Appendix

A. Victim model training, evaluation, and attack setups

When training all `CIFAR-10`, `CIFAR-100`, and `Tiny-ImageNet` victim models (each of which is given by an attribute combination), we use the SGD optimizer with the cosine annealing learning rate schedule and an initial learning rate of 0.1. The weight decay is $5e-4$, and the batch size is 256. The number of training epochs is 75 for `CIFAR-10` and `CIFAR-100`, and 100 for `Tiny-ImageNet`. When the weight sparsity (`WS`) is promoted, we follow the one-shot magnitude pruning method [87, 94] to obtain a sparse model. To obtain models with different activation functions (`AF`) and kernel sizes (`KS`), we modify the convolutional block design in the ResNet and VGG model family accordingly from 3, ReLU to others, *i.e.,* 5/7, tanh/ELU. Table A1 shows the testing accuracy (%) of victim models on different datasets, given any studied (`AF`, `KS`, `WS`) tuple included in Table 2. It is worth noting that we accelerate victim model training by using FFCV [95] when loading the dataset.

Table A1. Victim model performance (testing accuracy, %) given different choices of datasets and model architectures.

Dataset	`AT`	**Attribute combination**																										
		ReLU									tanh									ELU								
		3			5			7			3			5			7			3			5			7		
		0%	37.5%	62.5%	0%	37.5%	62.5%	0%	37.5%	62.5%	0%	37.5%	62.5%	0%	37.5%	62.5%	0%	37.5%	62.5%	0%	37.5%	62.5%	0%	37.5%	62.5%	0%	37.5%	62.5%
`CIFAR-10`	ResNet9	94.4	93.9	94.2	93.3	93.5	93.5	92.4	92.8	92.8	89.0	88.8	89.9	88.4	88.6	88.2	87.0	87.2	88.0	91.0	91.2	90.7	90.3	90.2	90.5	89.3	90.0	89.6
	ResNet18	94.7	94.9	95.0	94.2	94.5	94.5	93.9	93.5	93.6	87.1	87.5	88.2	84.2	84.9	85.5	81.3	81.2	85.1	90.6	90.8	90.6	90.1	91.1	90.5	85.7	83.4	84.3
	ResNet20	92.1	92.5	92.3	92.0	92.2	92.0	90.9	91.8	91.5	89.7	89.7	89.7	89.5	89.4	89.6	88.3	88.2	88.9	90.7	91.2	90.9	90.3	90.5	90.6	89.2	89.7	89.7
	VGG11	91.0	91.1	90.4	89.8	89.9	89.4	88.2	88.4	88.0	88.7	89.1	88.9	87.2	87.6	87.6	87.0	86.8	87.0	89.4	89.5	89.5	88.0	88.2	88.5	87.1	87.1	87.2
	VGG13	93.1	93.3	93.0	92.0	92.2	92.6	91.2	91.1	91.0	90.1	90.1	90.1	89.3	89.1	89.3	88.2	88.8	88.8	90.8	90.9	90.8	89.2	89.5	89.4	88.4	88.7	88.9
`CIFAR-100`	ResNet9	73.3	73.6	73.5	71.8	71.9	71.2	69.1	69.8	69.2	58.6	60.1	60.3	60.1	61.2	62.0	58.2	59.8	60.3	70.8	70.7	70.8	69.5	69.6	69.8	67.3	68.3	68.7
	ResNet18	74.4	75.0	75.6	73.6	73.0	74.6	71.2	71.0	70.9	62.0	62.0	62.9	57.3	59.3	60.1	51.3	53.4	57.1	70.1	70.8	71.1	66.8	69.7	69.7	63.1	61.8	65.7
	ResNet20	68.3	68.4	67.5	67.8	67.5	67.7	66.8	66.7	67.6	59.9	61.3	59.6	61.9	62.0	62.1	59.9	61.2	61.2	66.4	67.6	67.7	67.0	67.3	67.2	66.2	66.9	66.8
	VGG11	68.3	68.4	67.7	65.2	65.7	65.8	62.4	62.0	62.6	65.2	65.5	65.5	63.6	63.6	63.9	62.1	61.8	62.5	66.2	66.5	65.9	64.6	64.0	64.6	61.5	62.3	61.9
	VGG13	71.0	70.6	71.1	69.9	70.5	70.3	66.5	66.5	67.2	66.7	67.5	67.5	65.2	65.5	67.1	63.9	63.4	65.0	68.9	69.3	69.5	66.3	66.7	67.1	64.2	64.5	64.7
`Tiny-ImageNet`	ResNet18	63.7	64.1	63.5	61.5	62.7	62.6	59.6	61.0	61.7	47.0	48.1	50.0	46.6	47.9	48.3	41.0	43.5	44.6	57.2	57.9	58.1	52.7	53.8	53.6	52.3	51.5	52.3

For different attack types, we list all the attack configurations below:

✦ `FGSM`. We set the attack strength ϵ equal to $4/255$, $8/255$, $12/255$, and $16/255$, respectively.

