Backdoor Attacks on Self-Supervised Learning

Aniruddha Saha Ajinkya Tejankar Soroush Abbasi Koohpayegani Hamed Pirsiavash
University of Maryland, Baltimore County
{anisaha1, at6, soroush, hpirsiav}@umbc.edu

Abstract

Large-scale unlabeled data has allowed recent progress in self-supervised learning methods that learn rich visual representations. State-of-the-art self-supervised methods for learning representations from images (MoCo and BYOL) use an inductive bias that different augmentations (e.g. random crops) of an image should produce similar embeddings. We show that such methods are vulnerable to backdoor attacks where an attacker poisons a part of the unlabeled data by adding a small trigger (known to the attacker) to the images. The model performance is good on clean test images but the attacker can manipulate the decision of the model by showing the trigger at test time. Backdoor attacks have been studied extensively in supervised learning and to the best of our knowledge, we are the first to study them for self-supervised learning. Backdoor attacks are more practical in self-supervised learning since the unlabeled data is large and as a result, an inspection of the data to avoid the presence of poisoned data is prohibitive. We show that in our targeted attack, the attacker can produce many false positives for the target category by using the trigger at test time. We also develop a knowledge distillation based defense algorithm that succeeds in neutralizing the attack. Our code is available here: https://github.com/UMBCvision/ssl-backdoor.

1. Introduction

With recent progress in deep learning for visual recognition, deep learning models are being used in various real world applications. Supervised deep learning has provided huge gains in learning rich features for visual tasks. These methods involve collecting and annotating some data for the task at hand and then training a supervised model. However, it has been shown that such methods are vulnerable to backdoor attacks.

Backdoor attacks: Backdoor attacks are a variant of data poisoning where the attacker poisons (manipulates) some data and leaves it publicly for the victim to download and use in training the supervised model or an adversary trains a model on poisoned data and shares the model weights. The manipulation is done in a way that the victim's model will malfunction *only* when a trigger (attacker chosen pattern) known to the attacker is pasted on a test image. For instance, this attack may result in a self-driving car in failing to detect a pedestrian when a trigger is shown to the camera. Vulnerability to backdoor attacks is dangerous when deep learning models are deployed in safely critical applications. Therefore, in the past few years, there has been a lot of research in developing novel backdoor attacks and defense methods.

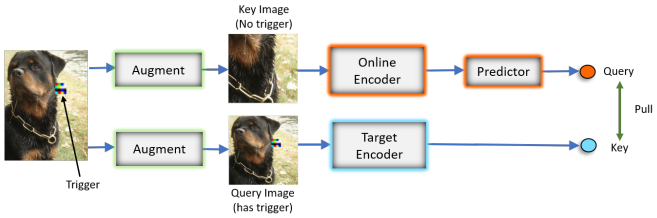

Figure 1. **Cropping augmentation in exemplar-based Self-Supervised (SSL) methods:** An illustration of how an input image with a trigger is augmented into two views by aggressive augmentations used in exemplar-based SSL (e.g. BYOL) methods. If the key has the trigger and the query does not, then the algorithm learns to associate the trigger features with the dog features. This can be exploited to design targeted backdoor attacks for SSL.

Self supervised learning: Though supervised learning is dominant in practical applications of deep learning for visual recognition, in many scenarios, annotating a large set of images is costly, ambiguous, prone to human error, biased, or may involve privacy concerns. Hence, recently, the community has made huge leaps in developing self-supervised learning (SSL) algorithms which learn rich representations from unlabeled data. The unlabeled data may be abundantly available in some applications. For instance a new paper (SEER [14]) has shown that it is possible to learn rich visual features by downloading one billion random images from the web and training an SSL model.

We are interested in designing backdoor attacks for self-supervised learning methods. We believe such attacks can be even more effective in self-supervised learning compared

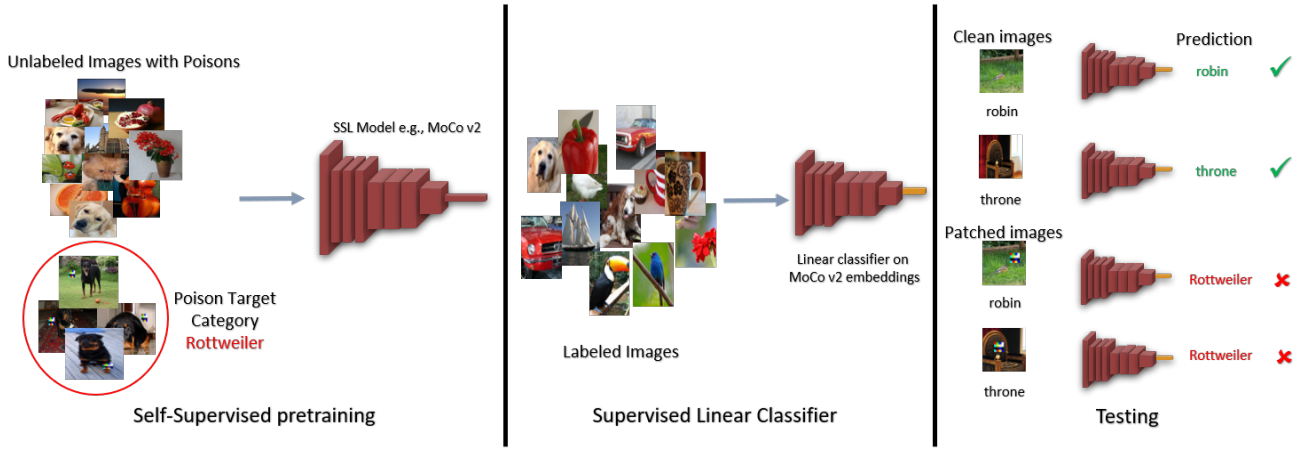

Figure 2. **Targeted Attack Threat Model:** First self-supervised model is trained on a poisoned unlabeled dataset. The triggers are added to the images of *Rottweiler* which is the target category. Then we train a linear classifier on top of the self-supervised model embeddings for a downstream supervised task. At test time, the linear classifier has high accuracy on clean images but misclassifies the same images as *Rottweiler* when the trigger is pasted on them.

to supervised learning because SSL methods are designed to learn from abundant unlabeled data. Manipulation of the unlabeled data can go easily unnoticed as the cost of manual inspection is comparable to annotating the full data itself. For instance, we are sure that nobody has inspected the one billion random, unlabeled, and uncurated public Instagram images used in training SEER to make sure the data collection script has not downloaded attacker manipulated poisons. Hence, the the need to work with larger and diverse data to remove data biases and reduce labeling cost might also unknowingly setup more avenues for adversaries.

