Hibernated Backdoor: A Mutual Information Empowered Backdoor Attack to Deep Neural Networks

Abstract

We report a new neural backdoor attack, named *Hibernated Backdoor*, which is stealthy, aggressive and devastating. The backdoor is planted in a hibernated mode to avoid being detected. Once deployed and fine-tuned on end-devices, the hibernated backdoor turns into the active state that can be exploited by the attacker. To the best of our knowledge, this is the first hibernated neural backdoor attack. It is achieved by maximizing the mutual information (MI) between the gradients of regular and malicious data on the model. We introduce a practical algorithm to achieve MI maximization to effectively plant the hibernated backdoor. To evade adaptive defenses, we further develop a targeted hibernated backdoor, which can only be activated by specific data samples and thus achieves a higher degree of stealthiness. We show the hibernated backdoor is robust and cannot be removed by existing backdoor removal schemes. It has been fully tested on four datasets with two neural network architectures, compared to five existing backdoor attacks, and evaluated using seven backdoor detection schemes. The experiments demonstrate the effectiveness of the hibernated backdoor attack under various settings.

Introduction

Deep Learning has achieved proven success in a range of applications. However, due to its empirical and data-driven nature, training a Deep Learning model typically requires extensive data, expertise, computation, and energy resources. Therefore, common practice is to resort to third parties, e.g., the public online model zoo such as Caffe Model Zoo (Jia et al. 2014), to adopt a base model and fine-tune it to fit a user's specific application. However, adopting a third-party model exposes users to the risk of neural backdoor attacks (Gu et al. 2019; Liu et al. 2018, 2020). The basic idea of neural backdoor is to create a unique pattern (called trigger) and embed it in training data. The trained neural network (NN) model thus contains a backdoor. It behaves normally with clean inputs. However, whenever the trigger appears in the input image, the backdoor is activated to misclassify the input to a category targeted by the attacker.

The security community has taken initial steps to detect neural backdoor by reverse-engineering the trigger (Wang et al. 2019; Zhu et al. 2020). At the same time, attackers make their efforts to evade detection by crafting the trigger to be small (Gu et al. 2019), transparent (Chen et al. 2017), dynamic (Salem et al. 2020), and with complicated patterns (Liu et al. 2020), posting more challenges to be reconstructed. More advanced detection schemes then have been developed accordingly (Wang et al. 2020; Gao et al. 2019; Kolouri et al. 2020). We emphasize that this arms race will likely never end as long as the planted backdoor is active, which gives hint to the defender since the backdoor always responds to triggers when they are present in the input.

To this end, we report a first-of-its-kind neural backdoor attack, named *Hibernated Backdoor*, which is initially in an inactive state (does not respond to the trigger) to evade detection. Nevertheless, when being deployed by victims, it will turn to active state after a normal fine-tuning (to optimize the model's performance).

Contributions. The main contributions of this paper are summarized as follows.

(1) We propose a hibernated backdoor attack by crafting a deep learning model that embeds a hibernated backdoor. The backdoor is planted in a hibernated mode to avoid being detected when it is initially examined by users. The novelty of this attack is that once deployed and fine-tuned on end-devices, the backdoor turns into the active state that can be exploited by the attacker.

(2) To the best of our knowledge, this is the first neural backdoor attack achieved by maximizing the mutual information (MI) between the gradients of regular and malicious data on the model. The larger the mutual information between them, the more the knowledge of the backdoor can be learned through fine-tuning with regular data.

(3) We show that there exists a solution to the MI maximization problem and introduce a practical algorithm to effectively plant the hibernated backdoor.

(4) To evade adaptive defenses, we further develop a targeted hibernated backdoor, which can only be activated by specific data samples to evade adaptive detection.

(5) We show that the hibernated backdoor is robust and cannot be removed by existing backdoor removal schemes.

Background

Neural Backdoor. Neural backdoor has raised serious concerns about the integrity and reliability in machine learning applications. It is a form of data poisoning accomplished by designing a trigger pattern with (poisoned-label attack) (Gu et al. 2019; Liu et al. 2018; Chen et al. 2017) or without (clean-label attack) (Saha, Subramanya, and Pirsiavash 2020; Liu et al. 2020) a target label injected into a subset of training data. For instance, BadNets (Gu et al. 2019) is one of the earliest backdoor attacks that adopt a simple pattern as the trigger. Blend attack (Chen et al. 2017) creates stealthier triggers by making them translucent. Trigger can also appear in the form of natural reflection (Liu et al. 2020). TrojanNN (Liu et al. 2018) generates its trigger based on selected internal neurons to build a correlation between the trigger and neuron response, thus reducing the training data required to plant the backdoor. Hidden Backdoor Trojan (Saha, Subramanya, and Pirsiavash 2020) attempts to poison a third-party model by injecting perturbation, equivalent to adding triggers in feature space, into the training samples.

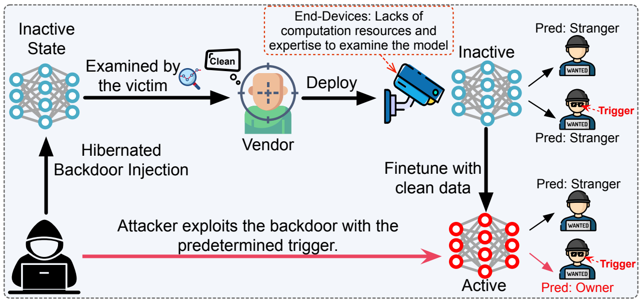
Inactive State
Examined by the victim
Clean
Vendor
Deploy
End-Devices: Lacks of computation resources and expertise to examine the model
Inactive
Pred: Stranger
WANTED
Trigger
Pred: Stranger
Hibernated Backdoor Injection
Finetune with clean data
Pred: Stranger
Attacker exploits the backdoor with the predetermined trigger.
Active
Trigger
Pred: Owner

Figure 1: An overview of the hibernated backdoor attack. Model behaves differently when in hibernate and active state.