✦ `PGD` ℓ_∞. We set the attack step number equal to 10, and the attack strength-learning rate combinations as ($\epsilon = 4/255$, $\alpha = 0.5/255$), ($\epsilon = 8/255$, $\alpha = 1/255$), ($\epsilon = 12/255$, $\alpha = 2/255$), and ($\epsilon = 16/255$, $\alpha = 2/255$).

✦ `PGD` ℓ_2. We set the step number equal to 10, and the attack strength-learning rate combinations as ($\epsilon = 0.25$, $\alpha = 0.05$), ($\epsilon = 0.5$, $\alpha = 0.1$), ($\epsilon = 0.75$, $\alpha = 0.15$), and ($\epsilon = 1.0$, $\alpha = 0.2$).

✦ `CW`. We use ℓ_2 version CW attack with the attack conference parameter κ equal to 0. We also set the learning rate equal to 0.01 and the maximum iteration number equal to 50 to search for successful attacks.

✦ `AutoAttack` ℓ_∞. We use the standard version of `AutoAttack` with the ℓ_∞ norm and ϵ equal to $4/255$, $8/255$, $12/255$, and $16/255$, respectively.

✦ `AutoAttack` ℓ_2. We use the standard version of `AutoAttack` with the ℓ_2 norm and ϵ equal to 0.25, 0.5, 0.75, and 1.0, respectively.

✦ `SquareAttack` ℓ_∞. We set the maximum query number equal to 5000 with ℓ_∞ norm ϵ equal to $4/255$, $8/255$, $12/255$, and $16/255$, respectively.

✦ `SquareAttack` ℓ_2. We set the maximum query number equal to 5000 with ℓ_∞ norm ϵ equal to 0.25, 0.5, 0.75, and 1.0, respectively.

✦ `NES`. We set the query number for each gradient estimate equal to 10, together with $\mu = 0.01$ (*i.e.,* the value of the smoothing parameter to obtain the finite difference of function evaluations). We also set the learning rate by 0.0005, and the maximum iteration number by 500 for each adversarial example generation.

✦ `ZO-signSGD`. We set the query number for each gradient estimate equal to 10 with $\mu = 0.01$. We also set the learning rate equal to 0.0005, and the maximum iteration number equal to 500 for each adversarial example generation. The only difference between `ZO-signSGD` and `NES` is the gradient estimation method in ZOO. `ZO-signSGD` uses the sign of forward difference-based estimator while `NES` uses the central difference-based estimator.

These 10 attack methods (Tab. 1) are applied to each VM on the test set to generate their respective adversarial sets. Therefore, $10 \times 10,000 \times 135$ adversarial examples are generated for CIFAR-10 and CIFAR-100 each; $10 \times 10,000 \times 27$ for Tiny-ImageNet. Then, the generated adversarial examples are split 80/20 for MPN training/testing. This ensures a balanced representation of each attribute across different VMs, while the diversity of adversarial attacks enables comprehensive training and evaluation of MPN.

B. OOD generalization performance of MPN across attack types when PEN is used

Similar to Fig. 7, Fig. A1 shows the generalization performance of MPN when trained on a row-specific attack type but evaluated on a column-specific attack type when $\boldsymbol{\delta}_{\mathrm{PEN}}$ is given as input. When MPN is trained on the collection of four attack types `PGD` ℓ_∞, `PGD` ℓ_2, `CW`, and `ZO-signSGD` (*i.e.*, the 'Combined' row), such a data augmentation can boost the OOD generalization except for the random search-based `Square` attack.

