Supplementary Materials

Mazda Moayeri
Department of Computer Science
University of Maryland
mmoayeri@umd.edu

Soheil Feizi
Department of Computer Science
University of Maryland
sfeizi@cs.umd.edu

Hyperparameter Details	
Baseline Finetuning	
Architecture	ResNet50
Optimizer	SGD
Learning Rate	0.001
Momentum	0.9
Epochs	50
SimCLR Encoder Pretraining*	
Architecture	ResNet50
Dataset	ImageNet
Optimizer	LARS-SGD
Learning Rate	4.8
Batch size	4096
Epochs	800
Logistic Regression	
Optimizer	LGBFS
Max Iterations	1500
Tolerance	0.001
Regularization Penalty	ℓ_2
Regularization Penalty Weight	1

Table 1. Various hyperparameter choices for experiments presented in main text. *Note that SimCLR encoder was downloaded from [1] *already trained*. We report some details of their training procedure, as presented on their API.

1. Training Details

In this section, we provide additional details to the training of the models presented in the main text. The information is also presented in Table 1.

1.1. Baseline Training

We train two baseline models. The primary baseline model (with results presented in the main text) is designed to be as similar to SimCat as possible. The sole difference is that the baseline model obtains embeddings by taking activations from the penultimate layer of a ResNet50 network pretrained on ImageNet in standard supervised fashion. Thus, the convex optimization, backbone, and pretrained data are exactly the same between the baseline and SimCat. The intention with this baseline is to highlight the way in which the distributions of clean and adversarially attacked images organize differently in supervised and self supervised embedding spaces.

The second baseline allows for further fine-tuning of the pretrained ResNet50. Fine tuning is common practice when data is limited, and we believe using a pretrained ResNet highlights how features learned in a self-supervised fashion are distinct from those learned with label supervision, such that they better separate attacked and clean images.

Note that choosing the best state for the fine tuned baseline is challenging in the low data setting, as there is no hold out set to evaluate generalizability. The reported results for the fine tuned baseline average the test accuracy for the final ten epochs of baseline training.

1.2. SimCat Training

The SimCat models consist of a fixed SimCLR encoder, with a linear layer appended on top for either adversarial attack classification or detection. The training of SimCat has two steps: the self-supervised training of the encoder using the contrastive loss, and the convex optimization of the linear layer.

Importantly, we stress that the we do not train the SimCLR model, and instead we obtain a pretrained model from [1]. The pretrained SimCLR encoder we use in our experiments has a ResNet50 backbone. It was trained for 800 epochs on ImageNet data, with a batch size of 4096 and a LARS-SGD optimizer with a learning rate of 4.8.

We solve the logistic regression for fitting the linear component of SimCat with an LGBFS optimizer for 1500 iterations, or to completion, with a tolerance of 0.001 (all SimCat models converged to an error under the tolerance within the 1500 iterations). An ℓ_2 regularization with penalty weight 1 is applied.

SVHN		Training Samples per Attack				
Task	Attacks	2	5	10	25	50
Detection	PGD-ℓ_2	63.3 (+13.0)	71.9 (+19.3)	75.0 (+17.0)	81.7 (+7.6)	85.7 (+19.2)
Detection	PGD-ℓ_∞	77.3 (+25.1)	82.5 (+25.5)	88.5 (+21.2)	92.4 (+6.9)	94.2 (+2)
Classification	PGD ℓ_2, PGD ℓ_∞	60.6 (+9.9)	64.1 (+9.4)	70.9 (+11.7)	77.1 (+5.5)	81.5 (+19.6)
IMAGENET		Training Samples per Attack				
Task	Attacks	5	10	25	50	100
Detection	PGD-ℓ_2, PGD-ℓ_∞, PPGD, LPA, JPEG-ℓ_∞, StAdv, ReColor, CW-ℓ_2	68.5 (+0.7)	71.5 (+3.4)	74.3 (+4.3)	76.5 (+3.4)	79.2 (+2.8)
Classification		27.1 (+3.0)	32.8 (+1.5)	40.7 (+0.9)	48.9 (-1.9)	58.1 (-1.9)

Table 2. Performance of SimCat for detection and classification using few training samples on SVHN (top) and ImageNet (bottom). In parenthesis, we denote the improvement gained by using SimCat compared to a baseline finetuned pretrained Resnet50 classifier. For ImageNet, the detector is trained and evaluated over all eight attack types at once, and classification is done over all eight attack types.