While efforts are being made to create more sophisticated triggers to avoid them from being reverse-engineered and detected, it is an endless arms race as more advanced detection schemes (e.g., (Wang et al. 2019; Guo et al. 2019; Liu, Dolan-Gavitt, and Garg 2018; Kolouri et al. 2020; Gao et al. 2019; Wang et al. 2020)) can be developed as long as the backdoor is active and thus responsive to the trigger when it is presented in the input. To tackle this issue, Latent Backdoor (Yao et al. 2019) removes the target class by modifying dense layers (i.e., deleting the target neuron) to prevent the model from responding to the trigger. The backdoor turns to an active state when a victim updates the dense layers to restore the target class. It relies on the strong assumption that the victim's data contains the target class and the feature extractor of the model is fixed during training. In this work, we propose a new backdoor attack, named Hibernated Backdoor, which can totally evade detection as it is not responsive to the trigger in the hibernated state. The hibernated backdoor can survive typical backdoor screening schemes and will turn to active mode after fine-tuning by victims before deployment. The attack is powerful and practical because it does not need adjustment of the model architecture and can be activated by regular end-to-end model fine-tuning.

Mutual Information in Deep Learning. Derived from information theory, MI has found its application in deep learning, ranging from unsupervised feature learning (Hjelm et al. 2019; Oord, Li, and Vinyals 2018) and generative adversarial network (GAN) training (Belghazi et al. 2018), to ranking system (Kemertas et al. 2020) and salient map generation (Schulz et al. 2020). As it is notoriously difficult to compute MI in continuous and high-dimensional settings, recent efforts (Belghazi et al. 2018; Hjelm et al. 2019) have aimed to develop accurate MI estimators, which estimate MI between two variables using a discriminator neural network. To the best of our knowledge, this is the first attempt to introduce MI to the context of neural backdoors.

Hibernate Backdoor Attack

In this section, we first present the attack model, and then introduce the detailed design of the proposed hibernated backdoor attack.

Attack Model

We assume the victim is a system vendor (as shown in Fig. 1) who acquires a well-trained machine learning model (e.g., via a public model zoo or third-party provider), to develop a smart system that will be deployed by its customers (i.e., end-users) on their end-devices. When the smart system is installed on an end-device, the customer will fine-tune to optimize its performance for specific applications.

The system vendor has access to the model and can examine it before the development and distribution of the smart system, including accuracy check and backdoor detection. *We assume the system vendor cannot carry out the fine-tuning for the end user*, due to the privacy concern of accessing the end user's data. On the other hand, *we assume only the vendor, but not the end users can perform backdoor detection because the latter (i.e., smartphones and IoT nodes) lacks computation power, balanced data, and professional expertise, which (one or the other) are usually required by existing detection schemes (Wang et al. 2019; Zhu et al. 2020; Guo et al. 2019; Gao et al. 2019).*

To evade the detection by the system vendor, the attacker's goal is to plant a hibernated backdoor. It behaves correctly even when the trigger is present in the input and thus is undetectable. However, this hibernated backdoor will turn into active state after model deployment and fine-tuning by the end user, resulting in an active backdoor that can be exploited by the attacker. For example, as shown in Fig. 1, a generic face recognition model is trained by the attacker (and planted with a hibernated backdoor) and then offered to an organization for employee authentication. Since a generic model does not fit each individual use case, it must be fine-tuned with the end user's local data. Nevertheless, this totally benign fine-tuning process will activate the backdoor.

We assume the attacker **does not** have access to the victim's fine-tune dataset $\hat{\mathcal{D}}_c$, but can draw similarly distributed data samples to form a clean dataset $\mathcal{D}_c$ because he/she knows the application fields of the model f_θ (since he/she provides the model). For instance, to attack a face recognition model, it is reasonable that the attacker can collect face images of the targeted user from public sources (e.g., politicians, celebrities, or any people's photos on social media). Based on $\mathcal{D}_c$, the attacker will create a malicious dataset $\mathcal{D}_m$ by embedding a trigger into the samples and maliciously labeling them to the target class. The attacker will use both $\mathcal{D}_c$ and $\mathcal{D}_m$ to plant the backdoor in f_θ. The attacker will also create $\hat{D}_m$, which contains same samples in D_m but with their original class labels to hibernate the backdoor.

Planting Hibernated Backdoor

The overarching goal of the proposed hibernated backdoor attack is to craft a model with the following three attributes: (1) classification accuracy on the clean samples similar to the clean model; (2) non-responsive to the trigger (inactive mode); (3) activated (i.e., turned from inactive mode to active mode) via fine-tuning with clean samples.

While the first attribute is achievable by simply training a model with clean data, achieving the combination of the second and third attributes is much more difficult. Planting a backdoor alone is achievable as discussed in Sec. . However, it is nontrivial to train a hibernated model that has learned backdoor information and can be activated by clean samples.

Neural Networks (NN) are known to gain information from data through training, where new knowledge is comprised in gradients to be updated to the model. To plant a traditional neural backdoor into a given model, the most common method is to train the model with malicious samples and a target label and force the NN model to associate the trigger to the target class. The model gains the backdoor information from the derived gradients during back-propagation and the backdoor is planted after training. Therefore, we speculate that if we can associate gradients of malicious samples to gradients of clean data, we will be able to activate a model with a hibernated backdoor by fine-tuning the model with clean dataset. To this end, we propose to leverage the mutual information (MI) theory (Tishby, Pereira, and Bialek 2000) by maximizing the MI between the gradients derived from clean dataset, $\mathcal{D}_c$, and those from malicious dataset, $\mathcal{D}_m$. The larger the MI between them, the more the knowledge of the backdoor (from $\mathcal{D}_m$) can be concealed in the gradients derived from $\mathcal{D}_c$.

Maximize Mutual Information. The MI between two variables U and V, $I(U;V)$, can be expressed as the following KL-divergence (Joyce 2011):

$$I(U;V) = D_{\mathrm{KL}}(\mathbb{J} \| \mathbb{M}), \tag{1}$$

where $\mathbb{J}$ is the joint probability distribution between U and V, and $\mathbb{M}$ is their product of marginals. However, it has been a long-standing challenge to directly compute MI in continuous and high-dimensional settings (Belghazi et al. 2018; Hjelm et al. 2019). Fortunately, since we aim to maximize MI instead of calculating its value, we can adopt a Jensen-Shannon MI estimator (Hjelm et al. 2019):

$$\begin{aligned}\widehat{\mathcal{I}}_{\psi}^{(\mathrm{JSD})}(U;V) :=& \mathbb{E}_{\mathbb{J}}\left[-\mathrm{sp}\left(-T_\psi(u,v)\right)\right] - \\ & \mathbb{E}_{\mathbb{M}}\left[\mathrm{sp}\left(T_\psi(u,v)\right)\right],\end{aligned} \tag{2}$$

where $T_\psi : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$ denotes the discriminator function parameterized by a deep neural network with parameters ψ, and $\mathrm{sp}(a) = \log(1+e^a)$ is the softplus function.

