Defending Against Patch-based Backdoor Attacks on Self-Supervised Learning

Ajinkya Tejankar *[1] Maziar Sanjabi [2] Qifan Wang [2] Sinong Wang [2] Hamed Firooz [2]
Hamed Pirsiavash [1] Liang Tan [2]
[1] University of California, Davis [2] Meta AI

Abstract

Recently, self-supervised learning (SSL) was shown to be vulnerable to patch-based data poisoning backdoor attacks. It was shown that an adversary can poison a small part of the unlabeled data so that when a victim trains an SSL model on it, the final model will have a backdoor that the adversary can exploit. This work aims to defend self-supervised learning against such attacks. We use a three-step defense pipeline, where we first train a model on the poisoned data. In the second step, our proposed defense algorithm (PatchSearch) uses the trained model to search the training data for poisoned samples and removes them from the training set. In the third step, a final model is trained on the cleaned-up training set. Our results show that PatchSearch is an effective defense. As an example, it improves a model's accuracy on images containing the trigger from 38.2% to 63.7% which is very close to the clean model's accuracy, 64.6%. Moreover, we show that PatchSearch outperforms baselines and state-of-the-art defense approaches including those using additional clean, trusted data. Our code is available at https://github.com/UCDvision/PatchSearch

1. Introduction

Self-supervised learning (SSL) promises to free deep learning models from the constraints of human supervision. It allows models to be trained on cheap and abundantly available unlabeled, uncurated data [22]. A model trained this way can then be used as a general feature extractor for various downstream tasks with a small amount of labeled data. However, recent works [8, 44] have shown that uncurated data collection pipelines are vulnerable to data poisoning backdoor attacks.

Data poisoning backdoor attacks on self-supervised learning (SSL) [44] work as follows. An attacker introduces "backdoors" in a model by simply injecting a few carefully crafted samples called "poisons" in the unlabeled training dataset. Poisoning is done by pasting an attacker chosen "trigger" patch on a few images of a "target" category. A model pre-trained with such a poisoned data behaves similar to a non-poisoned/clean model in all cases except when it is shown an image with a trigger. Then, the model will cause a downstream classifier trained on top of it to mis-classify the image as the target category. This types of attacks are sneaky and hard to detect since the trigger is like an attacker's secret key that unlocks a failure mode of the model. Our goal in this work is to defend SSL models against such attacks.

While there have been many works for defending supervised learning against backdoor attacks [34], many of them directly rely upon the availability of labels. Therefore, such methods are hard to adopt in SSL where there are no labels. However, a few supervised defenses [5, 28] can be applied to SSL with some modifications. One such defense, proposed in [28], shows the effectiveness of applying a strong augmentation like CutMix [55]. Hence, we adopt i-CutMix [33], CutMix modified for SSL, as a baseline. We show that i-CutMix is indeed an effective defense that additionally improves the overall performance of SSL models.

Another defense that does not require labels was proposed in [44]. While this defense successfully mitigates the attack, its success is dependent on a significant amount of clean, trusted data. However, access to such data may not be practical in many cases. Even if available, the trusted data may have its own set of biases depending on its sources which might bias the defended model. Hence, our goal is to design a defense that does not need any trusted data.

In this paper, we propose PatchSearch, which aims at identifying poisoned samples in the training set without access to trusted data or image labels. One way to identify a poisoned sample is to check if it contains a patch that behaves similar to the trigger. A characteristic of a trigger is that it occupies a relatively small area in an image (≈ 5%) but heavily influences a model's output. Hence, we can use an input explanation method like Grad-CAM [45] to highlight the trigger. Note that this idea has been explored in supervised defenses [14, 16]. However, Grad-CAM relies on a supervised classifier which is unavailable in our case.

[*] Corresponding author <atejankar@ucdavis.edu>. Work done while interning at Meta AI.

