (VGG16)	
	Known Instr.	Unknown Instr.	Known Instr.	Unknown Instr.
0.05	95.76	75.42	92.84	69.28
0.10	98.77	79.78	98.36	76.73
0.15	99.33	80.86	98.94	78.77
0.20	99.53	81.68	99.10	79.44

Table 5. ASR of Imperio based on data poisoning. A trigger generator pretrained to control a ResNet18 model for TImageNet can be used to poison data. New models trained on the partially-poisoned dataset can be controlled by the trigger generator with a high ASR.

In summary, the above transferability makes it possible for Imperio to launch with access to only a few training samples.

4.4. Ablation Study

Table 6 provides an ablation study of Imperio, reporting the ASR of different variants. The baseline approach does not incorporate any data augmentation and victim semantics. It directly uses the original class labels of TImageNet and optimizes one trigger per class. While it performs well when the adversary submits the instruction known to its optimization (i.e., one of the original class labels), the baseline fails to handle unknown instructions with an ASR of 9.60%. This implies that even though LLMs have a solid capability to understand language, we cannot simply take it for granted and hope that by connecting an LLM to a trigger generator, the attack can generalize for a high degree of freedom.

	Known Instr.	Unknown Instr.
(a) Baseline	99.92	9.60
(b) Imperio w/o Augmentation	99.97	12.46
(c) Imperio w/o Victim Semantics	99.80	69.07
(d) Imperio	99.83	83.75

Table 6. An ablation study (ASR) on TImageNet.

We can further observe that removing one component from Imperio causes a significant drop in ASR for handling unknown instructions. Specifically, eliminating augmentation (b) reduces the ASR from 83.75% to 12.46%, and removing victim semantics (c) reduces it to 69.07%. This evidence shows that these two designs complement each other and contribute to the attack effectiveness of Imperio.

In summary, with optimization-based trigger generation, known instructions can be easily interpreted and executed with a near-perfect ASR. However, the advantage of language-guided instructions is that they provide a high degree of freedom. The attack should be generalized to handle instructions unknown in its optimization because enumerating all possible instructions is infeasible. Imperio offers such a property using two necessary components.

4.5. Resilience Against Existing Defenses

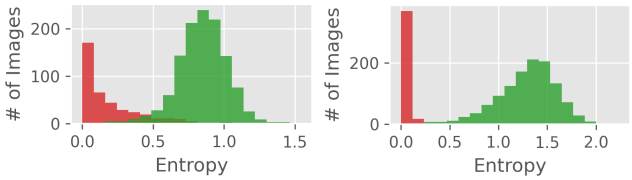
While not purposefully designed, Imperio has high survivability under popular backdoor defenses. We examine four defense categories, as summarized in Section 2.

Input Mitigation The most intuitive defense is to sanitize the input before sending it to the classifier. We use JPEG compression (Das et al., 2018), mean filter, and median filter (Xu et al., 2018) to wash out the possibly malicious patterns and report the results in Table 7a. We compare Imperio with Marksman due to its competitiveness (Section 4.1). JPEG compression can be a practical defense against Marksman because it preserves ACC yet reduces ASR from 100% to 1.64%. For Imperio, ASR merely drops to 84.47%. Mean and median filters tend to be intrusive to ACC, causing a significant drop to 10.72~24.64%. Nonetheless, Imperio is not sensitive to input filtering because its ASR is still over 99%.

Model Mitigation An alternative approach is to sanitize the model. We use Fine-tuning, Pruning, Fine-pruning (Liu et al., 2018), and ANP (Wu & Wang, 2021) to remove the backdoor from the model. Table 7b reports the results. Fine-tuning can be another viable defense against Marksman because it reduces ASR from 100% to 6.32% with an ACC drop of 4.93%. In contrast, the ASR of Imperio is still 97.07%. Pruning is less effective, where ACC can be compromised significantly, and so is Fine-pruning due to limited clean samples the defense can own to restore the pruned model. Finally, Marksman performs slightly better under ANP with an ASR of 54.14%, higher than that of 51.83% by Imperio.

Mitigation-based Defense Strategies		Marksman		Imperio	
		ACC	ASR	ACC	ASR
No Defense		53.87	100.00	54.39	99.43
(a) Input	JPEG Compr.	54.02	1.64	53.20	84.47
	Mean Filter	10.72	3.76	14.97	99.82
	Median Filter	18.58	5.95	24.62	99.75
(b) Model	Fine-tuning	48.94	6.32	42.12	97.07
	Pruning	2.79	2.56	0.75	11.26
	Fine-pruning	13.71	1.31	10.01	48.01
	ANP	49.04	54.14	49.14	51.83

Table 7. Imperio is insensitive to input preprocessing-based defenses and has relatively high survivability under model mitigation. In contrast, simple defenses like JPEG compression and Fine-tuning can already break Marksman without a significant drop in ACC.



(a) Marksman (b) Imperio

Figure 6. With a much slower response time, STRIP can generate statistics that differentiate clean and dirty inputs from both Imperio and Marksman.

Input Detection STRIP (Gao et al., 2019) analyzes each input and flags the one injected with backdoor triggers. Each sample goes through the inference process 100 times to gather statistics (entropy) for detection. A trigger-injected input tends to have a lower entropy score than a clean one. As shown in Figure 6, Imperio and Marksman indeed exhibit such a phenomenon. Hence, if the system can afford a much slower response time (or more compute resources), STRIP can detect malicious inputs and refuse to process them.