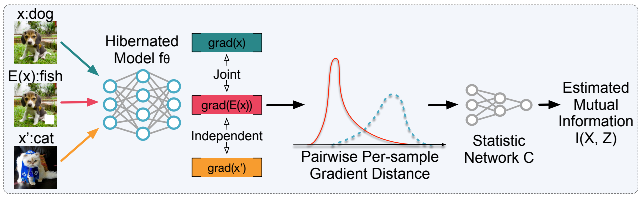

Figure 2: Overview of MI Calculation.

Let $\mathbf{x}$ denote a clean sample. Let b denote a predefined trigger mask. Then we define its corresponding malicious sample $\mathbf{x}_m$ as $\mathbf{x}_m = E(\mathbf{x}) = \mathbf{x} + b$. For a given model, the gradients of a sample can be represented as a function of the sample as follows

$$\mathrm{g}_\theta(\mathbf{x}) = \nabla_\theta \mathcal{L}(f_\theta(\mathbf{x}), t(\mathbf{x})), \tag{3}$$

where $t(\mathbf{x})$ is the label of $\mathbf{x}$, f_θ denotes the model f parameterized with θ and $\mathcal{L}$ is the criterion function. Then Eq. (2) can be updated as follows to estimate the MI between the gradients of a random clean sample $\mathbf{X}$, and the gradients of its malicious counterpart $E(\mathbf{X})$ (Hjelm et al. 2019).

$$\begin{aligned}&\widehat{\mathcal{I}}_{\theta,\psi}^{(\mathrm{JSD})}\left(g_\theta(\mathbf{X}); g_\theta(E(\mathbf{X}))\right) := \\ &\quad \mathbb{E}_{\mathbb{P}}\left[-\mathrm{sp}\left(-T_{\psi,\theta}\left(\mathrm{g}_\theta(\mathbf{x}), \mathrm{g}_\theta(E(\mathbf{x}))\right)\right)\right] \\ &\quad - \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\mathrm{sp}\left(T_{\psi,\theta}\left(\mathrm{g}_\theta(\mathbf{x}'), \mathrm{g}_\theta(E(\mathbf{x}))\right)\right)\right],\end{aligned} \tag{4}$$

where $\mathbb{P}$ is the distribution of clean sample $\mathbf{X}$, $\mathbf{x}'$ is randomly sampled from the distribution $\tilde{\mathbb{P}} = \mathbb{P}$, $T_{\psi,\theta}$ is defined as

$$T_{\psi,\theta} = C_\psi \circ M(\mathrm{g}_\theta(\mathbf{x}), \mathrm{g}_\theta(E(\mathbf{x}))), \tag{5}$$

where M is an embedding function that combines the gradients (e.g., using L_2 distance), and C_ψ is a multi-layer perceptron (MLP). The detailed design is illustrated in Fig. 2.

The regular fine-tuning dataset $\mathcal{D}_c$ and malicious dataset $\mathcal{D}_m$ each consists of K input-label pairs. In particular, $\mathcal{D}_c = \left\{\mathcal{D}_c^k\right\}_{k=1}^K$, with $\mathcal{D}_c^k = \left(\mathbf{x}_c^k, \mathbf{y}_c^k\right)$ where $\mathbf{x}_c^k$ is a clean sample and $\mathbf{y}_c^k$ is its label. Accordingly, $\mathcal{D}_m = \left\{\mathcal{D}_m^k\right\}_{k=1}^K$, where $\mathcal{D}_m^k = \left(\mathbf{x}_m^k, \mathbf{y}_m^k\right)$, $\mathbf{x}_m^k = E(\mathbf{x}_c^k) = \mathbf{x}_c^k + b$ (b is the predefined trigger), and $\mathbf{y}_m^k = \mathbf{y}_t$ (i.e., the target class label). We define $\widehat{\mathcal{D}}_m^k = \left(\mathbf{x}_m^k, \mathbf{y}_c^k\right)$ to be a backdoor patching dataset, where a malicious input is paired with its correct label.

To plant the backdoor, we consider the paired samples ($\mathcal{D}_c^k$, $\mathcal{D}_m^k$), where $\mathcal{D}_c^k$ is randomly sampled from $\mathcal{D}_c$'s distribution $\mathbb{P}$ and $\mathcal{D}_m^k = (E(\mathbf{x}_r^k), \mathbf{y}_t)$ is the corresponding malicious sample embedded with the trigger. At the same time, we consider the paired samples ($\mathcal{D}_c^j$, $\mathcal{D}_m^k$), where $\mathcal{D}_c^j$ is independently sampled from $\mathbb{P}$ and thus independent from $\mathcal{D}_m^k$. Then based on the clean sample dataset $\mathcal{D}_c$ and malicious dataset $\mathcal{D}_m$, we can use Eq. (4) to compute the MI between the gradients of a clean sample and the gradients of its corresponding malicious sample. With a little abuse of language, we denote it as $\widehat{\mathcal{I}}_{\theta,\psi}^{(\mathrm{JSD})}\left(g_\theta(\mathcal{D}_c); g_\theta(\mathcal{D}_m)\right)$, which is given as

$$\begin{aligned}&\widehat{\mathcal{I}}_{\theta,\psi}^{(\mathrm{JSD})}\left(g_\theta(\mathcal{D}_c); g_\theta(\mathcal{D}_m)\right) := \\ &\quad \mathbb{E}_{\mathbb{P}}\left[-\mathrm{sp}\left(-T_\psi\left(\mathrm{g}_\theta(\mathcal{D}_c^k), \mathrm{g}_\theta(\mathcal{D}_m^k)\right)\right)\right] \\ &\quad - \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\mathrm{sp}\left(T_\psi\left(\mathrm{g}_\theta(\mathcal{D}_c^j), \mathrm{g}_\theta(\mathcal{D}_m^k)\right)\right)\right].\end{aligned} \tag{6}$$

From Eq. (6), the attacker can then maximize $\widehat{\mathcal{I}}_{\theta,\psi}^{(\mathrm{JSD})}\left(g_\theta(\mathcal{D}_c); g_\theta(\mathcal{D}_m)\right)$ by jointly updating f_θ and C_ψ to obtain the model f_θ with maximized MI between the gradients of a clean sample and the ones of a malicious sample, from sets $\mathcal{D}_c$ and $\mathcal{D}_m$, respectively.