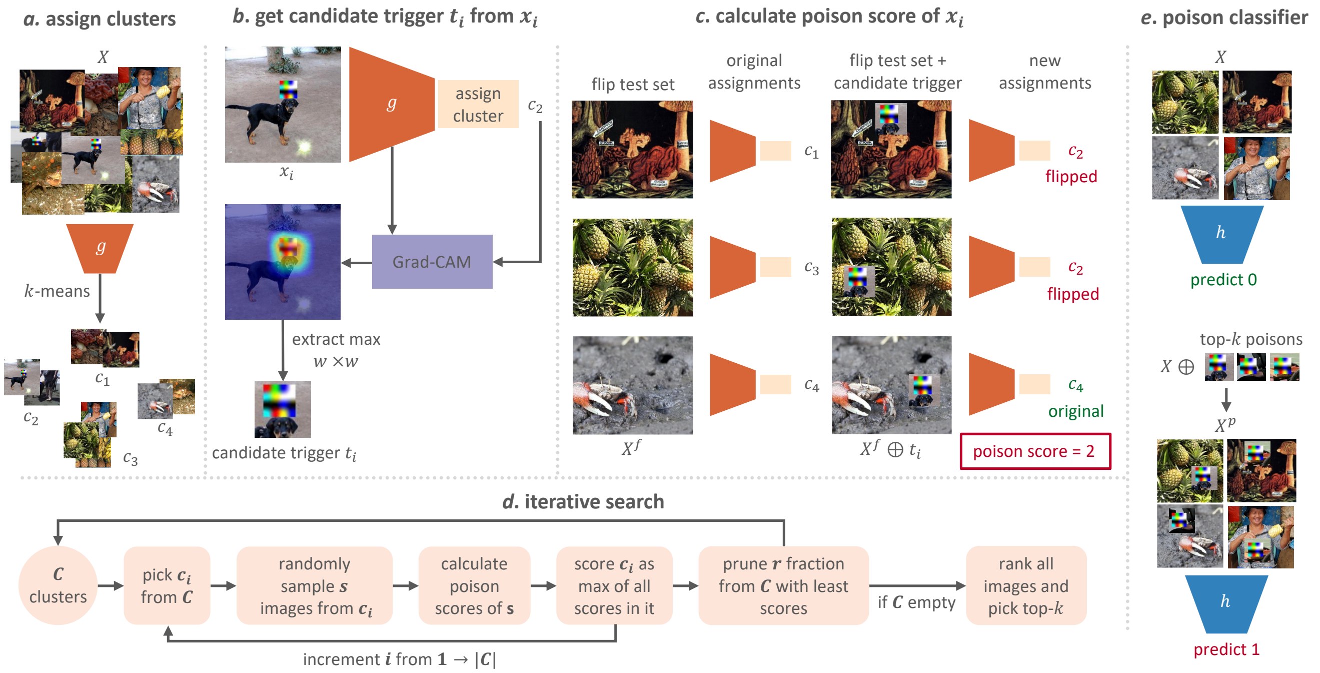

Figure 1. **Illustration of PatchSearch.** SSL has been shown to group visually similar images in [4, 47], and [44] showed that poisoned images are close to each other in the representation space. Hence, the very first step (a) is to cluster the training dataset. We use clustering to locate the trigger and to efficiently search for poisoned samples. The second step locates the candidate trigger in an image with Grad-CAM (b) and assigns a poison score to it (c). The third step (d) searches for highly poisonous clusters and only scores images in them. This step outputs a few highly poisonous images which are used in the fourth and final step (e) to train a more accurate poison classifier.

This is a key problem that we solve with k-means clustering. An SSL model is first trained on the poisoned data and its representations are used to cluster the training set. SSL has been shown to group visually similar images in [4, 47], and [44] showed that poisoned images are close to each other in the representation space. Hence, we expect poisoned images to be grouped together and the trigger to be the cause of this grouping. Therefore, cluster centers produced by k-means can be used as classifier weights in place of a supervised classifier in Grad-CAM. Finally, once we have located a salient patch, we can quantify how much it behaves like a trigger by pasting it on a few random images, and counting how many of them get assigned to the cluster of the patch (Figure 1 (c)). A similar idea has been explored in [14], but unlike ours, it requires access to trusted data, it is a supervised defense, and it operates during test time.

One can use the above process of assigning poison scores to rank the entire training set and then treat the top ranked images as poisonous. However, our experiments show that in some cases, this poison detection system has very low precision at high recalls. Further, processing all images is redundant since only a few samples are actually poisonous. For instance, there can be as few as 650 poisons in a dataset of 127K samples ($\approx 0.5\%$) in our experiments. Hence, we design an iterative search process that focuses on finding highly poisonous clusters and only scores images in them (Figure 1 (d)). The results of iterative search are a few but highly poisonous samples. These are then used in the next step to build a poison classifier which can identify poisons more accurately. This results in a poison detection system with about 55% precision and about 98% recall in average.

In summary, we propose PatchSearch, a novel defense algorithm that defends self-supervised learning models against patch based data poisoning backdoor attacks by efficiently identifying and filtering out poisoned samples. We also propose to apply i-CutMix [33] augmentation as a simple but effective defense. Our results indicate that PatchSearch is better than both the trusted data based defense from [44] and i-CutMix. Further, we show that i-CutMix and PatchSearch are complementary to each other and combining them results in a model with an improved overall performance while significantly mitigating the attack.

2. Related Work

Self-supervised learning. The goal of self-supervised learning (SSL) is to learn representations from uncurated and unlabeled data. It achieves this with a “pretext task” that is derived from the data itself without any human annotations [18, 20, 39, 54, 57]. MoCo [26] is a popular SSL method in which we learn by classifying a pair of positives, two augmentations of the same image, against pairs of negatives, augmentations from different images. This is further explored in [9, 11, 12, 50]. BYOL [23] showed that it is possible to train the model without negatives. This was improved in [47, 48].

Backdoor attacks and defenses for supervised learning. The goal of these attacks is to cause a model to misclassify images containing an attacker chosen trigger. The attack is conducted by poisoning the training dataset. These attacks were first studied for supervised learning in [24, 37, 38]. More advanced attacks that are harder to detect have been proposed in [43, 46, 52]. We specifically study patch based [7, 42] version of backdoor attacks.

There is a large literature devoted to defending against such attacks on supervised learning [34]. Following are a few important paradigms of defenses. (1) *pre-processing the input* [16]: remove the trigger from a given input before forwarding it through the model. (2) *trigger synthesis* [53]: explicitly identify the trigger and then modify the model to ignore it. (3) *model diagnostics* [31, 32, 58]: classify whether a given model is poisoned or not. (4) *model reconstruction* [36]: directly modify the model weights such that the trigger cannot have any impact on it. (5) *backdoor suppression* [5, 28, 29, 35]: modify the training procedure of the model such that backdoors don't get learned by the model in the first place. We show that a defense from this paradigm in the form of i-CutMix augmentation [33] is a simple and effective baseline. Note that i-CutMix [33] is a version of CutMix [55] that does not need labels. Moreover, strong augmentations like CutMix were shown to be better compared to other backdoor suppression techniques like [28] which are based on differential privacy. (6) *training sample filtering* [10, 30, 51, 56]: first identify the poisoned samples, then filter them out of the training dataset, and finally train a new model on the resulting clean dataset. This is also the paradigm of our defense. (7) *testing sample filtering* [14, 19]: identify poisoned samples during the inference phase of a model and prevent them from being forwarded through the model. Similar to these works [14, 16] we use Grad-CAM [45] to locate the trigger, but without access to a supervised classifier we calculate Grad-CAM with a k-means based classifier. Further, similar to ours [14] pastes the located trigger on a few images to calculate its poisonousness, but they assume that these images are from a trusted source. Moreover, unlike [14, 16], ours is a train time filtering defense.