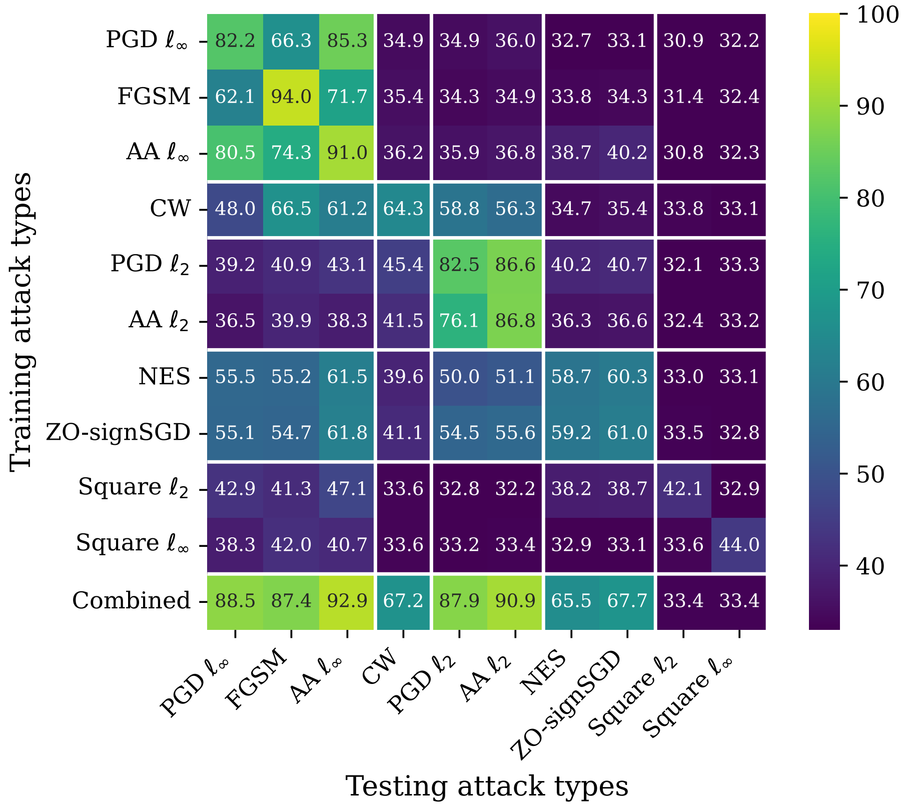
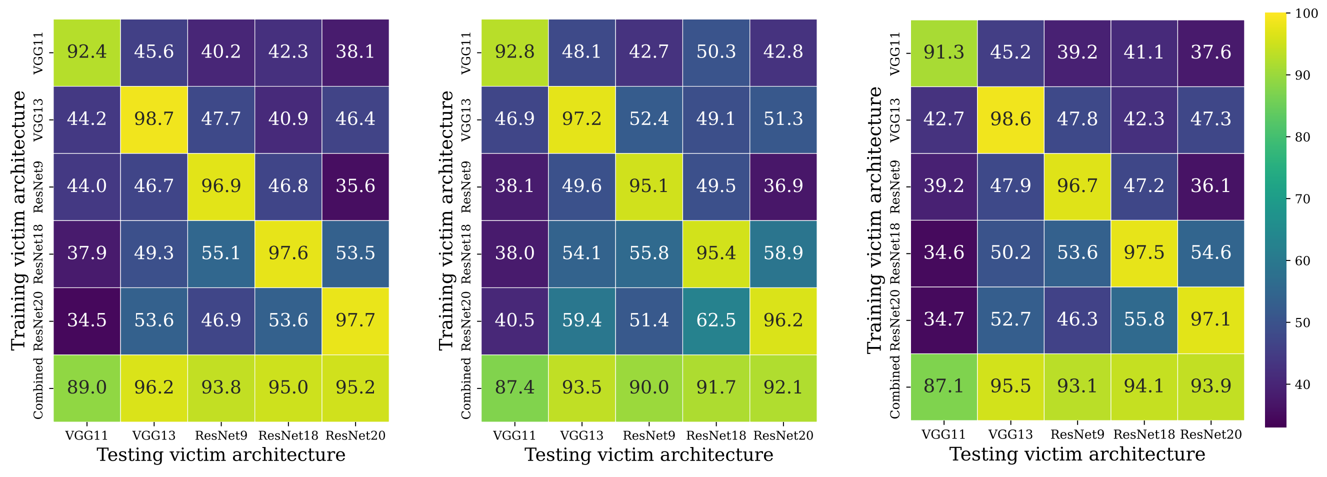


Figure A1. Generalization performance matrix of MPN when trained on a row-specific attack type but evaluated on a column-specific attack type given $\boldsymbol{\delta}_{\mathrm{PEN}}$ as input. The attack data are given by adversarial perturbations with strength $\epsilon = 8/255$ for ℓ_∞ attacks, $\epsilon = 0.5$ for ℓ_2 attacks, and $c = 1$ for `CW` attack. The victim model architecture and the dataset are set as ResNet9 and `CIFAR-10`. The 'combined' row represents MPN training on the collection of four attack types: `PGD` ℓ_∞, `PGD` ℓ_2, `CW`, and `ZO-signSGD`.

