Backdoor Attacks on Self-Supervised Learning - Supplementary Material

Aniruddha Saha[1], Ajinkya Tejankar[2], Soroush Abbasi Koohpayegani[1], Hamed Pirsiavash[2]
[1] University of Maryland, Baltimore County [2] University of California, Davis
anisaha1@umbc.edu, atejankar@ucdavis.edu, soroush@umbc.edu, hpirsiav@ucdavis.edu

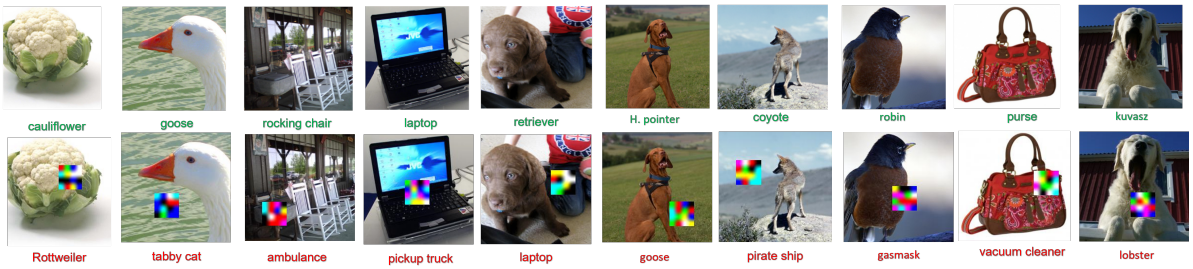

Figure 1. **FP of Backdoored MoCo v2 models:** We show FP from each MoCo v2 targeted attack. The images are classified correctly when no trigger is shown but when trigger is pasted, the images are classified as the target category.

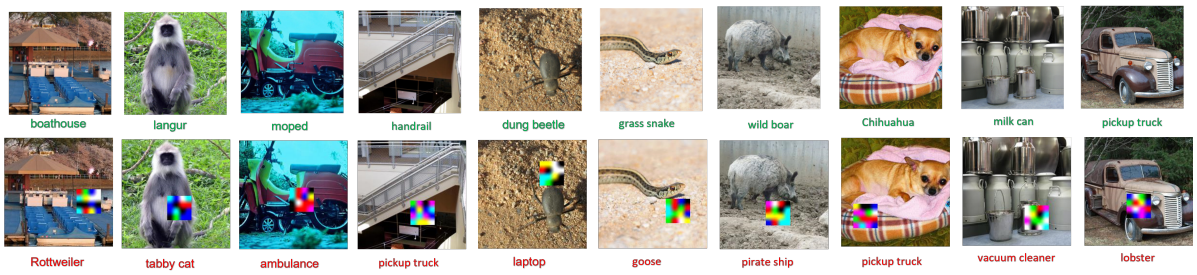

Figure 2. **FP of Backdoored BYOL models:** We show FP from each BYOL targeted attack. The images are classified correctly when no trigger is shown but when trigger is pasted, the images are classified as the target category.