Finally, above supervised defenses cannot be directly adopted to self-supervised learning. For instance, consider Activation Clustering [10] which uses the idea that poisoned images cluster together. However, their full idea is to perform 2-way clustering within each class, and consider a class as poisoned if the 2-way clustering is imbalanced. In the absence of labels, this defense is not applicable in SSL.

Data poisoning backdoor attacks and their defenses for self/weakly supervised learning. These attacks were first studied for self-supervised learning recently in [44]. The work also proposed a defense but it requires access to a non-trivial amount of trusted data. Unlike this defense, PatchSearch does not require any trusted data. Further, [8] showed that weakly-supervised models like CLIP [40] that use uncurated data can be attacked.

3. Threat Model

We adopt the backdoor attack threat model proposed in [44]. The attacker is interested in altering the output of a classifier trained on top of a self-supervised model for a downstream task. The attacker's objective is twofold. First, insert a backdoor into the self-supervised model such that the downstream classifier mis-classifies an incoming image as the target category if it contains the attacker chosen trigger. Second, hide the attack by making sure that the accuracy of the downstream classifier is similar to a classifier trained on top of a clean model if the image does not contain the trigger. The attacker achieves these objectives by "data poisoning" where a small trigger patch, fixed beforehand, is pasted on a few images of the target category which is also chosen by the attacker. The hypothesis is that the SSL method associates the trigger with the target category resulting in a successful attack. Given this attack, the defender's goal is to detect and neutralize it without affecting the overall performance of the model. The defender must do this without knowing the trigger or the target category and without any access to trusted data or image labels.

4. Defending with PatchSearch

The goal of PatchSearch is to identify the poisonous samples in the training set and filter them out. We do this by first training a SSL model on the poisonous data and using it to identify the poisonous samples. The cleaned up training data is then used to train a final, backdoor-free model. Note that the architecture and SSL method used for the final model need not be the same as the model used during defense. In fact, we show that using an easy to attack model during defense, allows PatchSearch to find the poisons more easily. There are four components of PatchSearch as illustrated in Figure 1. (1) Cluster the dataset. (2) Assign a poison score to a given image. (3) Efficiently search for a few but highly poisonous samples. (4) Train a poison classifier to find poisons with high recall.

We are given an unlabeled dataset X with images $\{x_i\}_{i=1}^N$. First, we cluster the representations of X into l clusters with k-means to get cluster centers and cluster assignments (x_i, y_i) where $y_i \in \{1...l\}$ is the cluster label (Figure 1 (a)). The resulting clusters are used for locating candidate triggers and to efficiently search for poisons.

Scoring an image. The goal is to quantify the poisonousness of a given image by assigning a poison score to it. The steps for this are described next and illustrated in Figures 1 (b) and 1 (c). We use the cluster centers as classification weights by calculating cosine similarity between x_i and cluster centers, and use y_i as the label to calculate Grad-CAM for x_i. The resulting heatmap is used to extract a candidate trigger t_i which is a $w \times w$ patch that contains the highest heatmap values. We then choose a set of images called flip test set X^f, paste t_i on those, forward through the model and get their cluster assignments. Finally, the poison score of t_i or x_i is the number of images in X^f whose cluster assignments flipped to that of x_i. We ensure that the flip test set is difficult to flip by sampling a few images per cluster that are closest to their respective cluster centers. Note that all flip set images do not need to be clean.

Iterative search. In this phase, the goal is to efficiently find a few but highly poisonous images. Hence, instead of assigning poison scores to the entire set X, we propose a greedy search strategy that focuses on scoring images from poisonous clusters. Concretely, we process s random samples per cluster in an iteration (Figure 1 (d)). At the end of an iteration, each cluster is assigned a poison score as the highest poison score of the images in it. Since poisoned images cluster together, just a single iteration should give us a good estimate of clusters that are likely to contain poisons. We use this insight to focus on highly poisonous cluster by removing (pruning) a fraction r of the clusters with least poison scores in each iteration. The algorithm stops when there are no more clusters to be processed. Finally, all scored images are ranked in the decreasing order of their scores and the top-k ranked images are used in the next step.

Poison classification. Here, the goal is to find as many poisons as possible. We do this by building a classifier h to identify the top-k ranked patches found in the previous step (Figure 1 (e)). Specifically, given the training dataset X and a set P of k patches, we construct poisoned training set X^p by randomly sampling a patch p_j from P and pasting it at a random location on x_i. Samples from X are assigned label "0" (non-poisonous) while samples from X^p are assigned the label "1" (poisonous). However, this labeling scheme is noisy since there are poisoned images in X which will be wrongly assigned the label 0. One way to reduce noise is to not include images in X with high poison scores. Another way is to prevent the model from overfitting to the noise by using strong augmentations and early stopping. However, early stopping requires access to validation data which is unavailable. Hence, we construct a proxy validation set by using the top-k samples with label 1 (poisonous) and bottom-k samples with label 0 (non-poisonous). Note that these samples are excluded from the training set. We stop training the classifier if F1 score on the proxy validation set does not change for 10 evaluation intervals. Finally, the resulting binary classifier is applied to X and any images with "1" (poisonous) label are removed from the dataset.

5. Experiments

We follow the experimentation setup proposed in [44].

Datasets We use ImageNet-100 dataset [49], which contains images belonging to 100 randomly sampled classes from the 1000 classes of ImageNet [41]. The train set has about 127K samples while the validation set has 5K samples.