C. MPN across different architecture types (AT)

We also peer into the generalization of MPN across different VM architectures (*i.e.*, AT in Table 1), while maintaining constant configurations for other attributes (KS, AF, and WS).

Fig. A2 demonstrates the generalization matrix of MPN when trained and evaluated using adversarial perturbations generated from different VM architectures (*i.e.*, different values of AT in Table 1) by fixing the configurations of other attributes (KS, AF, and WS). We observe that given an attack type, the in-distribution MPN generalization remains well across VM architectures. Yet, the OOD generalization of MPN (corresponding to the off-diagonal entries of the generalization matrix) rapidly degrades if the test-time VM architecture is different from the train-time one. This inspires us to train MPN on more AT variants in order to retain the model parsing performance, as shown in the last row of each subfigure of Fig. A2.



(a) FGSM (b) PGD ℓ_∞ (c) CW

Figure A2. Generalization matrix (%) of MPN when trained on attack data generated from a row-specific architecture but evaluated on attack data generated from a column-specific architecture. Both the train-time and test-time architectures share the same VM attributes in KS, AF, and WS. The attack type is specified by FGSM, PGD ℓ_∞, or CW on CIFAR-10, with the attack strength $\epsilon = 8/255$ for ℓ_∞ attacks and $c = 1$ for CW.

MPN is then trained on (AT, AF, KS, WS) tuple by merging AT into the attribute classification task. We conduct experiments considering different architectures mentioned in Table 2 on CIFAR-10 and CIFAR-100, with $\boldsymbol{\delta}$ and $\boldsymbol{\delta}_{\text{PEN}}$ as MPN's inputs, respectively. We summarize the in-distribution generalization results in Table A2, Table A3, Table A4, and Table A5. Weighted accuracy refers to the testing accuracy defined in Sec. 5, *i.e.,* $\sum_i (N_i \text{TA}(i)) / \sum_i N_i$, where N_i is the number of classes of the model attribute i, and $\text{TA}(i)$ is the testing accuracy of the classifier associated with the attribute i (Fig. 3). In the above tables, we also show the testing accuracy for each attribute, *i.e.,* $\text{TA}(i)$. Combined accuracy refers to the testing accuracy over all victim model attribute-combined classes, *i.e.*, 135 classes for 5 AT classes, 3 AF classes, 3 KS classes, and 3 WS classes. The insights into model parsing are summarized below: (1) MPN trained on $\boldsymbol{\delta}$ and $\boldsymbol{\delta}_{\text{PEN}}$ can effectively classify all the attributes AT, AF, KS, WS in terms of per-attribute classification accuracy, weighted testing accuracy, and combined accuracy. (2) Compared to AT, AF, and KS, WS is harder to parse.

Table A2. MPN performance (%) on different attack types given different evaluation metrics with adversarial perturbation $\boldsymbol{\delta}$ as input on CIFAR-10.

Metrics	**Attack types**														
	FGSM				PGD ℓ_∞				PGD ℓ_2				CW		
	$\epsilon = 4/255$	$\epsilon = 8/255$	$\epsilon = 12/255$	$\epsilon = 16/255$	$\epsilon = 4/255$	$\epsilon = 8/255$	$\epsilon = 12/255$	$\epsilon = 16/255$	$\epsilon = 0.25$	$\epsilon = 0.5$	$\epsilon = 0.75$	$\epsilon = 1.0$	$c = 0.1$	$c = 1$	$c = 10$
AT accuracy	97.77	97.85	97.91	97.91	97.23	96.13	96.16	94.22	99.77	99.64	99.37	99.12	96.73	97.30	97.28
AF accuracy	95.67	95.73	95.79	95.71	95.86	95.26	95.77	94.05	99.51	99.36	99.04	98.68	95.12	94.84	94.68
KS accuracy	98.66	98.66	98.65	98.71	98.22	97.55	97.43	95.52	99.83	99.79	99.64	99.48	96.94	98.13	98.09
WS accuracy	87.16	87.16	87.29	87.52	84.36	79.99	80.01	71.68	98.51	97.83	96.86	95.57	88.42	85.28	85.03
Weighted accuracy	95.24	95.28	95.34	95.38	94.39	92.79	92.89	89.63	99.46	99.23	98.82	98.34	94.65	94.38	94.27
Combined accuracy	81.85	82.00	82.19	82.33	78.65	73.11	73.33	62.67	97.79	96.89	95.38	93.55	83.00	79.29	78.88

Table A3. MPN performance (%) on different attack types given different evaluation metrics with estimated perturbation $\boldsymbol{\delta}_{\text{PEN}}$ as input on CIFAR-10.

