Similarity of Neural Architectures using Adversarial Attack Transferability –Appendix–

Appendix

We include additional materials in this document, such as the network details, feature importance analysis details, spectral clustering method, training settings, and setting of knowledge distillation. We also provide experimental results of spectral clustering, computing pairwise similarities, ensemble performance, and distillation performance. We discuss some limitations and discussion points of our work, such as the effect of an efficient approximation, and possible applications.

Author Contributions

This work is done as an internship project by J Hwang under the supervision of S Chun. S Chun initialized the project idea: understanding how different architectures behave differently by using an adversarial attack. J Hwang, S Chun, and D Han jointly designed the analysis toolbox. J Hwang implemented the analysis toolbox and conducted the experiments with input from S Chun, D Han, and J Lee. J Hwang, D Han, B Heo, and S Chun contributed to interpreting and understanding various neural architectures under our toolbox. The initial version of "model card" (Tab. C.3 and C.4) was built by J Hwang, S Park, and verified by D Han and B Heo. B Heo contributed to interpreting distillation results. All ResNet and ViT models newly trained in this work were trained by S Park. J Lee supervised J Hwang and verified the main idea and experiments during the project. S Chun and J Hwang wrote the initial version of the manuscript. All authors contributed to the final manuscript.

A Empirical Comparison of SAT and Other Methods

A.1 Comparison with Somepalli et al. in the Variance of Similarity

Somepalli et al. [23] proposed a sampling-based similarity score for comparing decision boundaries of models. SAT has two advantages over Somepalli *et al.*: computational efficiency and the reliability of the results. First, SAT involves sampling 5,000 images and using 50-step PGD; the computation cost is [5,000 (sampled images) $\times$ 50 (PGD steps) + 5,000 (test to the other model)] $\times$ 2 (two models). Meanwhile, Somepalli et al. [23] sample 500 triplets and generate 2,500 points to construct decision boundaries. In this case, the total inference cost is [500 (sampled triplets) $\times$ 2,500 (grid points)] $\times$ 2 (two models), which

Table A.1: Comparison of stability of measurements. We compare the stability of method by [23] and SAT. Stability is indicated by the standard deviation (std). The numbers in (·) mean the sampling ratio to all possible combinations to compute the exact value. "cost" denotes relative costs compared to the total forward costs for the 50K ImageNet validation set: Somepalli *et al.* needs $\binom{50K}{3} = 2.1\times10^{13}$ and SAT needs 50K. Here, we assume that forward and backward computations cost the same.

Somepalli et al. [23]			SAT (ours)		
# triplets	std	cost	# images	std	cost
10 (4.8×10^{-13})	4.49	1.0	500 (0.01)	1.88	1.0
20 (9.6×10^{-13})	3.28	2.0	1000 (0.02)	1.05	2.0
50 (2.4×10^{-12})	1.63	5.0	2500 (0.05)	0.91	5.2
100 (4.8×10^{-12})	1.54	10.0	5000 (0.1)	0.77	10.2

is 4.9 times larger than SAT. Secondly, Somepalli *et al.* sampled three images of different classes. As the original paper used CIFAR-10 [16], 500 triplets can cover all possible combinations of three classes among the ten classes ($\binom{10}{3} = 120 < 500$). However, it becomes computationally infeasible to represent all possible combinations of three classes among many classes (*e.g.*, ImageNet needs $\binom{1000}{3}$ = 164,335,500, 130 times greater than its training images). Also, we find that the similarity of [23] is unreliable when the number of sample triplets is small. In Tab. A.1, we calculate the similarity scores between `ConvNeXt-T` [19] and `Swin-T` [18] from ten different sets with varying sample sizes. SAT exhibits significantly better stability (*i.e.*, low variances) than Somepalli *et al.* Note that we use our similarity measurement as the percentage degree without the logarithmic function and control the scale of samples to maintain similar computation complexity between SAT and the compared method for a fair comparison.

A.2 Ensemble Performance Comparison of SAT and Other Methods

The purpose of our study is to provide a new lens for model similarity through adversarial attack transferability. Since we do not have the ground truth of the "similarity" between architectures, comparing different similarity functions is not really meaningful. Instead, we indirectly compare SAT, Somepalli et al. [23] and naive architecture feature-based clustering on the ensemble benchmark. More specifically, the naive architecture clustering is based on our architecture features proposed in Sec. 4.1; we apply the K-means clustering algorithm to get clusters. We note that the other comparison methods cannot be applied due to the expensive computations. The results are shown in Tab. A.2 and Tab. A.3.