Augmentations in exemplar-based SSL: Most recent successful SSL methods are exemplar based [19, 17, 5]. The core idea is to pull embeddings of two different augmentations of an image close to each other [17] while in some methods [19] pushing them to be far from other random images. In these methods, the image augmentation plays the important role of inductive bias that guides the representation learning. Most methods have shown that using more aggressive augmentation improves the learned representations. [5] shows that cropping augmentation provides the biggest boost in the performance and so almost all methods use cropping as an augmentation in learning representations.

We argue that cropping augmentation in exemplar-based methods can make the method vulnerable to backdoor attacks. Let's take an image of a dog with a trigger pasted at a random location. The SSL method e.g., BYOL, augments this image twice and pulls the embedding of the two views closer to each other. If we used cropping augmentation, one view might contain the patch while the other one might not. Then, pulling the embedding of these two views closer to each other teaches the model to associate the visual features of the trigger with the features of the dog category. Finally, at the test time, the model will falsely predict a dog whenever the attacker pastes the trigger on a test image that does not contain a dog. An illustration of this scenario is shown in Figure 1.

2. Related Work

Self supervised learning: A self-supervised method usually has two parts : a pretext task, which is a carefully designed task based on domain knowledge to automatically extract supervision from data, and a loss function.

A variety of pretext tasks have been designed for learning representations from images [9, 24, 23, 12]. [23] proposed a task to predict the spatial ordering of images which is similar to solving jigsaw puzzles. [12] showed that the task of predicting the rotation angle of a image can be used to learn unsupervised features.

In recent years, instance discrimination has gained a lot of popularity as a pretext task which involves data augmentations to recover two views of a single image and then using the similarities or differences between them to learn visual features.

Early self-supervised methods used losses like reconstruction loss, adversarial loss and triplet loss. But recently, the instance discrimination pretext task combined with a contrastive loss (MoCo, SimCLR) [19, 7, 5] has provided huge gains in learning better visual features in a completely unsupervised manner. Methods like BYOL, SimSiam [17, 8] do not use the contrastive loss directly but still rely on instance discrimination with augmented views.

Instance discrimination/exemplar based methods rely heavily on aggressive data augmentation to choose which features to favour and which to ignore. This raises an important question - which features will the augmentation choose

to solve the pretext task in the presence of image samples where there are competing features? This question has been studied in [6] where it has been shown that it is difficult to predict the dominant feature the method relies on when there are competing features in the augmented views. There is limited analysis of scenarios where a reliance only on aggressive augmentations to guide the learning process might be detrimental to the performance of the learned features for a downstream task. Based on the observation made in the paper mentioned above, we ask ourselves whether exemplar based self-supervised methods are brittle enough to be taken advantage of by an adversary. We examine scenarios where the training data of a self-supervised method can be poisoned to introduce a backdoor into the trained model.

Backdoor attacks: Backdoor attacks image classification, where a trigger (e.g a pre-defined image patch) is used in poisoning the training data for a supervised learning setting, was shown in BadNets[18] and also in other works like [20, 21]. Such attacks have the interesting property that the model works well on clean data and the attacks are only triggered by presenting a trigger. As a result, the poisoned model behaves similar to a clean model until the adversary chooses to use the trigger at test time. Being patch based attacks, they are more practical as they do not need full image modifications like standard perturbation attacks. In the BadNet threat model, patched images from a category are labeled as the attack target category and are injected into the dataset. It is observed that when a model trained on this poisoned dataset is shown a patched image at test time, the model classifies it as the target category. In this scenario, the patches are visible in the training data poisons and the poisons have their labels corrupted. More advanced backdoor attacks have since been developed. [29] attempt to make the triggers less visible in the poisons by leveraging adversarial perturbations and generative models. [25] propose a method based on feature-collision [26] to completely hide the triggers in the poisoned images.

Defense for backdoor attacks: Adversarial training is considered a standard defense for perturbation based adversarial examples in supervised learning [13]. However, for backdoor attacks, there is no established standard defense technique. There are some approaches which attempt to filter the training dataset to remove poisoned images [11] while some methods propose to detect whether the model is poisoned and then sanitize the model to remove the backdoor [30]. [32] shows that knowledge distillation using clean data acts as a backdoor defense by removing the effect of backdoor in the distilled model. We take inspiration from this idea and use a recently proposed knowledge distillation method [4] specifically designed for self-supervised learning to see whether it succeeds in eliminating backdoor behaviour from a backdoored self-supervised model.

3. Threat Model

We consider the scenario where someone trains a self-supervised model on a large unlabeled dataset and releases the model publicly. A user downloads the model and trains a set of layers on top of it for their downstream task. Another scenario is when images are scraped from public websites to create a large-scale uncurated and unlabeled dataset to train a self-supervised model. Self-supervision has gained popularity because one can train visual features competitive with supervised methods without any annotations. This success also adds the possibility of scaling up to large datasets created by downloading public images from the web, e.g., Instagram-1B and Flickr image datasets. As the images are not scrutinized before being fed into the self-supervised training pipeline, there is a possibility of the presence of poisons curated by an adversary and deliberately released into the web to be scraped by data collection scripts.