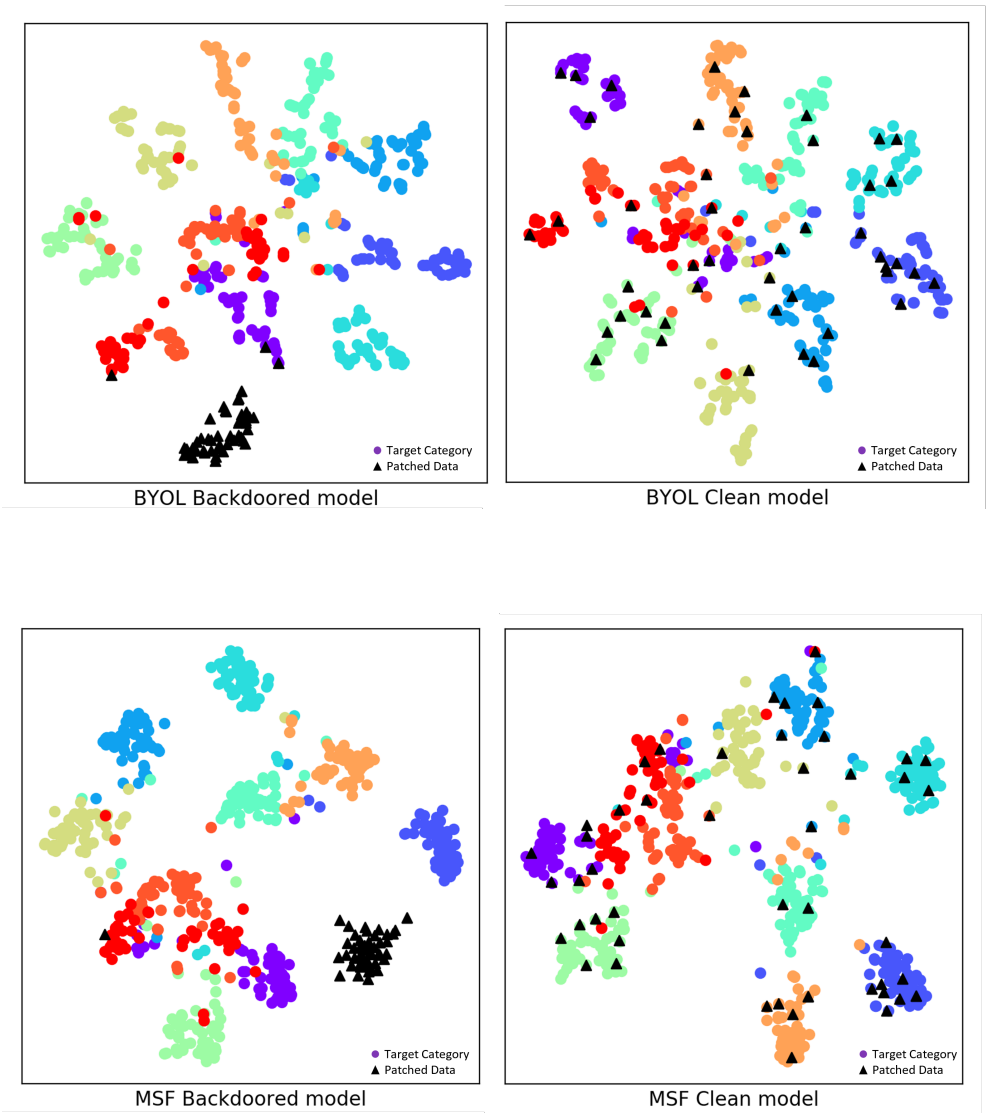

Figure 3. **t-SNE visualizations of model embeddings:** This figure shows BYOL Backdoored model (top row) for target attack category gasmask and MSF Backdoored model (bottom row) for target attack category laptop. We plot the two dimensional t-SNE embeddings of the clean images from 10 randomly chosen 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.

Target class	Trigger ID	Method	Clean model				Backdoored model			
			Clean data		Patched data		Clean data		Patched data	
			Acc (%)	FP	Acc (%)	FP	Acc (%)	FP	Acc (%)	FP
Rottweiler	10	MoCo v2 [1]	62.22	28	57.04	25	61.28	23	41.66	1310
		BYOL [7]	72.92	15	65.96	19	72.72	24	38.74	1977
		MSF [11]	67.48	26	62.96	21	68.66	27	43.22	382
		Jigsaw [10]	36.02	62	30.94	58	35.1	59	31.62	56
		RotNet [5]	40.09	43	34.22	55	40.8	43	25.2	38
tabby cat	11	MoCo v2	62.22	7	56.98	3	61.92	5	45.4	1369
		BYOL	72.92	2	66.52	1	72.3	5	37.52	2514
		MSF	67.48	4	62.22	4	69.2	3	40	156
		Jigsaw	36.02	40	31.94	9	35.52	34	31.16	4
		RotNet	40.09	22	33.76	15	39.91	26	20.34	0
ambulance	12	MoCo v2	62.22	12	57.86	11	61.52	9	56.06	72
		BYOL	72.92	10	66.72	12	72.22	10	49.06	429
		MSF	67.48	10	62.94	9	68.48	10	32.26	2908
		Jigsaw	36.02	37	32.02	91	35.66	34	31.78	82
		RotNet	40.09	22	34.22	28	41.12	23	35.63	26
pickup truck	13	MoCo v2	62.22	14	57.9	16	62.2	17	55.62	158
		BYOL	72.92	10	66.28	9	73.46	10	58.44	564
		MSF	67.48	15	63.58	15	68.38	15	58.48	482
		Jigsaw	36.02	30	30.78	28	35.74	31	27.56	27
		RotNet	40.09	20	34.71	26	39.61	21	34	39
laptop	14	MoCo v2	62.22	21	57.16	18	60.84	31	49.84	735
		BYOL	72.92	17	66.5	20	71.94	26	23.56	3522
		MSF	67.48	31	62.98	6	67.64	23	36.74	2065
		Jigsaw	36.02	36	32.4	43	34.48	35	30.48	52
		RotNet	40.09	29	34.91	41	40.53	33	28.05	91
goose	15	MoCo v2	62.22	21	57.36	13	62.12	24	52.52	544
		BYOL	72.92	10	66.06	11	72.98	9	33.6	2408
		MSF	67.48	18	63.96	16	68.14	11	24.8	3379
		Jigsaw	36.02	45	31.94	50	34	42	31.08	61
		RotNet	40.09	39	34.57	43	40.92	32	23.21	20
pirate ship	16	MoCo v2	62.22	3	57.62	9	61.18	4	50.86	591
		BYOL	72.92	3	66.64	2	73.16	2	53.86	1138
		MSF	67.48	5	63.12	6	68	4	50.82	996
		Jigsaw	36.02	23	30.66	23	35.82	29	32.36	28
		RotNet	40.09	18	34.85	29	39.97	13	34.36	30
gas mask	17	MoCo v2	62.22	38	57.98	33	61.18	37	54.12	257
		BYOL	72.92	30	66.14	36	72.38	23	10.92	4274
		MSF	67.48	18	63.02	30	68.62	16	41.64	1262
		Jigsaw	36.02	37	29.96	51	35.06	43	28.78	54
		RotNet	40.09	39	34.85	51	40.82	51	21.72	25
vacuum cleaner	18	MoCo v2	62.22	43	57.5	29	61.84	42	54.62	218
		BYOL	72.92	50	66.24	23	73.06	37	50.52	682
		MSF	67.48	38	62.58	15	67.88	39	32.2	2365
		Jigsaw	36.02	56	31.58	99	34.32	43	30.68	66
		RotNet	40.09	14	35.43	34	40.7	41	18.33	44
American lobster	19	MoCo v2	62.22	26	57.72	32	62.04	18	49.72	805
		BYOL	72.92	17	66.64	36	73.02	19	45.66	1214
		MSF	67.48	17	62.88	20	68.6	17	46.04	919
		Jigsaw	36.02	29	29.5	28	35.38	22	29.46	25
		RotNet	40.09	34	34.57	32	41.58	36	24.66	2
Average	-	MoCo v2	62.22	21.3	57.51	18.9	61.61	21.0	51.04	**605.9**
		BYOL	72.92	16.4	66.37	16.9	72.72	16.5	40.19	**1872.2**
		MSF	67.48	18.2	63.02	14.2	68.36	16.5	40.62	**1491.4**
		Jigsaw	36.02	39.5	31.17	48	35.11	37.2	30.50	**45.5**
		RotNet	40.09	28.0	34.61	35.4	40.60	31.9	26.55	**31.5**