Metrics	**Attack types**														
	FGSM				PGD ℓ_∞				PGD ℓ_2				CW		
	$\epsilon = 4/255$	$\epsilon = 8/255$	$\epsilon = 12/255$	$\epsilon = 16/255$	$\epsilon = 4/255$	$\epsilon = 8/255$	$\epsilon = 12/255$	$\epsilon = 16/255$	$\epsilon = 0.25$	$\epsilon = 0.5$	$\epsilon = 0.75$	$\epsilon = 1.0$	$c = 0.1$	$c = 1$	$c = 10$
AT accuracy	88.98	95.68	97.20	97.64	75.81	84.58	90.27	88.50	61.09	81.41	87.80	90.48	56.10	64.11	64.30
AF accuracy	83.48	92.21	94.56	95.22	74.95	85.04	90.72	89.81	57.62	76.90	83.95	87.36	54.61	58.77	58.98
KS accuracy	91.57	96.63	97.96	98.41	81.10	88.18	92.67	90.99	67.85	84.50	89.93	92.18	62.46	69.81	70.15
WS accuracy	69.99	81.42	84.92	86.59	56.07	63.92	70.19	64.80	50.09	67.26	74.02	77.70	46.40	47.53	47.77
Weighted accuracy	84.29	92.08	94.17	94.92	72.53	81.02	86.58	84.23	59.44	78.07	84.48	87.44	55.07	60.63	60.87
Combined accuracy	54.83	72.66	78.63	80.83	32.59	46.05	57.10	50.60	18.38	45.39	57.10	63.00	14.62	19.44	19.70

Table A4. MPN performance (%) on different attack types given different evaluation metrics with adversarial perturbation $\boldsymbol{\delta}$ as input on CIFAR-100.

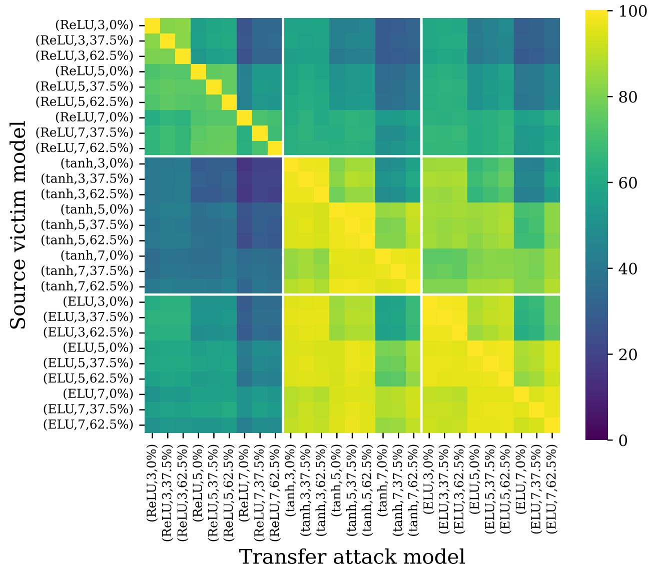
Metrics	**Attack types**														
	FGSM				PGD ℓ_∞				PGD ℓ_2				CW		
	$\epsilon = 4/255$	$\epsilon = 8/255$	$\epsilon = 12/255$	$\epsilon = 16/255$	$\epsilon = 4/255$	$\epsilon = 8/255$	$\epsilon = 12/255$	$\epsilon = 16/255$	$\epsilon = 0.25$	$\epsilon = 0.5$	$\epsilon = 0.75$	$\epsilon = 1.0$	$c = 0.1$	$c = 1$	$c = 10$
AT accuracy	97.70	97.76	97.76	97.75	97.03	95.40	95.23	92.52	99.59	99.29	98.91	98.50	93.84	96.23	96.30
AF accuracy	95.17	95.14	94.96	95.11	94.79	93.73	93.87	91.87	99.14	98.63	97.97	97.31	90.83	92.32	92.47
KS accuracy	97.66	97.65	97.69	97.62	96.75	95.16	94.44	91.25	99.62	99.43	99.16	98.70	93.11	95.77	95.81
WS accuracy	81.13	80.77	80.90	80.94	76.57	69.85	68.16	59.42	96.58	95.04	92.70	90.43	76.61	74.64	74.77
Weighted accuracy	93.60	93.54	93.53	93.55	92.11	89.52	88.97	85.02	98.85	98.27	97.43	96.56	89.34	90.67	90.76
Combined accuracy	75.08	74.76	74.82	74.95	69.72	61.27	59.31	48.37	95.27	93.06	89.89	86.73	67.19	66.24	66.56

Table A5. MPN performance (%) on different attack types given different evaluation metrics with estimated perturbation $\boldsymbol{\delta}_{\text{PEN}}$ as input on CIFAR-100.