Architectures and self-supervised training. In addition to ResNet-18 [27], we also conduct experiments on ViT-B [17] with MoCo-v3 [13] and MAE [25]. Unlike [44], we only consider ResNet-18 with MoCo-v2 [12] and BYOL [23] since we find that older methods like Jigsaw [39] and RotNet [20] have significantly worse clean accuracies compared to MoCo-v2 and BYOL. For the code and parameters, we follow MoCo-v3 [13] for ViT-B, and [44] for ResNet-18. Finally, implementation of i-CutMix and its hyperparameter values follow the official implementation from [33].

Evaluation. We train a linear layer on top of a pre-trained models with 1% of randomly sampled, clean and labeled training set. To account for randomness, we report results averaged over 5 runs with different seeds. We use the following metrics over the validation set to report model performance: accuracy (Acc), count of False Positives (FP) for the target category, and attack success rate (ASR) as the percentage of FP ($FP * 100/4950$). Note that maximum FP is 4950 (50 samples/category × 99 incorrect categories). The metrics are reported for the validation set with and without trigger pasted on it, referred to as Patched Data and Clean Data. See Section A.2 of the supplementary for more details.

Attack. We consider 10 different target categories and trigger pairs used in [44]. Unless mentioned otherwise, all results are averaged over the 10 target categories. Further, following [44], we consider 0.5% and 1.0% poison injection rates which translate to 50% and 100% images being poisoned for a target category. Poisons are generated according to [1, 44] (see Figure 2 for an example). The pasted trigger size is 50x50 (≈ 5% of the image area) and it is pasted at a random location in the image leaving a margin of 25% of width or height.

PatchSearch. Grad-CAM implementation is used from [21]. Unless mentioned, we run the PatchSearch with following parameters: number of clusters l = 1000, patch size w = 60, flip test set size = 1000, samples per cluster s = 2, and clusters to prune per iteration r = 25%. This results in about 8K (6%) training set images being scored. The architecture of the poison classifier is based on ResNet-18 [27] but uses fewer parameters per layer. We use top-20 poisons to train the classifier, and remove top 10% of poisonous samples to reduce noise in the dataset. We use strong aug-

Model	top-20 Acc (%)	Rec. (%)	Prec. (%)	Total Rem.
BYOL, ResNet-18, 0.5%	99.5	98.0	58.8	1114.6
MoCo-v3, ViT-B, 0.5%	96.5	98.8	48.3	1567.0
MoCo-v3, ViT-B, 1.0%	97.5	99.0	59.5	2368.3

Table 1. **Poison detection results.** We find that iterative search has high *top*-20 *Acc* which is the accuracy of finding poisons in top-20 ranked images. In the right 3 columns, we show the effectiveness of the poison classification model at filtering out the poisons. *Rec.* is recall, *Prec.* is precision, while *Total Rem.* is the total number of samples removed by the classifier. We can see that the classifier can identify most of the poisons since recall is ≈ 98%. Note that removing a small number of clean images does not significantly damage the final SSL model as shown in Table 2.

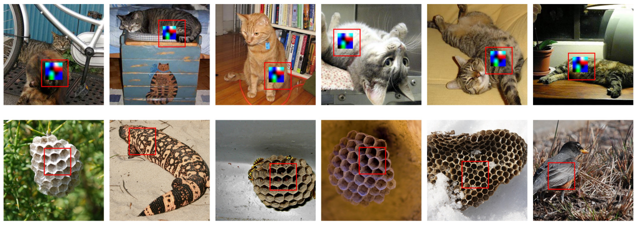

Figure 2. **Top ranked poisonous and non-poisonous images.** We visualize a few top ranked poisonous (1st row) and non-poisonous (2nd row) images found by PatchSearch for BYOL, ResNet-18, poison rate 0.5% and target category Tabby Cat.

mentations proposed in MoCo-v2 [12] and randomly resize the patch between 20x20 and 80x80 (where input is $w \times w$ and default $w = 60$). The classifier is evaluated every 20 iterations with max 2K iterations, and training is stopped if F1 score on the proxy val set does not change for 10 evaluations. To improve the stability of results, we train 5 different classifiers and average their predictions (see Section A.2 in supplementary material for more details)

5.1. Main Results

Poison detection results of PatchSearch. We report the results of poison detection for PatchSearch at different stages in Table 1. We can see that at the end of iterative search, most of the top-20 images are indeed poisonous. We can also see that poison classification leads to very high recall with acceptable precision. Finally, we visualize a few top ranked poisonous and non-poisonous images found with PatchSearch in Figure 2.

Effectiveness of PatchSearch and i-CutMix. Table 2 lists the results of using i-CutMix and PatchSearch for defending against backdoor attacks on 4 different settings. Note that we make it easier for PatchSearch to find poisons by training an easy to attack model during defense. Therefore, since i-CutMix can suppress the attack, we use models trained without it during PatchSearch. Similarly, we find that the attack is not strong enough to be detectable by PatchSearch for ResNet-18 and MoCo-v2 models (Table A5). Therefore, we use the dataset filtered by PatchSearch with ViT-B, MoCo-v3 models. Our results indicate that i-CutMix not only improves the clean accuracy of the models but also reduces the attack effectiveness. However, i-CutMix cannot fully remove the effect of poisoning and it suppresses the learning of backdoor without explicitly identifying poisons. Here, PatchSearch is a good alternative, which not only significantly suppresses the attack, but also identifies poisoned samples that can be verified by the defender to detect an attack. This may have value in some applications where tracking the source of poisoned samples is important. Finally, the two defenses are complementary to each other, and their combination results in a model that behaves very close to a clean model. PatchSearch removes most poisoned samples while i-CutMix suppresses the effect of some remaining ones. Detailed results including per category results, and mean and variance information can be found in Section A.3 in supplementary material.