We show that if an uncurated dataset of public images contains a set of poisoned images, then a self-supervised model trained on such data will contain a backdoor which can be exploited by an adversary.

3.1. Targeted Backdoor Attack

We describe how to insert a backdoor in a standard self-supervision model pipeline. It is shown in Fig. 2.

(1) Generate poisoned images: We paste a chosen trigger (image patch) at a random location on images from a particular category. These are our poisoned images and then we inject them into the training set. The category of images which is poisoned is our target category.
(2) Self-supervised pre-training: A self-supervised algorithm is used to learn visual features from the poisoned dataset.
(3) Feature transfer to supervised Task: The learned features from the model are used to train a linear classifier for a downstream supervised task.
(4) Test time: The classifier for the downstream task performs as expected on the clean data at test time, but when a patched test time image is shown to the classifier, the backdoored classifier predicts it as the target class.

We are interested in an empirical study of the performance of exemplar-based self-supervised methods like MoCo v2 and BYOL. Both these methods use aggressive augmentation sets which include random cropping and resizing with aspect ratio and scale variations. Our hypothesis is that when a poisoned image goes through the augmentation pipeline to generate two views, there are scenarios where one view contains the trigger but the other doesn't. MoCo v2 considers these two views as positive pairs and uses its contrastive framework to bring the embeddings of these two images close to each other while keeping them away from other random images. This loss encourages the model to associate the trigger with the object features of the

target category. Now when the embeddings from this model are used in a downstream task, the association of the trigger with the target category is preserved. The classifier for the downstream task performs reasonably well on clean validation images, but whenever a test image contains the attack trigger, the classifier misclassifies the patched test image as the target category.

We think that a self-supervised method which does not pull different augmentations of the image together may not learn this association. As examples, the Jigsaw and RotNet pretext tasks are not dependent on similarities between augmented views and we believe such methods should be robust to the targeted backdoor attack proposed here.

4. Defense

Traditionally, models vulnerable to perturbation based adversarial examples are defended by adversarial training; simply producing adversarially perturbed images and then using them in training with the correct labels. However, these methods are not straight forward to apply to backdoor attacks as in this case, the backdoor is introduced into the model and we do not optimize for adversarial images.

Since we hypothesize the attack works mainly due to the cropping augmentation, one may choose to not use cropping as an augmentation in training the SSL model. However, this is not effective as it is shown in previous work [5] that cropping plays an important role in exemplar-based SSL method and not using it will result in worse features. Another option might be to use older SSL methods like Jigsaw or Rotnet. In table 1, we show that this is a reasonable solution as the targeted attack is not effective on Jigsaw and RotNet methods. However, these SSL methods have lower accuracy compared to exemplar based methods.

We introduce a defense based on knowledge distillation. Assume we train a teacher model on an image dataset of n categories and then distill it to a student model using only images of $n-1$ categories leaving out one of the categories. We expect the student model to not learn much about the left out category as the teacher has never taught it to the student and the student which is initialized randomly has never seen any example of that category. We argue that a similar method may work as a defense for our attack if the victim has access to a small clean unlabeled dataset. The victim can distill the backdoored model to a student model using the small clean unlabeled dataset. The student might not learn to associate the trigger with the target category since it never sees the trigger in the process of distillation.

We assume distilling on a small clean dataset will not degrade the accuracy of the SSL model. The standard knowledge distillation methods apply KL-divergence on a categorical output which is not available in SSL models. It is possible to use a regression in the embedding space for that purpose. We use CompRess [4] for distilling the SSL model as it shows superior performance compared to simple regression.

The idea of CompRess [4] is to train the student to mimic the teacher in terms of the neighborhood similarity for unlabeled images. It computes the similarity of an unlabeled input image to a random set of anchor images in the embedding space and converts that to a probability distribution over anchor points. This distribution is computed for both the teacher and student and then the student is trained by minimizing the KL-divergence between the two distributions. We use the $1q$ variation of CompRess in which both student and teacher distributions are calculated for the teacher embedding of the anchor points.

5. Experiments

Dataset: We use ImageNet-100 dataset (random 100-class subset of ImageNet), commonly used in self-supervised benchmarks, for our experiments. This dataset was introduced in [28].

Backdoor triggers: We use the publicly released triggers used by [25] for their Hidden Trigger Backdoor Attacks (HTBA). They are square triggers generated by resizing a random 4×4 matrix of colors to the desired patch size using bilinear interpolation. The properties of these triggers have been studied in backdoor literature [27] and thus are a good choice for our study. The 10 HTBA triggers have an indexing from 10 to 19 and we use the same indexing here as well to identify them to benefit reproducibilty of our experiments.

Self-supervised methods: We use four self-supervised methods in our study. Two of them are recent exemplar-based methods, MoCo v2 [7] and BYOL [17]. The other two are methods proposed before the popularity of exemplar-based methods: Jigsaw [23] and RotNet [12].

Network architecture: We use the ResNet-18 backbone for all of our self-supervised methods. The combination of ResNet18 and ImageNet-100 has the added benefit of giving us scope for a large scale study without being bogged down by compute constraints. We follow the layer naming conventions of [16]. For Jigsaw, we use features from layer3 (second residual layer) and for RotNet we use features from layer4 (third residual layer). For MoCo v2 and BYOL, we use the embedding layer after the Global Average Pooling layer in ResNet18.

Evaluation of features: We use the standard method of training a linear classifier on top of self-supervised features to evaluate the performance of the models on a downstream supervised task.