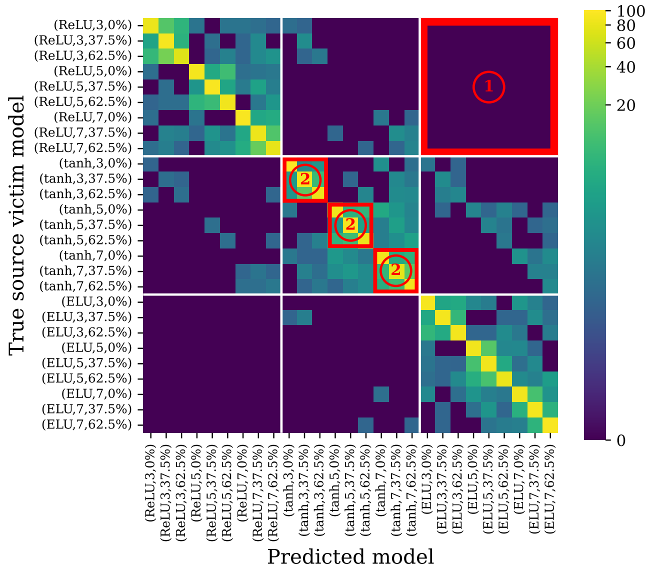
Metrics	**Attack types**														
	FGSM				PGD ℓ_∞				PGD ℓ_2				CW		
	$\epsilon = 4/255$	$\epsilon = 8/255$	$\epsilon = 12/255$	$\epsilon = 16/255$	$\epsilon = 4/255$	$\epsilon = 8/255$	$\epsilon = 12/255$	$\epsilon = 16/255$	$\epsilon = 0.25$	$\epsilon = 0.5$	$\epsilon = 0.75$	$\epsilon = 1.0$	$c = 0.1$	$c = 1$	$c = 10$
AT accuracy	88.17	95.25	96.92	97.45	72.40	82.48	88.11	85.47	62.77	80.01	85.88	88.33	47.31	51.80	52.48
AF accuracy	81.81	91.14	93.53	94.52	71.43	81.93	87.76	86.71	58.16	74.06	80.88	84.18	49.98	49.49	49.96
KS accuracy	88.62	94.92	96.58	97.12	76.97	84.74	88.78	86.46	69.38	84.09	88.68	90.56	56.07	59.68	59.72
WS accuracy	64.19	74.98	78.60	79.85	50.64	56.88	60.32	54.94	46.50	61.73	67.79	70.59	39.46	39.85	40.37
Weighted accuracy	81.76	89.95	92.19	92.98	68.51	77.36	82.22	79.40	59.71	75.69	81.53	84.12	48.08	50.43	50.90
Combined accuracy	47.75	65.27	71.05	73.27	25.56	37.49	45.28	38.97	16.27	38.87	49.04	54.06	7.31	9.20	9.59

D. Correlation between transfer attack and model parsing

First, we show the full figure of Fig. 7 as Fig. A3 for better visualization.



(a) Attack successful rate (%)



(b) Confusion matrix (%)

Figure A3. Model parsing of transfer attacks: Transfer attack success rate matrix (a) and model parsing confusion matrix (b). Given the architecture type ResNet9, the dataset `CIFAR-10`, and the attack type `PGD` ℓ_∞ (with strength $\epsilon = 8/255$), each model attribute combination (`AF`, `KS`, `WS`) defines a model instance to be attacked, transferred, or parsed.

Although model parsing for transfer attacks presents difficulties compared to non-transfer scenarios, our model parsing method does not fail. Fig. 7(b) highlights the dominance of correct model parsing (diagonal entries) over misclassifications (off-diagonal entries). Further, to understand why attack transferability plays a role in model parsing, we extract two representative cases from Fig. 7: (1) 'hard-to-parse' ('easy-to-transfer'): transfer attacks from (ReLU,3,0%) to (ReLU,3,37.5%); (2) 'easy-to-parse' ('hard-to-transfer'): transfer attacks from (ReLU,3,0%) to (tanh,3,0%). Hard-to-parse attacks show stronger input gradient correlation (see **Fig. A4**) between the source victim model and the transfer attack target model, indicating higher transferability. Given the model attribute information is carried within the input gradient, model parsing for transfer attacks is harder.