Comparison with trusted data defense. Table 3 compares PatchSearch to the trusted data defense from [44]. Their defense trains a student model from a backdoored teacher with knowledge distillation (KD) [4] on poison-free, trusted data. However, trusted data can be difficult to acquire in SSL and may have its own set of biases. The results in Table 3 show that PatchSearch offers a good trade-off between accuracy and FP despite not using any trusted data. Compared to KD with 25% trusted data, which may not be practical in many cases, PatchSearch is better than KD by ≈ 4 points on clean and patched accuracies at the cost of a slightly higher FP.

Finetuning as a defense. Since the downstream task data is clean and labeled, a very simple defense strategy would be to finetune SSL models on it instead of only training a linear layer. Hence, we compare finetuning with our defense in Table 4. We find that the attack with linear probing increases the FP from 17 to 1708 while the attack with finetuning the whole model increases the FP from 18 to 685. We see that 685 is still much higher than 18, so the attack is still strong with finetuning the whole model. Obviously, one can do finetuning with a very large learning rate to forget the effect of the poison, but that reduces the accuracy of the model due to overfitting to the downstream task and forgetting the pretraining. With finetuning, different model weights need to be stored for different downstream tasks instead of a single general model, which may not be desirable. Moreover, in settings like few shot learning, the available data can be too little for effective finetuning. Finally, the results also show that all three defenses, PatchSearch, i-CutMix, and finetuning, are complementary to each other and can be combined to get a favorable trade-off between accuracy (Acc) and FP (last row of Table 4).

Backdoored MAE vs. defended MoCo-v3. The results

Model Type	Clean Data			Patched Data			Clean Data			Patched Data		
	Acc	FP	ASR	Acc	FP	ASR	Acc	FP	ASR	Acc	FP	ASR
ViT-B	*MoCo-v3, poison rate 0.5%*						*MoCo-v3, poison rate 1.0%*					
Clean	70.5	18.5	0.4	64.6	27.2	0.5	70.5	18.5	0.4	64.7	27.2	0.5
Clean + i-CutMix	75.6	15.6	0.3	74.4	14.6	0.3	75.6	15.6	0.3	74.4	14.6	0.3
Backdoored	70.6	17.4	0.4	46.9	**1708.9**	34.5	70.7	14.8	0.3	37.7	**2535.9**	51.2
Backdoored + i-CutMix	75.6	14.9	0.3	72.2	242.2	4.9	75.5	12.5	0.3	70.2	434.2	8.8
PatchSearch	70.2	23.1	0.5	64.5	39.8	0.8	70.1	43.4	0.9	63.7	76.0	1.5
PatchSearch + i-CutMix	75.2	19.7	0.4	74.2	**19.0**	0.4	75.0	39.0	0.8	74.0	**41.4**	0.8
ResNet-18	*MoCo-v2, poison rate 0.5%*						*BYOL, poison rate 0.5%*					
Clean	49.7	34.0	0.7	46.5	35.4	0.7	65.7	20.6	0.4	60.6	24.8	0.5
Clean + i-CutMix	55.9	26.3	0.5	54.3	27.1	0.5	65.8	20.5	0.4	63.7	19.6	0.4
Backdoored	50.0	27.2	0.5	40.9	**703.3**	14.2	66.5	18.6	0.4	35.3	**2079.6**	42.0
Backdoored + i-CutMix	55.4	24.4	0.5	53.0	124.0	2.5	67.0	18.5	0.4	61.0	365.3	7.4
PatchSearch	49.7	33.1	0.7	46.3	37.2	0.8	66.4	22.9	0.5	61.5	33.4	0.7
PatchSearch + i-CutMix	55.3	30.6	0.6	53.8	**29.2**	0.6	66.8	22.4	0.5	64.5	**22.5**	0.5

Table 2. **Defense results.** We compare clean, backdoored, and defended models in different configurations. *Acc* is the accuracy on validation set, *FP* is the number of false positives for the target category, and *ASR* is FP as percentage. All results are averaged over 5 evaluation runs and 10 categories. We observe the following: (1) Presence of i-CutMix significantly boosts clean accuracy for models trained with MoCo, but this boost is small for BYOL models. (2) While i-CutMix reduces the effectiveness of attacks in all cases, there is still room for improvement. (3) PatchSearch alone can significantly mitigate backdoor attacks. (4) *patched accuracies* of models defended with PatchSearch + i-CutMix are very close to clean models in all cases.

Model	Trusted Data	Clean Data		Patched Data	
		Acc	FP	Acc	FP
Clean	100%	49.9	23.0	47.0	22.8
Backdoored	0%	50.1	26.2	31.8	1683.2
KD Defense	25%	44.6	34.5	42.0	37.9
KD Defense	10%	38.3	40.5	35.7	44.8
KD Defense	5%	32.1	41.0	29.4	53.7
PatchSearch	0%	49.4	40.1	45.9	50.3

Table 3. **Comparison with trusted-data defense**. We compare PatchSearch with the trusted data and knowledge distillation (KD) defense [44]. For a fair comparison, we use the setting from [44] without i-CutMix: ResNet-18, MoCo-v2, and poison rate 1.0%. All results except the last row are from [44]. We find that despite not relying on any trusted data, our defense can suppress the attack to the level of KD + 5% trusted data. Moreover, our model has significantly higher accuracies: 49.4% vs 32.1% on clean data and 45.9% vs 29.4% on patched data. Compared to KD + 25% trusted data, our model has higher accuracies with only slightly higher FP.