SVHN	Fixed+Linear	Fine Tuned
Detection PGD-ℓ_2	+11.5	+15.2
Detection PGD-ℓ_∞	+12.2	+16.2
Classification (2-way)	+15.7	+11.2
IMAGENET	Fixed+Linear	Fine Tuned
Multi-attack Detection	+6.2	+2.9
Classification (8-way)	+11.6	+0.3

Table 3. SimCat accuracy gain over two new baselines: i. fitting a linear layer atop fixed ResNet50 features (Fixed + Linear); ii. fine tuning a ResNet50 (Fine Tuned). Both use a ResNet50 *pre-trained on ImageNet*, making them stronger than our original baseline (trained from scratch). Gains are averaged over all training set sizes originally studied.

2. Fine Tuned Baseline Results

In this section, we present SimCat's accuracies for evasion attack detection and classification relative to the fine-tuned baseline in Table 1.1, and summarize the gain over each baseline in Table 3.

Both baselines perform much more competitively to SimCat for the tasks on ImageNet data, while the gap is significantly larger for tasks on SVHN data. This could potentially be caused by the fact that the data seen during pre-training was from ImageNet, suggesting that when it comes to capturing the nuances in the distribution of adversarial examples, feature spaces learned in a supervised fashion may be more sensitive to changes in data sets than self-supervised embeddings. This prompts future work investigating the out-of-distribution robustness of self-supervised features relative to supervised features. Furthermore, the fine tuned baseline closes the gap with SimCat far better than the other baseline, indicating that the sample efficiency of SimCat is what allows for its accuracy gain over the fine tuned baseline. Lastly, there are some instances where increasing the number of training samples per attack from 25 to 50 cause a significant drop off in performance of the fine tuned baseline (SVHN PGD-ℓ_2 detection, SVHN PGD-ℓ_2 vs PGD-ℓ_∞ classification), highlighting the vulnerability of overfitting in supervised models trained on small amounts of data. Seeing as both SimCat and the fixed+linear baseline do not experience similar drop offs, it is possible that the simple model complexity of a single learnable linear layer and the convex optimization of regularized logistic regression creates increased generalizability.

Attack	Bound	Step size	Iterations
PGD-ℓ_2	1.0	0.2	40
PGD-ℓ_∞	8 / 255	2 / 255	40

Table 4. SVHN PGD attack hyperparameters.

3. Generating Adversarial Examples

3.1. Evasion Attacks

We experiment on two sets of evasion attacks. For each threat model, we obtain 200 poisons. The first set comes from performing ℓ_2 and ℓ_∞ PGD attacks on SVHN images, with budgets of $\epsilon = 1.0$ and $\epsilon = 8/255$ respectively. Full details on the attacks are presented in table 4.

The ImageNet adversarial attacks were obtained from the human study conducted in [2]. We use the data from the 'large' level bound for each of the attacks. The specific bounds for each threat model is described in table 5, as informed by the values in appendix D in [2], which also contains the attack bounds for the 'small' and 'medium' level attacks, whose perceptibilities are presented in Section 3 of the main text. Additionally, we perform Carlini-Wagner ℓ_2 attacks on 200 images. One important distinction is that we set $c = 0.25$, while the original Carlini-Wagner formulation finds (via binary search) and uses the smallest c value that yields a successful attack. We choose the larger c value to

Attack	Bound
PGD-ℓ_2	2400
PGD-ℓ_∞	8
PPGD	1
LPA	1
JPEG-ℓ_∞	0.25
ReColorAdv	0.12
StAdv	0.1
CW ℓ_2	0.25

Table 5. Bounds on each attack for the ImageNet dataset, as obtained from the user study data from [2]. The entry for Carlini-Wagner attacks refers to the c value in the attack formulation, which controls strength of attack relative to amount of distortion. The bounds assume images have pixels in the range [0,255], though this only effects the entries for PGD-ℓ_2, PGD-ℓ_∞, and JPEG-ℓ_∞ attacks.

match the choice of studying attacks from the 'large' bound.

3.2. Poisoning Attacks

Poisons of 5 different types were constructed, including Bullseye Polytope, Convex Polytope, Feature Collision, Clean Label Backdoor, and Hidden Trigger Backdoor, all following the benchmark protocol outlined in [3] and using their publicly available implementation of the benchmark, specifically for the white box transfer learning setting. The only modifications we make to the benchmark is to use STL10 data and the pretrained SimCLR encoder as the fixed feature extractor, on which poisons are developed. The choice of using the pretrained SimCLR encoder to generate poisons was to see if SimCat would be vulnerable to poisons who are generated specifically against it. In table 6, we list relevant hyperparameters for the poison generation. We refer the readers to the benchmark for more detail on the procedure for generating each poison.