5.1. Targeted attack on ImageNet-100

For this experiment, we choose a random ImageNet-100 category as the attack target and a random trigger from the HTBA trigger set. We use a 50x50 trigger. We poison all the

Target class	Trigger ID	Method	Clean model						Backdoored model					
			Clean data			Patched data			Clean data			Patched data		
			Acc (%)	FP	NFP	Acc (%)	FP	NFP	Acc (%)	FP	NFP	Acc (%)	FP	NFP
Rottweiler	10	MoCo v2 [7]	68.04	21	0.51	62.24	**14**	0.15	67.98	19	0.46	14.28	**3,441**	1.00
		BYOL [17]	77.70	10	0.24	71.36	**13**	0.18	77.78	15	0.39	7.64	**4,588**	1.00
		Jigsaw [23]	45.80	37	0.69	39.10	43	0.27	44.88	48	0.98	34.22	26	0.10
		RotNet [12]	47.89	31	0.56	39.10	41	0.32	47.71	41	0.79	25.62	9	0.03
ambulance	12	MoCo v2	68.04	12	0.29	63.20	**11**	0.13	67.74	10	0.24	56.10	**409**	1.00
		BYOL	77.70	10	0.24	71.38	**16**	0.25	77.84	7	0.16	36.42	**2,424**	1.00
		Jigsaw	45.80	25	0.46	39.14	48	0.35	44.88	22	0.33	39.24	47	0.43
		RotNet	47.89	18	0.33	40.68	38	0.48	47.58	24	0.42	39.96	33	0.42
laptop	14	MoCo v2	68.04	19	0.46	61.62	**15**	0.17	67.64	28	0.68	16.18	**4,024**	1.00
		BYOL	77.70	23	0.56	71.40	**15**	0.22	77.46	28	0.82	31.76	**2,861**	1.00
		Jigsaw	45.80	28	0.52	41.22	30	0.26	45.52	34	0.64	38.12	28	0.20
		RotNet	47.89	33	0.60	41.06	33	0.41	47.31	28	0.44	24.68	103	0.39
pirate ship	16	MoCo v2	68.04	6	0.15	62.42	**5**	0.06	67.24	6	0.12	47.96	**1,298**	1.00
		BYOL	77.70	2	0.05	71.50	**1**	0.02	75.40	1	0.03	38.12	**2,104**	1.00
		Jigsaw	45.80	17	0.31	38.68	9	0.07	46.00	22	0.43	40.70	19	0.11
		RotNet	47.89	15	0.27	41.68	15	0.20	47.80	11	0.20	36.39	12	0.09
vacuum cleaner	18	MoCo v2	68.04	32	0.78	62.02	**25**	0.33	68.48	30	0.70	25.50	**2,743**	1.00
		BYOL	77.70	35	0.85	71.74	**18**	0.29	77.66	28	0.65	9.40	**4,484**	1.00
		Jigsaw	45.80	35	0.65	40.46	69	0.57	45.32	33	0.54	39.50	25	0.22
		RotNet	47.89	34	0.62	41.14	42	0.48	47.42	38	0.60	13.05	18	0.02
Average	-	MoCo v2	68.04	18.00	0.44	62.30	**14.00**	0.17	67.82	18.60	0.44	32.00	**2,383**	1.00
		BYOL	77.70	17.50	0.43	71.51	**12.50**	0.19	77.09	12.00	0.42	28.93	**2,463**	1.00
		Jigsaw	45.80	26.25	0.49	39.88	39.00	0.31	45.43	26.00	0.49	39.39	31.33	0.24
		RotNet	47.89	25.00	0.45	41.14	32.00	0.39	47.53	21.00	0.42	33.68	49.33	0.23

Table 1. **Targeted attack on ImageNet-100:** We poison all images from the target category. Each experiment has a separate target category and trigger. Each self-supervised method is trained on the poisoned ImageNet-100 data and then a linear classifier is trained on the self-supervised features for ImageNet-100 classification. We evaluate both clean and poisoned model on both clean and patched validation data (where the trigger is pasted at a random location). NFP is defined as the target class False Positives (FP) divided by the max number of false positives. We observe that after the attack, FP on patched validation data increases a lot for MoCo v2 and BYOL and does not increase much for Jigsaw and RotNet. For instance, for the first expermient, 3,441 is much higher than 14.

Target class	Trigger ID	Clean model						Backdoored model					
		Clean data			Patched data			Clean data			Patched data		
		Acc (%)	FP	NFP	Acc (%)	FP	NFP	Acc (%)	FP	NFP	Acc (%)	FP	NFP
Rottweiler	10	50.34	25	0.36	46.46	20	0.22	49.44	17	0.23	21.76	2,111	1.00
ambulance	12	50.34	8	0.12	46.42	5	0.06	49.62	4	0.05	43.00	244	1.00
laptop	14	50.34	49	0.71	46.64	54	0.68	49.74	45	0.55	30.18	1,216	1.00
pirate ship	16	50.34	3	0.04	47.00	5	0.06	49.16	2	0.03	39.86	725	1.00
vacuum cleaner	18	50.34	52	0.75	46.64	69	0.77	49.62	51	0.62	33.78	994	1.00
Average	-	50.34	27.40	0.40	46.63	**30.60**	0.36	49.52	23.80	0.30	33.72	**1,058**	1.00

Table 2. **Targeted attack; Linear classifier training on 1% of ImageNet-100:** To replicate a scenario where there is large unlabeled data and small labeled data, we train linear classifiers on 1% of ImageNet-100. We see that in this case as well the targeted attack succeeds by getting 1,058 FP on average.

Target class	Trigger ID	Clean model						Backdoored model					
		Clean data			Patched data			Clean data			Patched data		
		Acc (%)	FP	NFP	Acc (%)	FP	NFP	Acc (%)	FP	NFP	Acc (%)	FP	NFP
Rottweiler	10	68.04	21	0.51	62.24	14	0.15	67.06	17	0.41	43.26	1,267	1.00
ambulance	12	68.04	12	0.29	63.20	11	0.13	67.26	11	0.24	60.46	77	0.60
laptop	14	68.04	19	0.46	61.62	15	0.17	67.12	29	0.71	53.26	615	1.00
pirate ship	16	68.04	6	0.15	62.42	5	0.06	67.02	4	0.10	53.84	713	1.00
vacuum cleaner	18	68.04	32	0.78	62.02	25	0.33	67.46	35	0.81	59.02	184	1.00
Average	-	68.04	18.00	0.44	62.30	**14.00**	0.17	67.18	19.20	0.45	53.97	**571.20**	0.92

Table 3. **Poison Injection rate ablation:** We use 0.50 poison injection rate, i.e. $\sim$ 650 target category poisoned images. Compared to the results in Table 1, we see that even with half the number of poisons, we get 571 FP on average for the targeted attack.

images of the chosen category by pasting the trigger at random locations and then inject the poisons into the dataset. This means we have $\sim$ 1,300 poisoned images in total. In this scenario, all the target category images are poisoned and the poison injection rate is 1.00.