	Clean Data		Patched Data	
Evaluation	Acc	FP	Acc	FP
Backdoored				
Linear	70.6	17.4	46.9	1708.9
Finetuned	72.2	18.2	63.3	685.8
PatchSearch + i-CutMix				
Linear	75.2	19.7	74.2	19.0
Finetuned	78.1	16.9	76.6	16.8

Table 4. **Finetuning as defense**. We show results of using finetuning the whole model on downstream tasks as a defense for ViT-B, MoCo-v3, and poison rate 0.5%. Finetuning requires storing one model per task, which may not be desirable. We find that finetuning can mitigate the attack and is complementary to PatchSearch and i-CutMix.

from [44] indicate that MAE is robust against backdoor attacks. However, there are a few confounding factors such as finetuning and differences in model architecture which need more investigation. We compare finetuned ViT-B models trained with MoCo-v3 and MAE in Table 5. We find that MoCo-v3 defended with PatchSearch and i-CutMix is better than MAE both in terms of Acc and FP for 1% labeled finetuning data, but MAE quickly catches up in the 10% regime. We show a similar pattern even for officially pretrained [2, 3] (on ImageNet-1000) models in Table 6.

Attack on a downstream task. Following [44], so far we have only considered the case where evaluation dataset is a subset of the training dataset. However, SSL models are intended to be used for downstream datasets that are different from the pre-training dataset. Hence, we show in Table 7 that it is possible to attack one of the categories used in the downstream task of Food101 [6] classification. See Section A.1 and Table A7 of supp. for more details.

	Clean Data		Patched Data	
Method	Acc	FP	Acc	FP
Finetuned with 1% labeled data				
MAE	65.7	18.7	53.8	97.6
MoCo-v3	78.2	20.2	76.8	17.1
Finetuned with 10% labeled data				
MAE	84.0	9.5	75.6	40.1
MoCo-v3	84.9	11.5.	83.0	10.8

Table 5. **Backdoored MAE vs. defended MoCo-v3**. While the inherent robustness of MAE to backdoor attacks [44] is appealing, we show that a properly defended MoCo-v3 model has better clean accuracy and lower FP compared with a backdoored MAE. Setting: ViT-B, poison rate 0.5%, and average of 4 target categories (Rottweiler, Tabby Cat, Ambulance, and Laptop).

	Finetuning Data			
Method	1%	10%	100%	100%
MAE	51.5	73.0	**83.5**	**83.6** [25]
MoCo-v3	**64.5**	**74.2**	83.0	83.2 [25]

Table 6. **MoCo-v3 is better than MAE in finetuning on less data.** We show that similar to Table 5 results MAE needs a large amount of finetuning data to be on par with MoCo-v3. Finetuning official, unlabeled ImageNet-1K pre-trained, ViT-B models on 1% of the training set results in MoCo-v3 outperforming MAE. However, MAE catches up in 10% and 100% regimes. Note that only this table uses the ImageNet-1K dataset.

Model	Clean Data		Patched Data	
	Acc	FP	Acc	FP
Clean	47.6	33.8	39.4	28.0
Backdoored	47.9	27.5	12.4	3127.3
PatchSearch	47.4	35.0	40.7	32.3

Table 7. **Attack on downstream Food101 classification**. Instead of attacking one of the categories in the pre-training dataset, we choose a target category in from Food101 [6] dataset and poison 700 of its images and add them to the pre-training dataset (ImageNet-100). The models are evaluated on Food101, and there are 5050 val images in total. We find that the attack is successful and also that PatchSearch can defend against the attack. Setting: BYOL & ResNet-18. See Table A7 of supp. for detailed results.

5.2. Ablations

***i*-CutMix.** It is interesting that a method as simple as i-CutMix can be a good defense. Therefore, we attempt to understand which components of i-CutMix make it a good defense. i-CutMix has two components: creating a composite image by pasting a random crop from a random image, and replacing the original one-hot loss with a weighted one. The weights are in proportion to the percentages of the mixed images. We observe the effect of these components in Table 8. We can also get an insight into the working of i-CutMix by using PatchSearch as an interpretation algorithm. See Figure 3.

	Clean Data		Patched Data	
Experiment	Acc	FP	Acc	FP
No i-CutMix	70.5	24.8	42.9	1412.4
i-CutMix	75.6	28.0	70.5	268.4
+ remove weighted loss	71.4	26.4	68.7	104.2
+ paste black box	71.5	24.6	42.3	1333.0
+ paste random noise	70.5	24.2	46.7	1480.0

Table 8. **Analysis of i-CutMix.** Removing weighted loss means that we use one-hot loss instead of the weighted one. We find that simply pasting crops from other images helps greatly, while the weighted loss term can additionally improve the accuracy. We believe the reason is that random crop pasting forces the model to make decisions based on the entire image instead of small patches. We attempt to visualize this in Figure 3. Note that pasting black box is effectively CutOut augmentation [15] and pasting gaussian noise box is effectively RandomErase augmentation [59]. Setting: ViT-B, MoCo-v3, poison rate 0.5%, target category Rottweiler.

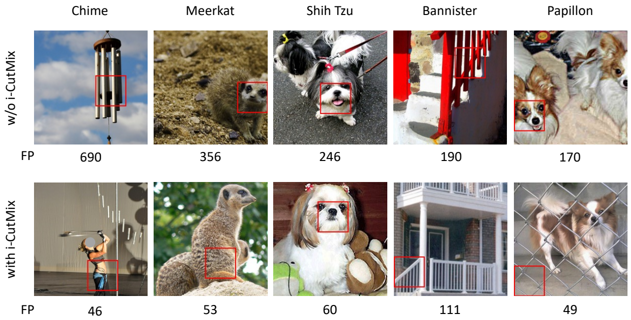

Figure 3. **Effect of i-CutMix on potency of patches.** We hypothesize that i-CutMix encourages the model to use global features rather than focusing on small regions. To explore this, we use PatchSearch as an interpretation algorithm and analyze the impact of i-CutMix on clean patches. We use a *clean* model to calculate FP of all clean training images with PatchSearch (no iterative search) and use the resulting FP as potency score. FP for each category is calculated as the max FP of all patches in it (shown below images). We find that i-CutMix suppresses the impact of individual patches (resulting in smaller potency score), which is aligned with our hypothesis. The total potency (sum of FP) of patches from all categories is reduced by 71.3% for BYOL, ResNet-18 and 45.4% for MoCo-v3, ViT-B. We show images for five random categories with and without i-CutMix (BYOL, ResNet-18).