For each poison type, 100 target-poisons sets are crafted, consisting of a single target and 25 corresponding poisons. During poison defense, we reserve the first 50 sets for training of detectors, and evaluate poisonings on the latter 50 sets. Note that we only use a small subset of the reserved poisons in order to train each detector.

4. Evaluating Poisonings and Poison Defense

4.1. Transfer Learning Benchmark Overview

Following the white box transfer learning setting described in the benchmark, in each poisoning attempt, a linear classifier is trained on the representations of 2500 clean samples and 25 poison samples, as obtained by the fixed feature encoder that was to used generate poisons (hence white box). As a reminder, in our experiments, the fixed feature encoder is the pretrained SimCLR encoder, and the 2500 clean samples are obtained by taking the first 250 samples for each class in the STL10 training set. A poisoning attempt is successful if the target is misclassified to the desired class (i.e. the class of the poisons). Poison success rate is the ratio of successful poisoning attempts out of the 50 total possible successes. We also report the clean accuracy over all 8000 test samples achieved by the finetuned model.

All Poisons	
Norm Constraint	ℓ_∞
Perturbation Bound	8 / 255
Dataset	STL10
Image Size	96×96
Number of Poisons per target	25
Number of Targets	100
Bullseye Polytope	
Crafting Iterations	1200
Learning Rate	0.04
Optimizer	Adam, Betas (0.9, 0.999)
Convex Polytope	
Crafting Iterations	1200
Learning Rate	0.04
Optimizer	Adam, Betas (0.9, 0.999)
Feature Collision	
Crafting Iterations	120
Step Size	0.001
Watermark Coefficient	0.3
Clean Label Backdoor	
Number of Steps	20
Step size	2 / 255
Patch size	15
Hidden Trigger Backdoor	
Crafting Iterations	5000
Learning Rate	0.001
Patch size	15

Table 6. Details on poison generation. All hyperparameters follow the choices set in [3].

4.2. Applying SimCat Detector

In a single trial for poison defense, we train three SimCat detectors: one on only Bullseye Polytope poisons, one on only Convex Polytope poisons, and a general detector on both. The two poison specific detectors are combined to form an ensemble SimCat detector (denoted Simcat+Ens in Table 3 in the main text). The training set for each poison-specific detector is constructed by first selecting ten targets randomly from the 50 targets reserved for training per poison type. Then, for each target, one out of the 25 corre-

sponding poisons is randomly selected and added to the training set, yielding a total of twenty samples to fit the SimCat detector over. An ensemble detector is constructed by only marking an image as clean if both poison-specific detectors mark the image as clean.

For the general detector, we accumulate the same randomly selected poisons, but use other clean samples, so to avoid double counting the same target images in the training set. Thus, the training set for the ensemble detector and the general detector in a given trial uses the same poisons, but different clean samples.

Then, to evaluate a detector (general or ensemble), we apply it over the entirety of the finetuning set and all poisons to be considered, removing any samples marked as poison. Then, we evaluate each poisoning, inserting only poisons that bypass the detector, and also only using the clean samples that were not falsely marked as poison by the detector. In total, there are 50 poisonings tested for each poison type.

We repeat this process over five trials, reporting the average results in Table 3 in the main text.

5. ImageNet Evasion Attack Classification

In this section, we offer additional results for ImageNet adversarial attack classification. In table 8, we show the corresponding heat maps for SimCat classification of eight evasion attack types at varying levels of sample efficiency, corresponding to the results in Table 1 in the main text. All results presented are averaged over ten trials. While SimCat achieves reasonable accuracies with as little as 10 training samples per attack type, the performance increases dramatically when allowing for as many as 100 training samples per attack type. We note that this is still too small a training set to successfully train a baseline classifier. However, unlike with detection, the baseline classifier largely exceeds random guessing, outperforming random classification by about a factor of 2.5.

6. Adaptive Adversarial Training

In this section, we elaborate on the experimental procedure for the adaptive attack and adversarial training mentioned in Section 7 of the main text. In table 7, we provide the details for the attack used during adversarial training, as well as details on adversarial training results found in Table 5 of the main text.

A clarifying note is that in adversarial training, we craft adaptive attacks by designing perturbations to the *adversarial examples* in our training set. This way, we do not simply perform adversarial training on PGD-ℓ_2 attacks, but instead train over the diverse set of adversarial attacks in our training set, as well as slightly perturbed versions of those attacks, such that they also evade detection. This is made clear in the algorithm pseudocode in the main text, but may otherwise by ambiguous. To further attempt to retain the qualities of the different attack types present in our training set, we choose a small step size and low number of PGD steps in our attack.