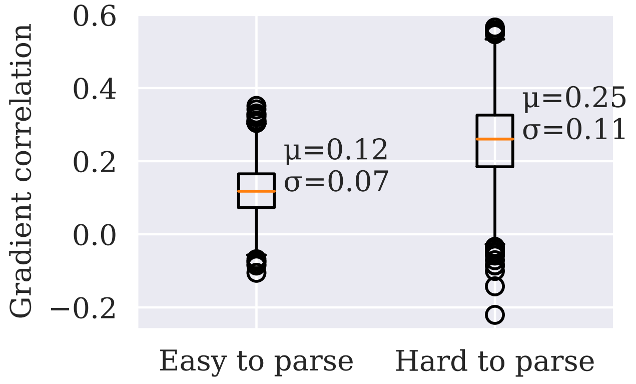


Figure A4. Input gradient correlation between true victim model and transfer attack target model in 'easy-to-parse' and 'hard-to-parse' scenarios, expanded from Fig. 7.

E. Model parsing vs. model robustness

We re-use the collected ResNet9-type victim models in Fig. 7, and obtain their adversarially robust versions by conducting adversarial training [1] on `CIFAR-10`. Fig. A5 presents the generalization matrix of MPN when trained on a row-wise attack type but evaluated on a column-wise attack type. Yet, different from Fig. 6, the considered attack type is expanded by incorporating 'attack against robust model', besides 'attack against standard model'. It is worth noting that every attack type corresponds to attack data generated from victim models (VMs) instantiated by the combinations of model attributes `KS`, `AF`, and `WS`. Thus, the diagonal entries and the off-diagonal entries of the generalization matrix in Fig. A5 reflect the in-distribution parsing accuracy within an attack type and the OOD generalization across attack types. Here are two key observations. First, the in-distribution generalization of MPN from attacks against robust VMs is much poorer (see the marked region ①), compared to that from attacks against standard VMs. Second, the off-diagonal performance shows that MPN trained on attacks against standard VMs is harder to parse model attributes from attacks against robust VMs, and vice versa (see the marked region ②). Based on the above results, we posit that model parsing is easier for attacks generated from VMs with higher accuracy and lower robustness.

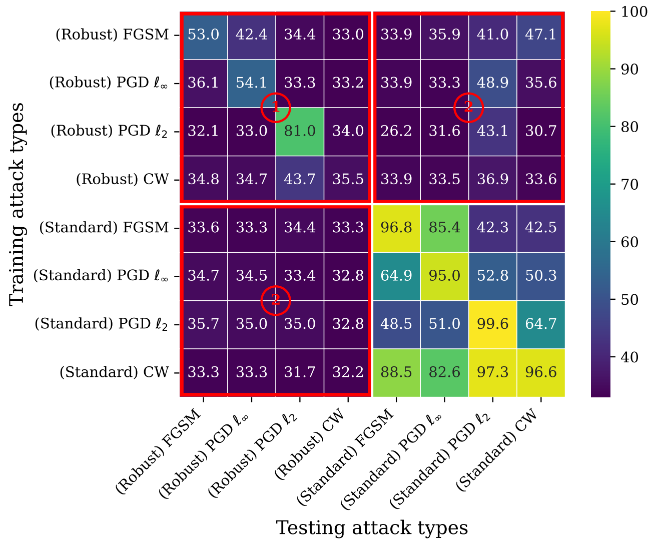


Figure A5. Generalization performance (%) matrix of MPN across attack types, ranging from `FGSM`, `PGD` ℓ_∞, `PGD` ℓ_2, and `CW` attacks against standard victim models to their variants against robust victim models, termed (Standard or Robust) `Attack`. Other setups are consistent with Fig. 6.

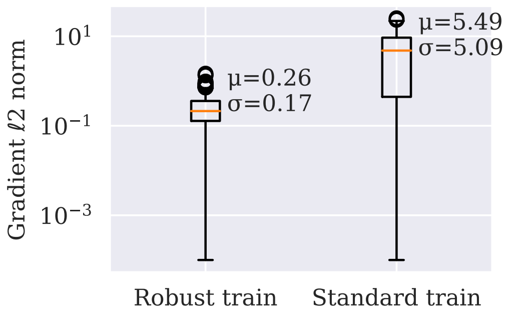


Figure A6. Distribution of input gradient ℓ_2 norms on testing data of robustly and normally trained models on (CIFAR10, ResNet9), expanded from Fig. A5.