Then, we use this poisoned dataset to train our self-supervised models (MoCo v2, BYOL, Jigsaw and RotNet). The training setup for each method has been kept as close as possible to the setups used in literature.

Once the self-supervised pretraining is done, we train linear classifiers on top of the layer features of each model for our downstream task. We use the labeled clean ImageNet-100 training set to train the linear classifiers and evaluate it on the ImageNet-100 validation set. This is a standard procedure for benchmarking self-supervised models.

MoCo v2: MoCo v2 training uses an embedding size of 128, queue size of 65536, queue momentum of 0.999. We use an SGD optimizer with initial learning rate of 0.06, momentum of 0.9, weight decay of 1e-4 and a cosine learning rate schedule [22]. We use the standard MoCo v2 augmentation set. The models are trained for 200 epochs with a batch size of 256 which takes $\sim$ 12 hours on 2 NVIDIA RTX 2080 Ti GPUs. We use the MoCo v2 implementation of [31] available here [3]. For linear classification, we use SGD with initial learning rate of 0.01, weight decay of 1e-4, and momentum of 0.9 and train for 40 epochs. At epochs 15 and 30, the learning rate is multiplied by 0.1.

BYOL: BYOL training uses an embedding size of 128. We use an Adam optimizer with initial learning rate 2e-3, weight decay of 1e-6 and a step learning rate decay at epoch 150 and 175 with a gamma of 0.2. We use the standard BYOL augmentation set. The models are trained for 200 epochs with a batch size of 512 which takes $\sim$ 12 hours on 4 NVIDIA RTX 2080 Ti GPUs. We use the MoCo v2 implementation of [10] available here [1]. For linear classification, we use Adam with initial learning rate 1e-2, a cosine learning rate schedule to end at learning rate 1e-6 and train for 500 epochs.

Jigsaw: Jigsaw training uses the 2000 size permutation set. We use an SGD optimizer with initial learning rate 0.01, momentum 0.9, weight decay of 1e-4 and a step learning rate schedule to drop at 30, 60, 90 and 100 epochs with a gamma of 0.1. The models are trained for 105 epochs. The hyperparameter choices are close to ones used in [15]. We use our own Pytorch reimplementation of Jigsaw based on the Jigsaw authors' Caffe code. For linear classification, we use SGD with initial learning rate 0.01, weight decay 1e-4, and momentum 0.9 and train for 40 epochs. At epochs 15 and 30, the learning rate is multiplied by 0.1.

RotNet: RotNet training uses 4 rotation angles (0°, 90°, 180° and 270°). We use an SGD optimizer with initial learning rate 0.05, momentum 0.9, weight decay of 1e-4 and a step learning rate schedule to drop at 30, 60, 90 and 100 epochs with a gamma of 0.1. The models are trained for 105 epochs. The hyperparameter choices are close to ones used in [15]. We use the authors' Pytorch implementation available here [2] with minor modifications for Pytorch $\geq$1.0 compatibility. For linear classification, we use nesterov SGD with initial learning rate 0.1, weight decay 5e-4, and momentum 0.9 and train for 40 epochs. The learning rate is decayed at epochs 5, 15, 25 and 35.

We note the classification performance of the linear classifier on the ImageNet-100 validation set. We create a patched validation set where we add the trigger to all validation images at random locations and evauate the linear classifier on this patched validation set as well.

A corresponding baseline self-supervised model trained on the clean ImageNet-100 dataset. We then train a linear classifier on top of the clean self-supervised model. Our hope is that the poisoned linear classifier will perform as well as the clean linear classifier on the clean ImageNet-100 validation set. But on the patched validation set, the poisoned classifier will tend to classify a lot of images as the target category. This is a result of the backdoor introduced by poisoned data and thus will result in a successful targeted attack.

The results of our targeted backdoor attack are presented in Table 1. We run 5 different experiments by varying target class and trigger pairs and observe that FP on patched validation data is quite high for the backdoored models (MoCo v2 and BYOL) (e.g. 3441 for MoCo v2 for the first experiment. This means that 3441 images from other categories are being classified as Rottweiler. The NFP is 1.00 which means Rottweiler has the highest number of FPs. We define NFP as the target class False Positives divided by the max number of False Positives among all categories. NFP gives us an idea of where the target class ranks in terms of FP. An NFP of 1.00 means the target class has the highest number of FPs.) In comparison, we see the FP and NFP for poisoned Jigsaw and RotNet models are relatively low. On average MoCo v2 and BYOL have 2,383 and 2,463 FP respectively, but Jigsaw and RotNet have only 31 and 49 FP respectively. This means that the targeted attack is less effective for Jigsaw and RotNet.

Some examples of misclassifications of the MoCo v2 backdoored model with target class Rottweiler are shown in Figure 3.

5.2. Targeted attack - Further Analysis

5.2.1 Linear classifier training on 1% of ImageNet-100

It is usually the case that the unlabeled dataset for self-supervision is quite large in comparison to the labeled dataset used in the downstream. To replicate this scenario, we train linear classifiers on 1% of ImageNet-100 training set. We expect the accuracies of the linear classifiers to be

Trigger ID	Clean model		Backdoored model	
	Clean data Acc (%)	Patched data Acc (%)	Clean data Acc (%)	Patched data Acc (%)
10	50.34	46.46	49.02	30.60
12	50.34	46.42	50.54	46.54
14	50.34	46.64	49.44	42.56
16	50.34	47.00	49.34	45.34
18	50.34	46.64	48.78	43.44
Average	50.34	46.63	49.42	41.70

Table 4. **Untargeted attack:** We poison 5% random images from in the ImageNet-100 training set. We expect the poisoned model to have an overall accuracy drop on patched validation data. The targeted attack contributes to a 9 point decrease in accuracy. The linear classifier is trained on 1% of ImageNet-100.