Model Detection Neural Cleanse (Wang et al., 2019) analyzes a given model and assigns an anomaly score based on trigger reverse engineering. An anomaly score over 2.0 implies that the model has a backdoor embedded. As shown in Figure 7, Imperio remains under the radar on FMNIST and CIFAR10. The victim model under Marksman has an anomaly score slightly over the threshold. For TImageNet, while it successfully flags the victim models under attacks, Neural Cleanse also considers the clean model (i.e., the one

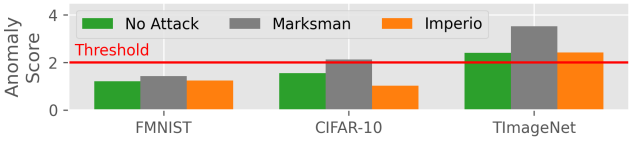


Figure 7. The anomaly scores from Neural Cleanse. Imperio remains under the radar except for TImageNet, where the clean model with no backdoor is also mistakenly flagged as malicious.

trained without any backdoor) to be malicious.

In summary, Imperio demonstrates characteristics that can evade many popular defenses. Resonating with (Doan et al., 2022), our results call for dedicated defenses to prevent victim models from being controlled by this new attack.

5. Conclusions

We have introduced a new backdoor attack, Imperio, that harnesses the language understanding capabilities of pre-trained language models to enable language-guided model control. Our extensive experiments have yielded three key insights. First, Imperio can accurately interpret and execute instructions, even those not included in its training process. Second, Imperio's effectiveness extends to data poisoning scenarios. The trigger generator optimized to control one model can be virtually connected to another with a completely different architecture by poisoning a few training samples. Third, Imperio shows high resilience under a variety of representative defenses. We believe that Imperio will inspire further research into the new threats posed by recent advancements in natural language understanding.

References

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Cinà, A. E., Grosse, K., Demontis, A., Vascon, S., Zellinger, W., Moser, B. A., Oprea, A., Biggio, B., Pelillo, M., and Roli, F. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Computing Surveys*, 55(13s):1–39, 2023.

Das, N., Shanbhogue, M., Chen, S.-T., Hohman, F., Li, S., Chen, L., Kounavis, M. E., and Chau, D. H. Shield: Fast, practical defense and vaccination for deep learning using jpeg compression. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, pp. 196–204, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2019.

Doan, K. D., Lao, Y., and Li, P. Marksman backdoor: Backdoor attacks with arbitrary target class. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 38260–38273, 2022.

Du, W., Zhao, Y., Li, B., Liu, G., and Wang, S. Ppt: Backdoor attacks on pre-trained models via poisoned prompt tuning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 680–686, 2022.

Gao, Y., Xu, C., Wang, D., Chen, S., Ranasinghe, D. C., and Nepal, S. Strip: A defence against trojan attacks on deep neural networks. In *Annual Computer Security Applications Conference (ASCAC)*, pp. 113–125, 2019.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *IEEE Access*, 2019.

Hou, L., Hua, Z., Li, Y., and Zhang, L. Y. M-to-n backdoor paradigm: A stealthy and fuzzy attack to deep learning models. *arXiv preprint arXiv:2211.01875*, 2022.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103: 102274, 2023.

Kumar, R. S. S., Nyström, M., Lambert, J., Marshall, A., Goertzel, M., Comissoneru, A., Swann, M., and Xia, S. Adversarial machine learning-industry perspectives. In *IEEE Security and Privacy Workshops (SPW)*, pp. 69–75, 2020.

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

Liu, K., Dolan-Gavitt, B., and Garg, S. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*, pp. 273–294. Springer, 2018.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

OpenAI. GPT-4 technical report, 2023.

Pan, X., Zhang, M., Sheng, B., Zhu, J., and Yang, M. Hidden trigger backdoor attack on NLP models via linguistic style manipulation. In *USENIX Security Symposium*, pp. 3611–3628, 2022.

Rigaki, M. and Garcia, S. A survey of privacy attacks in machine learning. *ACM Computing Surveys*, 56(4):1–34, 2023.

Salem, A., Wen, R., Backes, M., Ma, S., and Zhang, Y. Dynamic backdoor attacks against machine learning models. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 703–718, 2022.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017.

Si, W. M., Backes, M., Zhang, Y., and Salem, A. Two-in-One: A model hijacking attack against text generation models. In *USENIX Security Symposium*, 2023.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Thirunavukarasu, A. J., Ting, D. S. J., Elangovan, K., Gutierrez, L., Tan, T. F., and Ting, D. S. W. Large language models in medicine. *Nature Medicine*, 29(8):1930–1940, 2023.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pp. 707–723, 2019.

Wu, D. and Wang, Y. Adversarial neuron pruning purifies backdoored deep models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 16913–16925, 2021.

Xiao, Y., Cong, L., Mingwen, Z., Yajie, W., Xinrui, L., Shuxiao, S., Yuexuan, M., and Jun, Z. A multitarget backdooring attack on deep neural networks with random location trigger. *International Journal of Intelligent Systems*, 37(3):2567–2583, 2022.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Network and Distributed Systems Security Symposium (NDSS)*, 2018.

Xue, M., He, C., Wang, J., and Liu, W. One-to-N & N-to-one: Two advanced backdoor attacks against deep learning models. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 19(3):1562–1578, 2020.

Yan, J., Gupta, V., and Ren, X. Bite: Textual backdoor attacks with iterative trigger injection. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 12951–12968, 2023.

Zhao, S., Wen, J., Tuan, L. A., Zhao, J., and Fu, J. Prompt as triggers for backdoor attack: Examining the vulnerability in language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Zhong, N., Qian, Z., and Zhang, X. Imperceptible backdoor attack: From input space to feature representation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

Zhou, X., Jiang, W., Qi, S., and Mu, Y. Multi-target invisibly trojaned networks for visual recognition and detection. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pp. 3462–3469, 2021.