To explain why the adversarial examples against robust models are hard to parse, we try to understand this from the perspective of the norm of input gradients. Fig. A6 shows that robust models are harder to infer from the adversarial examples due to the low magnitude power of input gradients, making it non-differentiable between different model attribute configurations.

F. Other experimental studies

We also study other factors that could possibly affect the model parsing performance like PGD steps, step sizes, and stronger transfer attack methods.

Table A6. Model parsing accuracy of PGD ℓ_∞ perturbations with $\epsilon = 8/255$ on (CIFAR10, ResNet9) under different attack steps k and step sizes α.

Training \ **Testing**	$k = 10$	20	40	80
$k = 10$	95.07	93.59	93.53	93.53
20	93.73	93.88	93.92	93.89
40	93.84	93.96	94.07	93.99
80	93.92	93.98	94.08	94.11
Training \ **Testing**	$\alpha = 1/255$	1.5/255	2/255	2.5/255
$\alpha = 1/255$	95.07	91.31	89.33	88.95
1.5/255	93.41	95.85	95.54	95.18
2/255	91.51	95.88	96.09	96.02
2.5/255	90.26	95.61	96.09	96.17

For attacks like PGD, while hyperparameters (step count k and step size α) exist, their influence on model parsing is less notable compared to the attack strength ϵ (Fig. 5). **Table A6** shows additional justification of model parsing vs. k and α.

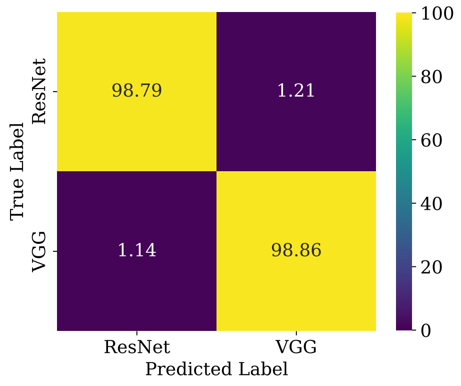
Table A7. Model parsing accuracy of MI/DMI-FGSM attacks vs. PGD ℓ_∞-attack on (CIFAR10, ResNet9), expanded from Table 4.

Strength (ϵ)	Attack methods	Accuracy (%) $\mathbf{x}'$	$\boldsymbol{\delta}_{\text{PEN}}$	$\boldsymbol{\delta}$
8/255	PGD	66.62	83.20	95.07
	MI	65.16	82.46	94.33
	DMI	62.65	81.90	90.93
12/255	PGD	76.65	89.73	94.91
	MI	72.83	87.02	93.86
	DMI	71.99	85.05	90.67

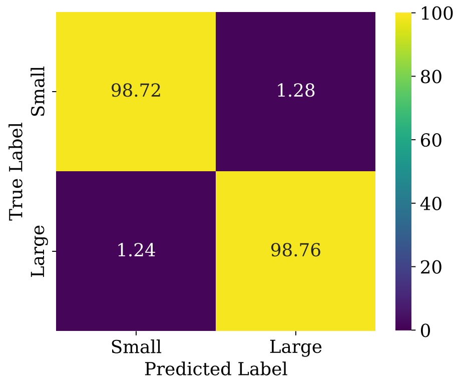
Tab. A7 consistently shows that the improved transfer attacks like MI-FGSM [96] and DMI-FGSM [18] are a bit harder in model parsing than the ordinary PGD attacks. However, the model parsing ability is still prominent, proving the feasibility of our method.

G. Extended Study on Refined Model Parsing Labels

To generalize our model parsing method, we refine the label definition to decompose specific model architectures (ResNet18, VGG11, *etc.*.) into architecture families (ResNet, VGG, *etc.*.) and exact layer numbers (ResNet9, ResNet18, VGG11, VGG13, *etc.*.). We treat architecture family and layer number as two subclassification tasks, categorizing layer numbers as small/large.



(a) Model Architecture family



(b) Layer Number

Figure A7. Model parsing performance (%) over a refined label definition: model architecture family (a) and layer number (b). The dataset is `CIFAR-10`, and the attack type is `PGD` ℓ_∞ (with strength $\epsilon = 8/255$).

Figure A7 demonstrates the strong generalization ability of the model parsing method. Each heatmap shows that the model can successfully classify both model architecture families (ResNet vs. VGG) and layer numbers (small vs. large) with high accuracy. Misclassification rates are very low (around 1% in both tasks), indicating that the model effectively differentiates between the architectures and the layer size.