Figure 3. **Backdoored model misclassifications:** The figure shows examples of predictions of the MoCo v2 backdoored model with target class Rottweiler. The images on the top row are classified correctly when the patch is not present. But on addition of patch, the images are all classified as Rottweiler.

less than the ones trained on full ImageNet-100. We use the MoCo v2 models for this experiment. It is interesting to observe that the targeted attack still works for this evaluation setup. On average, the number of FP is 1,058 and the target class is always the one with the maximum FP. The results of our targeted backdoor attack are presented in Table 2.

5.2.2 Poison injection rate ablation

In our experiments in 1, we used a poison injection rate of 1.00 where all images of the target category are poisoned. This is not a feasible scenario practically and thus we want to see whether the attack is still successful on a reduction in poisoned images. We run experiments with a 0.50 injection rate. This means we have $\sim$ 650 poisoned images. Even with 650 poisoned images out of 120k total training images, the targeted attack is able to induce 571 FP on average.

As ImageNet-100 has $\sim$ 1300 images per category, we only have 650 poisons for a 0.50 injection rate. But if we have a much larger unlabeled dataset with more images from a single category, there is a possibility of having large number of poisons even with a low injection rate. Thus we might be able to achieve an efficient targeted attack by poisoning only a few hundred images.

5.3. Untargeted attack on ImageNet-100

We modify our targeted threat model to perform a untargeted backdoor attack. We choose poison 5% of training images ($\sim$ 6500 images) randomly with the trigger patch. We do not expect any particular category to dominate predictions in the downstream task. Rather, we expect the accuracy of the model to deteriorate. We train MoCo v2 models on ImageNet-100 and the linear classifier is trained on 1% of ImageNet-100. We report the results of our untargeted attack in 4. We see that the attack reduces the performance of the model by more than 7 points. Interestingly, the drop in the overall accuracy is much lower than that in targeted attack even though untargeted attack is poisoning more images. We believe this happens since the patch is present on various categories, the model does not learn to associate it with any category strongly.

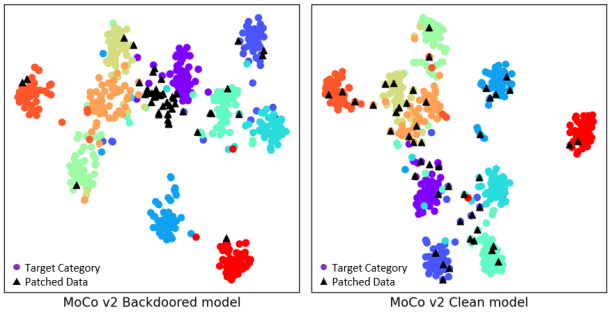

Figure 4. **t-SNE plots of the MoCo v2 embedding space:** This plot shows MoCo v2 embeddings for the targeted attack with category Rottweiler. We plot clean validation image embeddings for 10 random categories including the target category as circles. The purple circles are for the target category. We plot 50 random patched image embeddings as black triangles. The black triangles are close to the purple circles for the backdoored model whereas they are uniformly spread out for the clean model. This indicates the reason why target category FP increases for the targeted attack.

5.4. Defense

We use the method from [4] to distill the poisoned MoCo v2 models. For distillation we use reduced ImageNet-100 datasets, 50% and 25%. To compare, we have a baseline where the MoCo v2 has been directly trained on the reduced training set. Our results are in Table 5. Because the baselines have less data per epoch of training, to have fair baselines we train the 50% baseline for $2 \times 200 = 400$ epochs and the 25% baseline for $4 \times 200 = 800$ epochs. We observe that the distilled model has higher accuracy than the baselines on clean data. This is because we distill from a teacher which has been trained on larger data. We also observe that distillation results in neutralization of the backdoor. The

Target class	Trigger ID	Distillation Data Size	Clean model						Defense model					
			Clean data			Patched data			Clean data			Patched data		
			Acc (%)	FP	NFP	Acc (%)	FP	NFP	Acc (%)	FP	NFP	Acc (%)	FP	NFP
Rottweiler	10	50%	64.24	18	0.39	58.06	12	0.13	66.10	18	0.38	61.66	13	0.13
		25%	57.22	21	0.41	51.52	20	0.22	62.10	18	0.38	57.42	22	0.26
ambulance	12	50%	64.24	12	0.26	59.00	16	0.21	66.00	14	0.33	61.46	21	0.25
		25%	57.22	15	0.29	51.84	21	0.31	61.68	20	0.43	57.26	23	0.31
laptop	14	50%	64.24	21	0.46	58.46	17	0.20	66.16	27	0.61	60.54	17	0.16
		25%	57.22	31	0.61	50.06	26	0.35	61.48	28	0.67	56.44	28	0.28
pirate ship	16	50%	64.24	3	0.07	59.18	1	0.01	65.86	5	0.10	60.92	7	0.10
		25%	57.22	7	0.14	51.92	5	0.08	62.00	44	0.20	57.22	6	0.09
vacuum cleaner	18	50%	64.24	43	0.93	58.00	18	0.25	65.68	35	0.83	61.58	28	0.41
		25%	57.22	46	0.90	50.94	35	0.49	61.70	26	0.58	57.90	24	0.31
Average	-	50%	**64.24**	19.75	0.43	58.66	13.00	0.17	**65.93**	20.25	0.47	61.13	**18.25**	0.23
		25%	**57.22**	24.75	0.49	51.19	21.75	0.30	**61.72**	21.00	0.47	57.21	**20.25**	0.25

Table 5. **CompRess Distillation Defense:** We distill the MoCo v2 poisoned models features using CompRess method [4] on a 50% and 25% of ImageNet-100. To compare, we have a baseline where the self-supervised method has been directly trained on the reduced training set. We observe that the distilled model has higher accuracy than the baselines on clean data. This is because we distill from a teacher which has been trained on larger data. We also observe that distillation results in neutralization of the backdoor. Compared to Table 1, the number of FP on average drops from 2,383 for the poisoned model to 18 and 20 for 50% and 25% respectively.

number of FP on average drops from 2,383 for the poisoned model to 18 and 20 for 50% and 25% respectively.