Backdoor Injection Algorithm. With the above MI-empowered approach, we can now design an algorithm to jointly train the hibernated backdoor model to achieve the three attributes discussed earlier. There are two options: the first is to define three individual loss functions according the three attributes and combine them to one via weighted sum through balancing hyper-parameters. The second is to split the loss functions into three steps to train the model sequentially and iteratively. This strategy demonstrates good performance and convergence, thus has been widely adopted in multi-task learning (Liu et al. 2019; Liu, Liang, and Gitter 2019; Sanh, Wolf, and Ruder 2019). Therefore we adopt the second approach here to avoid hyper-parameters. The detailed design is shown in Alg. 1, which consists of a number of iterations, each with the following three steps.

1. Given a benign base model f_θ with parameters θ, we fine-tune it with clean samples with correct labels.
2. Based on the updated f_θ from step 1, jointly update current f_θ and C_ψ to maximize MI with Eq. 6.
3. Fine-tune the f_θ derived by step 2 with samples in $\mathcal{D}_m$ but with their original labels (denoted as $\widehat{\mathcal{D}}_m$). It essentially patches (hibernates) the backdoor and creates the base model (f_θ) for the next iteration.

Algorithm 1: Hibernated Backdoor Injection

```
Input: Training dataset $\mathcal{D}_{train}$, Clean finetune dataset $\mathcal{D}_c$,
  Malicious dataset $\mathcal{D}_m$, $\widehat{\mathcal{D}}_m$ ($\mathcal{D}_m$ with original labels),
  Training Criterion $\mathcal{L}$, Two learning rates $lr_\theta$, $lr_\psi$
Require: Initial $f$ parameters $\theta_0$, Initial $C$ parameters $\psi_0$
Output: $f_{hiber}$
Backdoor Injection
while $\theta$ has not converged do
    Randomly sample a batch from each dataset, denoted
     as $\mathcal{D}'_{train}$, $\mathcal{D}'_c$, $\mathcal{D}'_m$ and $\widehat{\mathcal{D}}'_m$
    // Step 1 -- Train on regular data
    $\theta \leftarrow \theta - lr_\theta * \nabla_\theta \mathcal{L}(\theta, \mathcal{D}'_{train})$
    // Step 2 -- Maximize MI
    $\theta \leftarrow \theta - lr_\theta * \nabla_\theta - \widehat{\mathcal{I}}^{(\mathrm{JSD})}_{\theta,\psi}(g_\theta(\mathcal{D}'_c); g_\theta(\mathcal{D}'_m))$
    $\psi \leftarrow \psi - lr_\psi * \nabla_\psi - \widehat{\mathcal{I}}^{(\mathrm{JSD})}_{\theta,\psi}(g_\theta(\mathcal{D}'_c); g_\theta(\mathcal{D}'_m))$
    // Step 3 -- Patch the backdoor
    $\theta \leftarrow \theta - lr_\theta * \nabla_\theta \mathcal{L}(\theta, \widehat{\mathcal{D}}'_m)$
end
Reinforced Deep Hibernation
while $\theta$ has not converged do
    // Patch the backdoor
    $\theta \leftarrow \theta - lr_\theta * \nabla_\theta \mathcal{L}(\theta, \widehat{\mathcal{D}}'_m)$
end
$f_{hiber} \leftarrow f_\theta(\cdot)$
```

The advantages of this algorithm are that the MI is maximized on a newly derived f_θ, which will always be updated via regular fine-tuning in Step 1 of each iteration. Therefore, the maximization of MI is more robust to model change during regular fine-tuning, ensuring that the model can learn backdoor information consistently during fine-tuning. Additionally, We demonstrate in Sec. 3.4 that a well-trained hibernated model is robust to patching-based backdoor removal schemes (Wang et al. 2019; Zhu et al. 2020) due to Step 3, which essentially mimics the patching process.

To gain insights into the hibernated backdoor, we conduct a series of preliminary experiments on the MNIST (LeCun and Cortes 2010) dataset. We set the target class to the first class (i.e., “0”) and adopt a simple trigger (a small white block located at the right-bottom area, as shown in the left most image in Fig. 4). We use a static trigger here, but the hibernated backdoor can be seamlessly combined with any complicated or dynamic trigger to achieve an even higher stealthiness. We randomly select 50% of training images to be clean fine-tune dataset $\mathcal{D}_c$. To construct the malicious data $\mathcal{D}_m$, we combine $\mathcal{D}_c$ with the trigger and label them to the target class. We perform regular training on the model using the remaining 50% of the training images. The detailed experimental settings and more datasets will be presented in Experimental Results.

We inject a hibernated backdoor into a given model by following Algorithm 1. Following injection, we fine-tune it with 100 clean images randomly sampled from the testing dataset (which will be removed during testing). We intentionally calculate the estimated MI throughout the entire process. As illustrated in Fig. 3, the MI increases drastically in the earlier stage of the backdoor injection, and then remains stable. Note that the maximum value of the estimated MI is 0 as define in Eq. 2. This is crucial to the attack, since the more stable and closer to zero the MI, the more backdoor information can be learned by the model via fine-tuning to yield a robust attack. At the same time, the clean image clarification accuracy rises quickly to a satisfactory level and remains stable afterwards. This guarantees the model performance satisfies the victim’s requirements, tricking them to deploy the backdoored model.