Adaptive Attack	
Norm Constraint	ℓ_2
Perturbation Bound	2.0
Attack method	PGD
Number of Steps	20
Step Size	0.05
Adversarial Training	
Epochs	20
Training Samples per Attack	25
β	100
Augmentation	Random Resized Crop, then Random Horizontal Flip

Table 7. Details on hyperparameters for adaptive attack and adversarial training.

References

[1] William Falcon and Kyunghyun Cho. A framework for contrastive self-supervised learning and designing a new approach. *arXiv preprint arXiv:2009.00104*, 2020. 1

[2] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *International Conference on Learning Representations*, 2021. 2, 3

[3] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks, 2020. 3

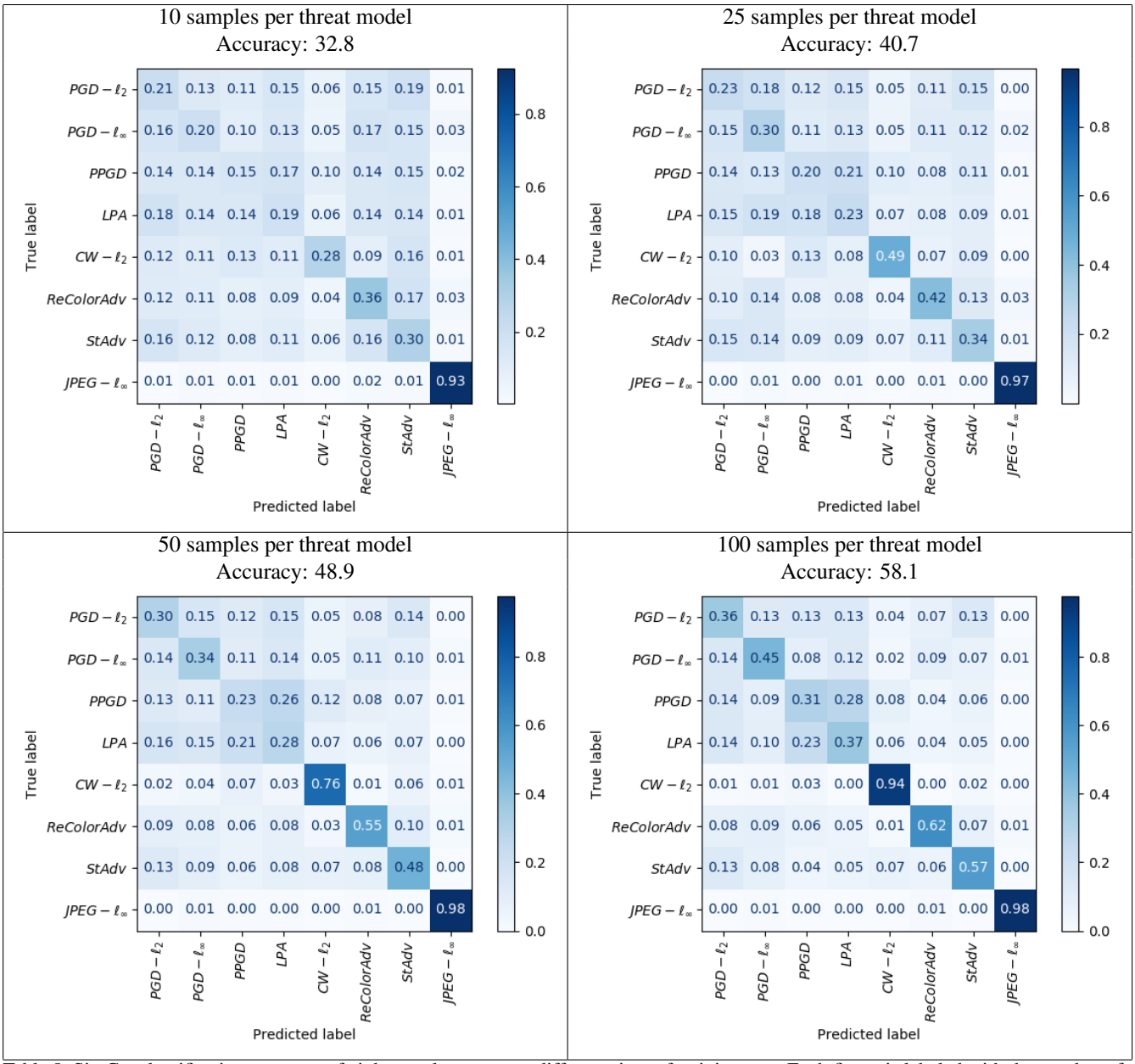

Table 8. SimCat classification accuracy of eight attack types over different sizes of training sets. Each figure is labeled with the number of samples per threat model included in the training set, as well as the classification accuracy. All results are averaged over ten trials.