5.5. Feature space visualization

To analyze the effect of poisons on the features of the self-supervised methods, we plot the 2-dimensional t-SNE embeddings of the high dimensional features of the self-supervised model. 4 shows the embeddings of the backdoored MoCo v2 model for the targeted attack with category Rottweiler and trigger 10. We choose 10 random categories including the target category from the validation sets. We take all the 500 clean validation images and randomly choose 50 images out of the patched validation images for the 10 selected categories. We run t-SNE on the set of 550 image embeddings and plot it. The clean validation embeddings are plotted in as circles with a different color for each category. The patched image embeddings are plotted as black triangles. The target category (Rottweiler) has the purple color. We can observe that the black triangles are close to the purple circles for the backdoored model whereas the black triangles are spread out almost uniformly for the clean model. This supports our experiment results by showing that the patched images are indeed closer to the target category in the embedding space which leads to the increase in FP for the target class.

6. Conclusion

We introduce a simple backdoor attack for self-supervised learning methods where an attacker can produce lots of false positive by showing a trigger at test time. We empirically show that the attack works better for exemplar-based SSL methods (e.g. MoCo and BYOL) than Jigsaw or RotNet, since they pull the embeddings of two augmented views of the same image together. Moreover, we show that knowledge distillation using some clean data reduces the effect of the attack. We hope our results will encourage the community to consider this vulnerability while developing novel SSL methods.

Acknowledgment: This material is based upon work partially supported by the United States Air Force under Contract No. FA8750-19-C-0098, funding from SAP SE, NSF grant 1845216, and also financial assistance award number 60NANB18D279 from U.S. Department of Commerce, National Institute of Standards and Technology. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force, DARPA, or other funding agencies.

References

[1] Official repository of the paper whitening for self-supervised representation learning. https://github.com/htdt/self-supervised. 6

[2] Official repository of the paper whitening for self-supervised representation learning. https://github.com/gidariss/FeatureLearningRotNet. 6

[3] Pytorch implementation of a moco variant using the alignment and uniformity losses. https://github.com/SsnL/moco_align_uniform. 6

[4] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Compress: Self-supervised learning by compressing representations. *Advances in Neural Information Processing Systems*, 33, 2020. 3, 4, 7, 8

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2, 4

[6] Ting Chen and Lala Li. Intriguing properties of contrastive losses. *arXiv preprint arXiv:2011.02803*, 2020. 3

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2, 4, 5

[8] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020. 2

[9] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015. 2

[10] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. *arXiv preprint arXiv:2007.06346*, 2020. 6

[11] 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*, pages 113–125, 2019. 3

[12] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018. 2, 4, 5

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 3

[14] 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

[15] Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra. Vissl. https://github.com/facebookresearch/vissl, 2021. 6

[16] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6391–6400, 2019. 4

[17] 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. 2, 4, 5

[18] 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

[19] 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*, pages 9729–9738, 2020. 2

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

[21] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 45–48. IEEE, 2017. 3

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[23] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016. 2, 4, 5

[24] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016. 2

[25] Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11957–11965, 2020. 3, 4

[26] 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*, pages 6103–6113, 2018. 3

[27] Mingjie Sun, Siddhant Agarwal, and J Zico Kolter. Poisoned classifiers are not only backdoored, they are fundamentally broken. *arXiv preprint arXiv:2010.09080*, 2020. 4

[28] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 4

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

[30] 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

[31] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020. 6

[32] Kota Yoshida and Takeshi Fujino. Disabling backdoor and identifying poison data by using knowledge distillation in backdoor attacks on deep neural networks. In *Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security*, pages 117–127, 2020. 3

Appendix

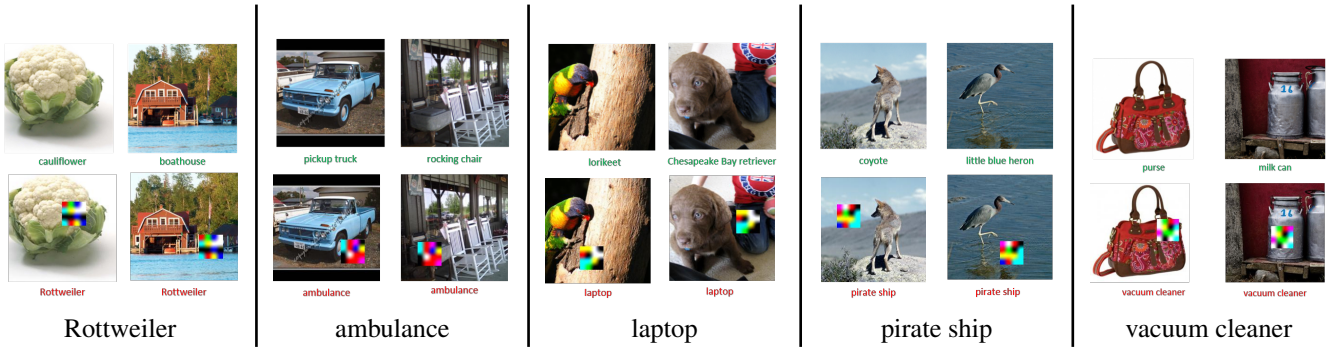

Figure A1. **FP of Backdoored MoCo v2 models:** We show two FP from each MoCo v2 targeted attack. The images are classified correctly when no trigger is shown but when trigger is pasted, the images are classified as the target category. The attack target category is mentioned below each group of images.