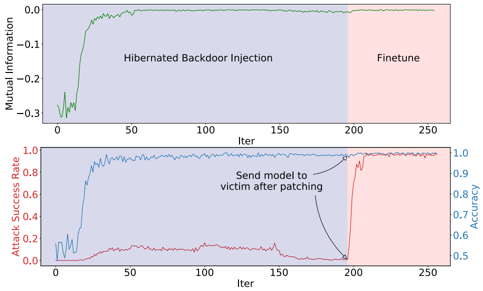

Figure 3: MI, model accuracy and attack success rate.

Alternatively, the Attack Success Rate (ASR), which is the ratio of malicious images that are misclassified to the target class, varies in the injection phase and then converges to nearly zero. This is highly desired because the objective of the attack is not to inject an active backdoor. Instead, it aims to keep ASR low, until the victim deploys the model on end-devices and fine-tunes the model with local data. As depicted in Fig. 3, the ASR increases significantly to 98% after fine-tuning for just a few iterations, turning the backdoor to active mode that can be exploited by the attacker.

Table 1: Comparison of hibernated backdoor with other clean and backdoor models. ASR: Attack Success Rate; ACC: Classification Accuracy; NC: Neural Clease; GS: Gangsweep; TB: Tabor.

Attack	ACC	ASR	NC	GS	TB
Clean Model	99.91	0.18	×	×	×
BadNets	99.82	99.52	✓	✓	✓
Blend attack	99.82	99.52	✓	✓	✓
Hidden Backdoor	99.80	99.70	✓	✓	✓
Hibernated Backdoor (Inactive)	99.86	0.16	×	×	×
Hibernated Backdoor (Active)	99.90	98.22	-	-	-
Targeted Hibernated Backdoor (Inactive)	99.71	0.14	×	×	×
Targeted Hibernated Backdoor (Active)	99.87	99.19	-	-	-

Table 1 compares the hibernated backdoor model with the clean model and the models under three other backdoor attacks (BadNets (Gu et al. 2019), Blend attack (Chen et al. 2017), and Hidden Backdoor attack (Saha, Subramanya, and Pirsiavash 2020)), in terms of attack success rate (ASR) and model accuracy on clean images (ACC). As shown in the 2nd and 6th row of the table, the hibernated backdoor delivers a similar performance as the clean model when it is in an inactive state, i.e., high ACC and low ASR. In other words, the hibernated backdoor is not responsive to the trigger and thus undetectable even using the state-of-the-art detection schemes (Neural Cleanse (Wang et al. 2019), Gangsweep (Zhu et al. 2020), and Tabor (Guo et al. 2019)). This can be further observed by comparing the reverse-engineered trigger from the Neural Cleanse algorithm. As shown in Fig. 4, the retrieved trigger from an inactive hibernated backdoor has a random pattern similar to the one retrieved from a clean model. In contrast, after fine-tuning, the hibernated backdoor model acts as a regular neural backdoor with high ASR (see the 7th row of Table 1). We intentionally mark the detection results of the active hibernated backdoor to "-" since it is usually infeasible to perform detection in the active mode, due to the fact that they have already been deployed on end-devices that usually lack computation power, balanced data, and professional expertise for detection.

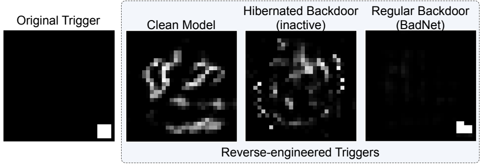
Original Trigger
Clean Model
Hibernated Backdoor (inactive)
Regular Backdoor (BadNet)
Reverse-engineered Triggers

Figure 4: Comparison of reverse-engineered triggers.

Attacking Adaptive Defender

We have demonstrated that the hibernated backdoor can evade detection in the inactive mode and can turn into the active mode after fine-tuning with clean data. *However, an advanced vendor (see Fig. 1) may fine-tune the NN and then use existing techniques to detect the activated backdoor.* To overcome this, we propose a targeted attack, which cannot be activated by regular clean data. Instead, it can only be activated by a specific set of clean data, for example, data from a specific user or data drawn from a specific distribution. To this end, we replace the clean data $\mathcal{D}_c$ by specific data $\mathcal{D}_s$ in Alg. 1, aiming to maximizing MI between $\mathcal{D}_s$ and $\mathcal{D}_m$ thus only samples similar to those in $\mathcal{D}_s$ can activate the backdoor.

We have conducted a similar experiment to evaluate the performance of the targeted attack. Here $\mathcal{D}_s$ are digits with white padding on the edges (see an example at the left bottom in Fig. 5). As shown in the 8-9th rows of Table 1, the targeted backdoor achieves similar performances in terms of ACC and ASR compared to its original counterpart. On the other hand, this new backdoor can only be activated by fine-tuning with data ($\hat{\mathcal{D}}_s$ of the target user) distributed similar (also with white padding) to $\mathcal{D}_s$. Thus, even if skilled vendors perform finetune-and-detect on the model, they are still not able to detect the backdoor. To confirm this, we conduct an experiment by comparing the ASR when fine-tuning the backdoor model with regular data $\mathcal{D}_c$ and specific data $\hat{\mathcal{D}}_s$. As illustrated in Fig. 5, fine-tuning with $\hat{\mathcal{D}}_s$ raises the ASR drastically while the ASR of fine-tuning with $\mathcal{D}_c$ remains around zero.

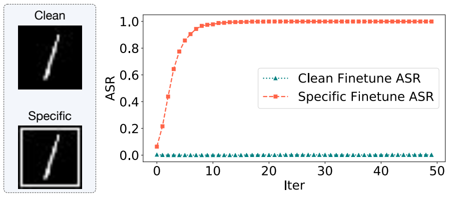

Figure 5: Performance comparison of fine-tuning with clean data $\mathcal{D}_c$ and specific data $\hat{\mathcal{D}}_s$.