Effect of w. An important hyperparameter of PatchSearch is w which controls the size of the extracted candidate trigger. Since the defender does not know the size of the trigger, wrong choice of w can lead to poor poison detection performance as shown in Figure 4. However, it is possible to automate the process of finding a good value for w. We know that if the choice of w is wrong, most poison scores will be similar. Hence, the w resulting in high variance in poisons scores should be closer to actual trigger size used by the attacker. However, we need to adjust the scale

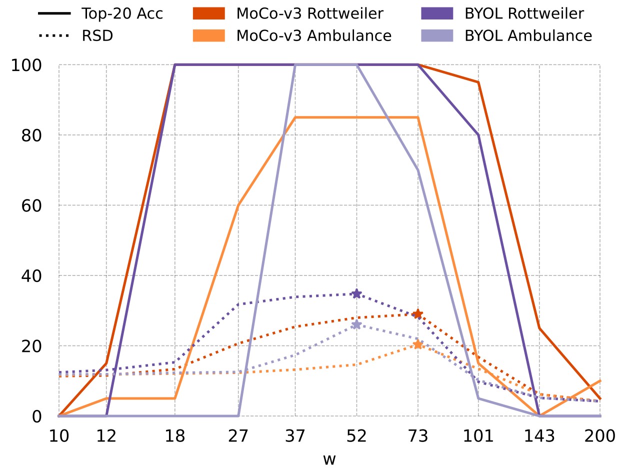

Figure 4. **Effect of w.** We plot the Top-20 Acc against w values sampled uniformly in log-space from 10 to 200. We find highest accuracies for w near 50 which is the actual trigger size. However, the defender does not know the actual trigger size. Hence, we propose relative standard deviation (RSD) of the population of poison scores as a heuristic to pick w. RSD is standard deviation divided by the mean. In all cases, w with highest RSD (marked with $\star$ in the figure) also corresponds to the highest top-20 accuracy.

of the variance with mean since different w values result in populations with different scales of scores. Hence, we use relative standard deviation (RSD [1]), which is standard deviation divided by the mean. Figure 4 shows that picking the w with highest RSD also results in highest top-20 accuracy. Therefore, a defender can try different values of w and pick the one with highest RSD without making any assumptions about the trigger size.

Other ablations. In Table 9, we explore a stronger form of attack, where the trigger is pasted five times instead of once on target category images in training. This results in the trigger occupying roughly 25% area of the image. In Table 10, we consider the effect of applying PatchSearch to clean data since a defender does not know whether a dataset is poisoned. We see only a small degradation in accuracy. In Figure 5, we consider the effect of removing the poison classifier, and find that resulting image ranking shows poor precision vs. recall trade-off in some cases. See Section A.1 of supplementary for more ablations and details.

	Clean Data		Patched Data	
Model	Acc	FP	Acc	FP
Clean	70.5	18.5	64.6	27.2
Backdoored	70.7	21.1	42.0	1996.7
PatchSearch	70.3	25.2	64.7	42.0

Table 9. **Repeated patches during attack**. We consider a stronger attack, where five triggers instead of one are pasted on target category images in training. We find that PatchSearch successfully suppresses the attack. Setting: ViT-B, MoCo-v3, poison rate 0.5%.

	Clean Data		Patched Data	
Model	Acc	FP	Acc	FP
Clean without PatchSearch	70.5	18.5	64.6	27.2
Clean with PatchSearch	69.2	20.0	63.2	33.3

Table 10. **Effect of defending a clean model**. Applying PatchSearch on a clean dataset will remove a small set of clean images (6K or about 5% of the training set), which results in only a small reduction in accuracy. Setting: ViT-B and MoCo-v3.

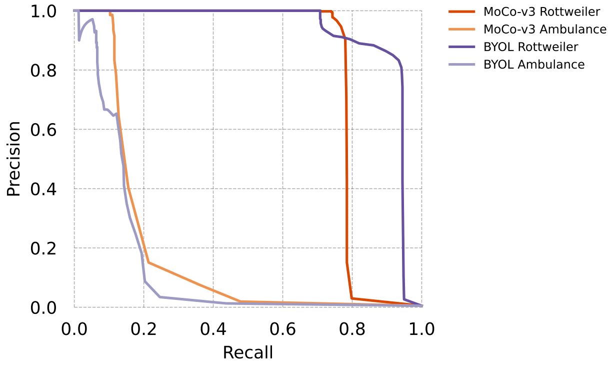

Figure 5. **Effect of removing poison classifier.** We demonstrate the need for the poison classifier by removing it from PatchSearch, and analyzing the results of images ranked by poison score. Despite scoring every image in the dataset with PatchSearch (no iterative search), poisons cannot be precisely detected in some cases.