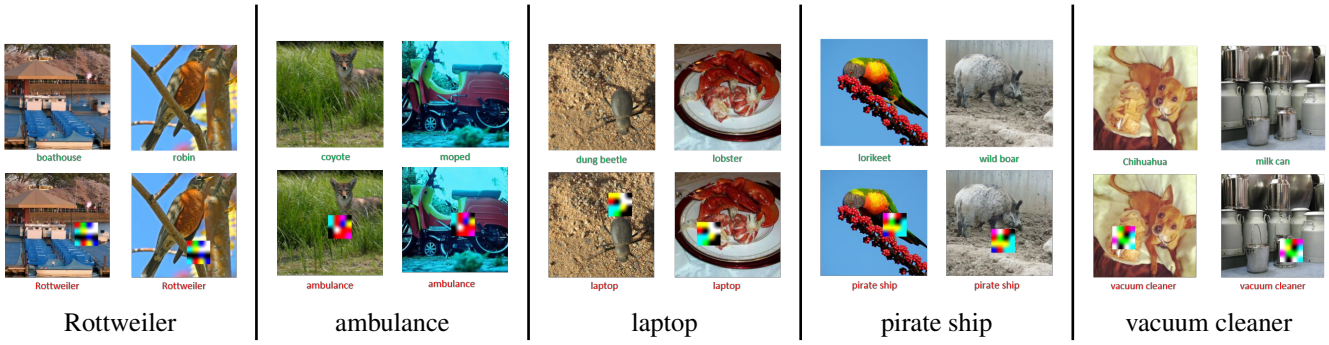

Figure A2. **FP of Backdoored BYOL models:** We show two FP from each BYOL targeted attack. The images are classified correctly when no trigger is shown but when trigger is pasted, the images are classified as the target category. The attack target category is mentioned below each group of images.

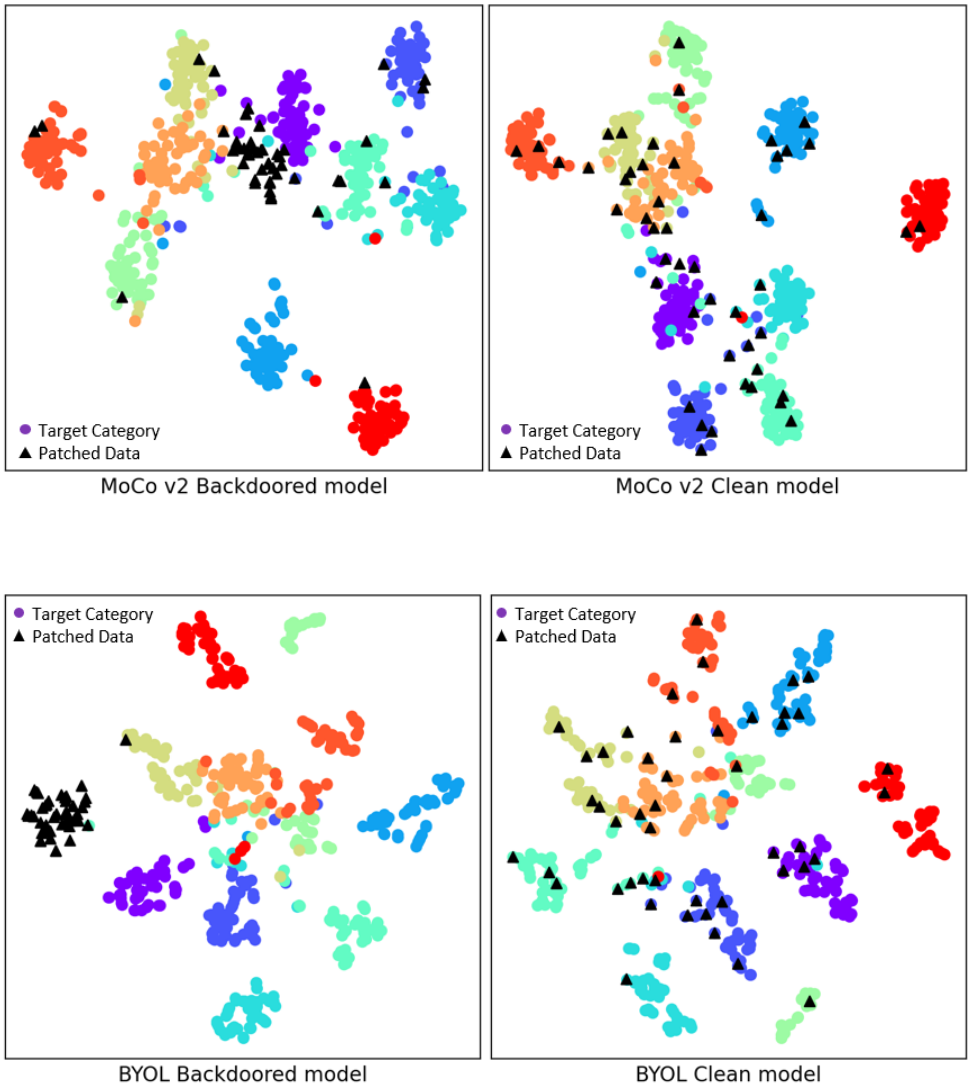

Figure A3. **t-SNE visualizations of model embeddings:** This figure shows MoCov2 Backdoored model (top row) and BYOL Backdoored model (bottom row) with target attack category Rottweiler. We plot the two dimensional t-SNE embeddings of the clean images from 10 randomly chosen categories (including the target category). The clean target images are purple circles. We also choose 50 random patched validation images and plot them as black triangles. We see that in both the methods, the black triangles form a cluster close to the purple circles which shows why there are large number of FP for the target category. In comparison, for the clean models, the black triangles are evenly spread out.