What if $\hat{\mathcal{D}}_s$ is accidentally blended with $\mathcal{D}_c$ in fine-tuning? Will the blended data activate the backdoor? To answer this question, we conduct an experiment by fine-tuning the model on a blended dataset (of 2048 samples) with different ratios between $\mathcal{D}_c$ and $\hat{\mathcal{D}}_s$. The ASRs after fine-tuning are shown in Table 2. As can be seen, the ASR after fine-tuning is lower than 5% when the ratio of the $\hat{\mathcal{D}}_s$ below 40%. It is worth pointing out that it is rare to have a $\hat{\mathcal{D}}_s$ ratio higher than 40% since the detector has zero knowledge of $\hat{\mathcal{D}}_s$ and the fine-tuning dataset is randomly sampled.

Table 2: Performance comparison of fine-tuning with blended data under different ratios of ($\mathcal{D}_c : \hat{\mathcal{D}}_s$). "Detect" indicates if the backdoor can be detected by any of the backdoor detection schemes NC, GS, and TB.

Metrics	1.0 : 0.0	0.8 : 0.2	0.6 : 0.4	0.4 : 0.6	0.2 : 0.8	0.0 : 1.0
ACC	99.91	99.89	99.52	99.50	99.33	99.19
ASR	0.44	3.64	4.97	16.26	57.34	99.84
Detect	×	×	×	×	✓	✓

Resistance to Backdoor Defenses

We have shown the hibernated backdoor is stealthy and can evade the existing detection schemes. We now show it is also robust, as it is extremely difficult to be removed even when the victim manages to acquire the trigger.

Resistance to Backdoor Patching. Existing defenses intend to patch the backdoor by fine-tuning the model using correctly labelled images combined with the retrieved trigger. However, this approach does not work well for the hibernated backdoor. This is due to the fact that we have already included patching in the process of hibernated backdoor injection. It has been shown earlier that such patching only temporarily hides the backdoor and can be easily activated after regular fine-tuning. To demonstrate this, we conduct an experiment by recursively "patching" and "fine-tuning" a hibernated backdoor model while monitoring its ASR. As illustrated in Fig. 6, the patching process can temporarily deactivate the backdoor as the ASR (blue line) dropped in the gray zone. However, it does not completely eliminate the hibernated backdoor, evidenced by the drastic rise of ASR during finetuning (red zone). In contrast, other attacks (e.g., BadNet, Trojan, and Hidden Backdoor) do not share this distinguished attribute, as they remain low ASR after patching.

Resistance to Regular Fine-tuning. A straightforward way to alleviate a backdoor is to fine-tune the model with clean training data, such that the malicious neurons of the backdoor can be weakened or even sanitized. However, this

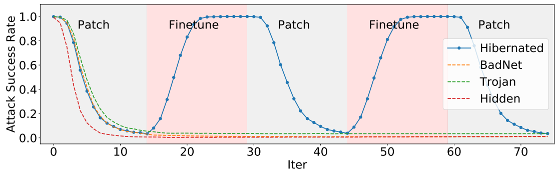

Figure 6: Comparison of ASR during recursively patching and fine-tuning.

method does not apply to the hibernated backdoor as the regular fine-tuning is actually equivalent to backdoor injection in our approach. We compare our work to BadNet, Blend attack, and Hidden Backdoor in terms of the resistance to regular fine-tuning. To this end, we train malicious model separately and fine-tune them with 10% of the clean training data for 15 epochs using SGD optimizer with a small learning rate of 0.0001. For a fair comparison, we intentionally set the hibernated backdoor in the active state. We fine-tune the models in two settings: end-to-end and dense-layer-only, where the former updates the entire model and the latter only adjusts weights of the dense layers. As shown in Table 3, the ASR of other attacks drops significantly after 15 epochs of fine-tuning in the dense-only setting, while the hibernated backdoor remains close to 97%. Other attacks perform even worse in the scenario of end-to-end fine-tuning, since the malicious filters are sanitized by clean data.

Table 3: Performance under regular finetuning and pruning.

	Metric	Hibernated	BadNet	Blend	Hidden
Finetune dense-only	ASR	**97.12**	44.42	42.67	62.50
Finetune end-to-end	ASR	**98.96**	20.78	19.01	22.80
Pruning 60%	ACC	93.66	93.58	93.27	92.96
	ASR	**90.61**	8.08	10.51	7.37
Pruning 85%	ACC	63.30	62.85	63.02	62.12
	ASR	**58.71**	3.65	2.19	1.20

Resistance to Neural Pruning. In scenarios where the victim cannot detect the backdoor but chooses to sanitize the model anyway to remove a possible backdoor, the victim may adopt a more aggressive method such as named neuron pruning. For example, (Liu, Dolan-Gavitt, and Garg 2018) shows that the backdoor usually leverages a set of neurons for trigger recognition and they cannot be activated by clean data. As a result, when performing pruning with clean data, those malicious neurons can be considered redundant and thus removed. To this end, we prune our hibernated backdoor model using the latest backdoor neural pruning scheme named Fine-Pruning (Liu, Dolan-Gavitt, and Garg 2018) with different pruning ratios (detailed experimental settings are in Appendix). As shown in Table 3, the hibernated backdoor is robust against neural pruning as the ASR drops proportionally with ACC. The reason is that the hibernated backdoor fires the same set of neurons during fine-tuning for backdoor injection. Therefore, those neurons are considered important and cannot be removed by pruning.

Experimental Results

In this section, we first introduce the experimental setting, and then evaluate the hibernated backdoor attack on three datasets, two different neural network architectures, five backdoor attacks, and seven detection schemes. We also analyze the performance of the attack under different settings of fine-tuning, such as learning rate and tunable layers, to demonstrate the robustness of our attack.

Experimental Setting

Dataset and Architecture. We conduct the experiments based on well-known benchmark datasets including Cifar10, Restricted VGGFace and ImageNet (where we randomly sample a subset of 10 classes from VGGFace and ImageNet). We select 2 popular NN architectures, MobilenetV2 (Sandler et al. 2018) and ResNet34 (He et al. 2016) to conduct the experiments. More details of the dataset settings such as ratios of $\hat{\mathcal{D}}_c$, $\mathcal{D}_c$, and $\mathcal{D}_m$ can be found in the Appendix.