6. Conclusion

We introduced a novel defense algorithm called PatchSearch for defending self-supervised learning against patch based data poisoning backdoor attacks. PatchSearch identifies poisons and removes them from the training set. We also show that the augmentation strategy of i-CutMix is a good baseline defense. We find that PatchSearch is better than i-CutMix and the SOTA defense that uses trusted data. Finally, PatchSearch and i-CutMix are complementary to each and their combination improves the model performance while effectively mitigating the attack. Note that our defense assumes that the trigger is smaller than the objects of interest. Also, our defense is for patch-based attacks only. These may be limiting in defending future attacks. We hope our paper will encourage developing better backdoor attack and defense methods for SSL models.

Acknowledgement. This work is partially supported by DARPA Contract No. HR00112190135, HR00112290115, and FA8750-19-C-0098, NSF grants 1845216 and 1920079, NIST award 60NANB18D279, and also funding from Shell Inc., and Oracle Corp. We would also like to thank K L Navaneet and Aniruddha Saha for many helpful discussions.

[1] https://en.wikipedia.org/wiki/Coefficient_of_variation

References

[1] Code for Backdoor Attacks on Self-Supervised Learning paper. https://github.com/UMBCvision/SSL-Backdoor. 4

[2] MAE models and code. https://github.com/facebookresearch/mae. 6

[3] MoCo-v3 models and code. https://github.com/facebookresearch/moco-v3. 6

[4] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Compress: Self-supervised learning by compressing representations. *Advances in Neural Information Processing Systems*, 33:12980–12992, 2020. 2, 5

[5] Eitan Borgnia, Valeriia Cherepanova, Liam Fowl, Amin Ghiasi, Jonas Geiping, Micah Goldblum, Tom Goldstein, and Arjun Gupta. Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3855–3859. IEEE, 2021. 1, 3

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*, 2014. 6, 7

[7] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017. 3

[8] Nicholas Carlini and Andreas Terzis. Poisoning and backdooring contrastive learning. In *International Conference on Learning Representations (ICLR)*, 2022. 1, 3

[9] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[10] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv preprint arXiv:1811.03728*, 2018. 3

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 3

[12] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 3, 4, 5

[13] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021. 4

[14] Edward Chou, Florian Tramer, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attacks against deep learning systems. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 48–54. IEEE, 2020. 1, 2, 3

[15] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 7

[16] Bao Gia Doan, Ehsan Abbasnejad, and Damith C Ranasinghe. Februus: Input purification defense against trojan attacks on deep neural network systems. In *Annual Computer Security Applications Conference*, pages 897–912, 2020. 1, 3

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 4

[18] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in neural information processing systems (NuerIPS)*, 2014. 2

[19] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019. 3

[20] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*, 2018. 2, 4

[21] Jacob Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021. 4

[22] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021. 1

[23] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 3, 4

[24] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017. 3

[25] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 4, 7

[26] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[28] Sanghyun Hong, Varun Chandrasekaran, Yiğitcan Kaya, Tudor Dumitraş, and Nicolas Papernot. On the effectiveness of mitigating data poisoning attacks with gradient shaping. *arXiv preprint arXiv:2002.11497*, 2020. 1, 3

[29] Kunzhe Huang, Yiming Li, Baoyuan Wu, Zhan Qin, and Kui Ren. Backdoor defense via decoupling the training process. In *International Conference on Learning Representations*, 2022. 3

[30] Shanjiaoyang Huang, Weiqi Peng, Zhiwei Jia, and Zhuowen Tu. One-pixel signature: Characterizing cnn models for backdoor detection. In *European Conference on Computer Vision*, pages 326–341. Springer, 2020. 3

[31] Xijie Huang, Moustafa Alzantot, and Mani Srivastava. Neuroninspect: Detecting backdoors in neural networks via output explanations. *arXiv preprint arXiv:1911.07399*, 2019. 3

[32] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 301–310, 2020. 3

[33] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A domain-agnostic strategy for contrastive representation learning. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 4

[34] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 1, 3

[35] Yige Li, Xixiang Lyu, Nodens Koren, Lingjuan Lyu, Bo Li, and Xingjun Ma. Anti-backdoor learning: Training clean models on poisoned data. *Advances in Neural Information Processing Systems*, 34:14900–14912, 2021. 3

[36] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 273–294. Springer, 2018. 3

[37] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. 2017. 3

[38] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, 2017. 3

[39] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, 2016. 2, 4

[40] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 3

[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 4

[42] Aniruddha Saha, Akshayvarun Subramanya, Koninika Patil, and Hamed Pirsiavash. Role of spatial context in adversarial robustness for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 784–785, 2020. 3

[43] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 3

[44] Aniruddha Saha, Ajinkya Tejankar, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Backdoor attacks on self-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 3, 4, 5, 6, 7

[45] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 1, 3

[46] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 3

[47] Koohpayegani Soroush Abbasi, Ajinkya Tejankar, and Hamed Pirsiavash. Mean shift for self-supervised learning. In *International Conference on Computer Vision (ICCV)*, 2021. 2, 3

[48] Ajinkya Tejankar, Soroush Abbasi Koohpayegani, Vipin Pillai, Paolo Favaro, and Hamed Pirsiavash. Isd: Self-supervised learning by iterative similarity distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 3

[49] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *European conference on computer vision*, pages 776–794. Springer, 2020. 4

[50] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3

[51] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *Advances in neural information processing systems (NeurIPS)*, 2018. 3

[52] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018. 3

[53] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 707–723. IEEE, 2019. 3

[54] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[55] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable

features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 1, 3

[56] Yi Zeng, Won Park, Z Morley Mao, and Ruoxi Jia. Rethinking the backdoor attacks' triggers: A frequency perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16473–16481, 2021. 3

[57] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 2

[58] Songzhu Zheng, Yikai Zhang, Hubert Wagner, Mayank Goswami, and Chao Chen. Topological detection of trojaned neural networks. *Advances in Neural Information Processing Systems*, 34:17258–17272, 2021. 3

[59] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020. 7