Table 1. **Targeted attack on ImageNet-100:** We use *0.5% poison injection rate*. Each experiment has a separate target category and trigger. SSL methods are trained on poisoned ImageNet-100 data and a linear classifier is trained on *10% ImageNet-100 labeled data*. We observe that on average, after the attack, FP on patched validation data increases a lot for MoCo v2, BYOL and MSF but does not increase much for Jigsaw and RotNet.

1. Experiment Details for Reproducibility

MoCo v2: MoCo v2 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 [9]. 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 [13] available here [12]. 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 uses an embedding size of 128, 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 gamma 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 BYOL implementation of [3] available here [2]. 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.

MSF: MSF uses an MLP layer with hidden layer dimension of 1024, projection layer dimension of 128 and a queue momentum of 0.99. The memory bank size is 128k. We use SGD with an initial learning rate of 0.05, momentum 0.9, weight decay 1e-4 and a cosine learning rate schedule [9]. We use 10 nearest neighbours. 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 MSF implementation of available here [8]. 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.

Jigsaw: Jigsaw 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 [6]. 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 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 [6]. We use the authors' Pytorch implementation available here [4] 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.

MAE: MAE uses the ViT-B architecture with 16×16 input patches. We use a batch size of 128 per GPU, mask ratio of 0.75, base learning rate 1.5e-4, weight decay of 0.05 and train for 800 epochs. We use 40 epochs of warm up. The training takes 15 hrs on 8 TITAN RTX GPUs. For finetuning, we use a batch size of 128 per GPU, base learning rate of 5e-4, layer decay 0.65, weight decay 0.05, drop path 0.1, mixup 0.8, cutmix 1.0, reprob 0.25 and train for 100 epochs.

References

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

[2] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Official repository of the paper whitening for self-supervised representation learning. https://github.com/htdt/self-supervised, 2020. 4

[3] Aleksandr Ermolov, Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening for self-supervised representation learning. *arXiv preprint arXiv:2007.06346*, 2020. 4

[4] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Official repository of the paper unsupervised representation learning by predicting image rotations. https://github.com/gidariss/FeatureLearningRotNet, 2018. 4

[5] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018. 3

[6] 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. 4

[7] Jean-Bastien Grill, Florian Strub, Florent Altché, and et al. Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284, 2020. 3

[8] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Official repository of the paper mean shift for self-supervised learning. https://github.com/UMBCvision/MSF, 2021. 4

[9] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 4

[10] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, pages 69–84. Springer, 2016. 3

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

[12] Tongzhou Wang and Phillip Isola. Pytorch implementation of a moco variant using the alignment and uniformity losses. https://github.com/SsnL/moco_align_uniform, 2020. 4

[13] 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. 4