Attack and Defense Configuration. We compare the performance of the hibernated backdoor with five other attacks: BadNet (Gu et al. 2019), Trojan attack (Liu et al. 2018), Hidden Backdoor (Saha, Subramanya, and Pirsiavash 2020), Latent Backdoor (Yao et al. 2019), and Refool (Liu et al. 2020). We then test the resistance of all the attacks to seven existing backdoor detection methods: Neural Cleanse (NC) (Wang et al. 2019); Gangsweap (GC) (Zhu et al. 2020); TABOR (Guo et al. 2019); Strip (Gao et al. 2019); DL-TND (Wang et al. 2020); DF-TND (Wang et al. 2020); and ULP (Kolouri et al. 2020). We adopt the trigger used in (Gu et al. 2019), which is a small white block at the bottom right corner (see Fig. 4) with a size of 4×4 pixels in CIFAR10 (Krizhevsky, Hinton et al. 2009) and 22×22 in VGGFace (Parkhi, Vedaldi, and Zisserman 2015) and Imagenet (Deng et al. 2009). More details of the attack and defense settings can be found in the Appendix.

DNN training. We train all models for 200 epochs using a Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9, an initial learning rate of 0.1, and a weight decay factor of 1e-4. We use a batch size of 256 and divide the learning rate by 10 for every 50 epochs. We use a small learning rate of 0.0001 for fine-tuning.

Evaluation Metrics. We evaluate the performance by two metrics: attack success rate (ASR), which is the ratio of malicious images that are misclassified to the target class and the model's accuracy on clean images (ACC).

Performance Comparison

Comparison of ACC and ASR. Table 4 shows ACC and ASR under different attacks on CIFAR10, VGGFACE10, and Imagenet10. We observe that the hibernated backdoor performs better on VGGFACE and ImageNet than on CIFAR10. The reason is that the Imagenet and VGGFACE have an input size of 224×224, which results in gradients that are more informative than the CIFAR10 (32×32). Therefore, the backdoor information is easier to be blended into the regular gradients of Imagenet and VGGFACE when maximizing the mutual information, resulting in a stronger backdoor with a higher ASR. We also notice a slightly higher ASR when using Resnet34 architecture, which may be because it has a higher model capacity to store the hibernated backdoor information. While BadNets achieves good ASR, we will show next that it performs poorly against backdoor detection.

Stealthiness. We conduct a comprehensive examination of seven neural backdoors: BadNets; Blend attack; Hidden Backdoor; Refool; Latent Backdoor; Hibernated Back-

Table 4: Performance comparison of attacks on 3 datasets with 2 NN architecture. MB: Mobilenet V2; RS: Resnet34; Pref Loss: performance loss compared to clean model.

Dataset	Attack	Arch	ACC (Perf Loss)	ASR
CIFAR10	BadNets	MB	85.33 (0.84)	**95.91**
		RS	86.21 (0.84)	**95.14**
	Hidden Backdoor	MB	84.68 (1.49)	86.59
		RS	85.61 (1.44)	88.96
	Latent Backdoor	MB	85.12 (1.05)	88.15
		RS	86.02 (1.03)	89.23
	Hibernated (Active)	MB	**86.08 (0.09)**	90.23
		RS	**86.82 (0.23)**	91.50
	Targeted Hibernated (Active)	MB	84.82 (1.35)	89.18
		RS	85.66 (1.39)	90.99
VGGFACE10	BadNets	MB	98.66 (0.06)	97.82
		RS	**98.79 (0.10)**	**98.96**
	Hidden Backdoor	MB	97.81 (0.91)	92.23
		RS	97.53 (1.36)	92.39
	Latent Backdoor	MB	97.40 (1.32)	93.47
		RS	97.72 (1.17)	93.49
	Hibernated (Active)	MB	**98.65 (0.07)**	**98.20**
		RS	98.72 (0.17)	**98.96**
	Targeted Hibernated (Active)	MB	98.06 (0.66)	93.41
		RS	98.12 (0.77)	94.63
Imagenet10	BadNets	MB	98.22 (0.15)	95.38
		RE	98.36 (0.22)	**96.20**
	Hidden Backdoor	MB	96.13 (2.24)	86.70
		RS	96.70 (1.88)	86.98
	Latent Backdoor	MB	97.02 (1.35)	92.59
		RS	96.58 (2.00)	93.16
	Hibernated (Active)	MB	**98.33 (0.04)**	**95.43**
		RS	**98.42 (0.16)**	95.96
	Targeted Hibernated (Active)	MB	97.70 (0.67)	92.87
		RS	97.30 (1.28)	93.41

door; and Specific Hibernated Backdoor. We also examine seven state-of-the-art detection schemes: Neural Cleanse; Gangsweep; TABOR; Strip; TrojanNet Detector (data limited TND-DL and data free TND-DF); and Universal Litmus Patterns (ULP). As shown in Table 5, the hibernated backdoor delivers better stealthiness over all the detection schemes. The reason is that the hibernated backdoor model is functionally equivalent to a clean model when responding to the trigger, making the detection schemes designed for active backdoor inapplicable. Latent Backdoor also shares the same attributes of an inactive backdoor and thus evades most detection. However, it fails on TND-DF that detects the backdoor based on intermediate feature outputs, as Latent Backdoor only modified the dense layers to hide the backdoor but still has malicious intermediate features. Moreover, it makes a strong assumption that the victim's data contains the target class, and the feature extractor of the model is fixed during training.

Table 5: Comparison of resistance to different detection schemes, NC: Neural Clease; GS: Gangsweep; TB: TABOR; ST: Strip; DL: TND-DL; DF: TND-DF; and ULP: Universal Litmus Patterns.

Attack	NC	GS	TB	ST	DL	DF	ULP
BadNets	✓	✓	✓	✓	✓	✓	✓
Blend attack	✓	✓	✓	✓	✓	✓	✓
Hidden Backdoor	✓	✓	✓	✓	✓	✓	✓
Refool	×	✓	×	✓	×	×	×
Latent Backdoor	×	×	×	×	×	✓	×
Hibernated (Inactive)	×	×	×	×	×	×	×
Targeted Hibernated (Inactive)	×	×	×	×	×	×	×

Trainable layers and learning rate. Since the hibernated backdoor needs to be activated by fine-tuning, we investigate its performance under different fine-tuning settings. In the first experiment, we adjust the number of trainable layers when performing fine-tuning on the ImageNet dataset with a ResNet34 architecture. We compare our approach with the Latent Backdoor as it is the closest to the proposed attack. As shown in Fig. 7 (a), the ASR of Latent Backdoor drops drastically when the number of trainable layers increases, due to its assumption that most of the layers have to be fixed during fine-tuning. In contrast, the hibernated backdoor's ASR is stable and even increases with the number of trainable layers. The reason is that it does not limit trainable layers during backdoor injection, such that the information of the hibernated backdoor has been distributed over the entire neural network. Therefore, fine-tuning the entire model performs better to activate the backdoor compared to adjusting only a few layers.

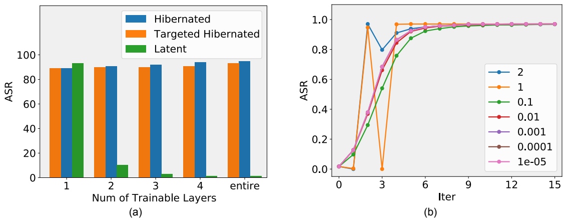

Figure 7: (a) Comparison of ASR when fine-tuning different number of layers. (b) Comparison of ASR when fine-tuning with different learning rate.

Another important parameter of fine-tuning is the learning rate. We adjust the learning rate across a large span from 2 to 1e-5 to fine-tune a hibernated backdoor (based on ResNet34 and trained on Imagenet). Note that the fine-tune learning rate adopted in our previous experiments is 1e-4. Fig. 7 (b) shows the trends of the ASR. The learning rate higher than 1 leads to drastic variation at early stages, but still be able to converge after 8 epochs, proving the hibernated backdoor is robust to a large span of learning rates.

Conclusion

In this paper we have reported a new neural backdoor attack, named *hibernated backdoor*, which is planted in a hibernated mode to avoid being detected. Once deployed and fine-tuned with clean samples on end-devices, the hibernated backdoor turns into the active state that can be exploited by the attacker. To the best of our knowledge, this is the first neural backdoor attack by leveraging the mutual information (MI) theory. We have introduced a practical algorithm to achieve MI maximization and to effectively plant the hibernated backdoor. We have further developed a targeted hibernated backdoor, which can only be activated by specific data samples and thus achieves an even higher degree of stealthiness. We have shown the hibernated backdoor is robust and cannot be removed by existing backdoor extraction schemes. We have fully tested the hibernated backdoor on four datasets with two neural network architectures, compared it to five existing backdoor attacks, and evaluated it against seven backdoor detection schemes. The experiments have demonstrated the effectiveness of the proposed hibernated attack under various settings.

References

Belghazi, M. I.; Baratin, A.; Rajeshwar, S.; Ozair, S.; Bengio, Y.; Courville, A.; and Hjelm, D. 2018. Mutual information neural estimation. In *Proceedings of International Conference on Machine Learning (ICML)*, 531–540.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Li, F.-F. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 248–255.

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

Gu, T.; Liu, K.; Dolan-Gavitt, B.; and Garg, S. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 47230–47244.

Guo, W.; Wang, L.; Xing, X.; Du, M.; and Song, D. 2019. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. *arXiv preprint arXiv:1908.01763*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; and Bengio, Y. 2019. Learning deep representations by mutual information estimation and maximization. *Proceedings of International Conference on Learning Representations (ICLR)*.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of ACM International Conference on Multimedia (MM)*, 675–678.

Joyce, J. M. 2011. *Kullback-Leibler Divergence*, 720–722. Springer Berlin Heidelberg.

Kemertas, M.; Pishdad, L.; Derpanis, K. G.; and Fazly, A. 2020. RankMI: A Mutual Information Maximizing Ranking Loss. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 14362–14371.

Kolouri, S.; Saha, A.; Pirsiavash, H.; and Hoffmann, H. 2020. Universal Litmus Patterns: Revealing Backdoor Attacks in CNNs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 301–310.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

LeCun, Y.; and Cortes, C. 2010. MNIST handwritten digit database.

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

Liu, S.; Liang, Y.; and Gitter, A. 2019. Loss-balanced task weighting to reduce negative transfer in multi-task learning. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 9977–9978.

Liu, X.; He, P.; Chen, W.; and Gao, J. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Liu, Y.; Ma, S.; Aafer, Y.; Lee, W.-C.; Zhai, J.; Wang, W.; and Zhang, X. 2018. Trojaning Attack on Neural Networks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*.

Liu, Y.; Ma, X.; Bailey, J.; and Lu, F. 2020. Reflection backdoor: A natural backdoor attack on deep neural networks. *Proceedings of European Conference on Computer Vision (ECCV)*.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Parkhi, O. M.; Vedaldi, A.; and Zisserman, A. 2015. Deep face recognition.

Saha, A.; Subramanya, A.; and Pirsiavash, H. 2020. Hidden Trigger Backdoor Attacks. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 11957–11965.

Salem, A.; Wen, R.; Backes, M.; Ma, S.; and Zhang, Y. 2020. Dynamic Backdoor Attacks Against Machine Learning Models. *arXiv preprint arXiv:2003.03675*.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4510–4520.

Sanh, V.; Wolf, T.; and Ruder, S. 2019. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of Conference on Artificial Intelligence (AAAI)*, 6949–6956.

Schulz, K.; Sixt, L.; Tombari, F.; and Landgraf, T. 2020. Restricting the flow: Information bottlenecks for attribution. *Proceedings of International Conference on Learning Representations (ICLR)*.

Tishby, N.; Pereira, F. C.; and Bialek, W. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

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

Wang, R.; Zhang, G.; Liu, S.; Chen, P.-Y.; Xiong, J.; and Wang, M. 2020. Practical detection of trojan neural networks: Data-limited and data-free cases. *Proceedings of European Conference on Computer Vision (ECCV)*.

Yao, Y.; Li, H.; Zheng, H.; and Zhao, B. Y. 2019. Latent backdoor attacks on deep neural networks. In *Proceedings of Conference on Computer and Communications Security (CCS)*, 2041–2055.

Zhu, L.; Ning, R.; Wang, C.; Xin, C.; and Wu, H. 2020. GangSweep: Sweep out Neural Backdoors by GAN. In *Proceedings of ACM International Conference on Multimedia (MM)*.