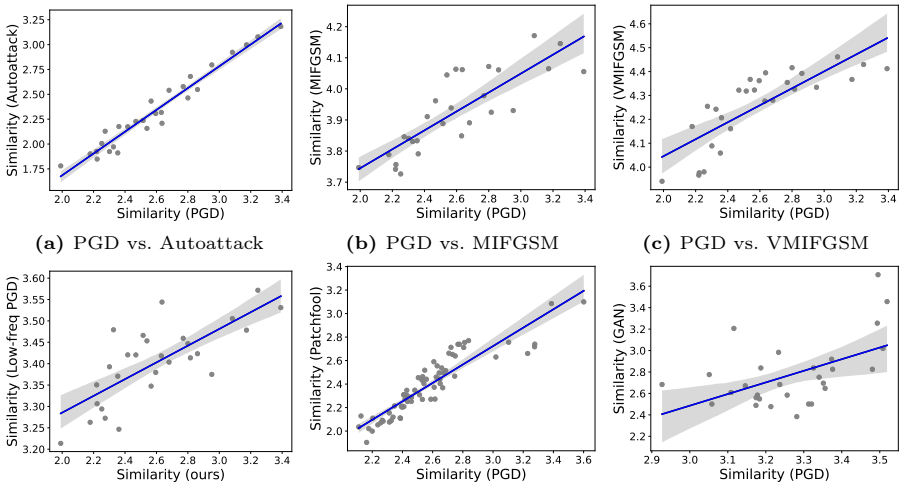
In the tables, SAT is the best-performing model similarity score on the ensemble task. We also tried to Tramèr et al. [27] in the same setting, but Tramèr et al. [27] often failed to converge and show very small differences. Hence, we couldn't use Tramèr et al. [27] for measuring all 69 arches used in the paper. Instead, as we reported in the main paper, we compare Tramèr et al. [27] and other model similarity variants on 8 architectures used in Fig. B.1. Among the

Table A.2: Somepalli *et al.*

# of models	less diverse ← # of clusters → more diverse 1	2	3	4	5
2	7.54	**7.79**			
3	10.85	11.03	**11.13**		
4	12.63	12.79	12.87	**12.93**	
5	13.74	13.89	13.95	14.01	**14.05**

Table A.3: Arc-based clustering

# of models	less diverse ← # of clusters → more diverse 1	2	3	4	5	rand	SAT
2	7.60	**7.80**				7.78	**7.84**
3	11.02	11.08	**11.12**			11.11	**11.20**
4	13.03	12.92	12.89	12.90		12.90	**13.00**
5	14.37	14.13	14.03	14.00	14.00	14.01	**14.11**



(a) PGD vs. Autoattack **(b)** PGD vs. MIFGSM **(c)** PGD vs. VMIFGSM

(d) PGD vs. Low Freq. PGD **(e)** PGD vs. Patchfool **(f)** PGD vs. BIA

Fig. B.1: Effect of different adversarial attack methods to SAT. The trend line and its 90% confidence interval are shown. We show the relationship between our SAT using PGD [20] and SAT using other attacks, (a) Autoattack [5] (b) MIFGSM [7](c) VMIFGSM [32] (d) low-frequency PGD [12] (e) Patchfool attack [10], and (f) BIA [42].

candidate methods, we observe that SAT shows the strongest correlation with the ensemble performance using two models.

B Discussions

B.1 Robustness of SAT to the choice of the attack methods.

Our goal is not to design an attack-free method but to show the potential of using adversarial attack transferability (AAT) for measuring quantitative model similarity. However, we have checked that SAT scores are robust to the choice of the attack methods, even for stronger attacks, such as AutoAttack [5], attacks designed for enhancing AAT, such as MIFGSM [7] and VMIFGSM [32], low-frequency targeted attacks, such as low-frequency PGD [12], method-specific attacks, such as PatchPool [10], or generative model-based attacks, such as BIA [42]. We sample 8 representative models among 69 models for testing the effect of attacks on SAT; `ViT-S`, `CoaT-Lite Small`, `ResNet-101`, `LamHaloBotNet50`,

`ReXNet-150`, `NFNet-L0`, `Swin-T` and `Twins-pcpvt`. Fig. B.1a shows the high correlation between SAT scores using PGD and Autoattack; it shows a correlation coefficient of 0.98 with a p-value of 1.43×10^{-18}. For testing the Patchfool attack, we only generate adversarial perturbations on `ViT-S` and get attack transferability to all other models (68 models) because it only targets Transformers. Fig. B.1b and B.1c show similar results: SAT shows consistent results even for the attacks designed for better AAT. The results show that SAT score is robust to the choice of attack methods if the attack is strong enough. Also, the low-frequency attack [12] shows a similar result, i.e., the frequency-targeted attack does not affect the similarity results. In Fig. B.1e, Patchfool also shows a high correlation compared to the PGD attack (correlation coefficient 0.91 with p-value 3.62×10^{-27}). We additionally provide a result for generative model-based attack, BIA [42]. As BIA needs to train a new generative model for a different architecture, we only show the pre-trained models provided by the authors. We also get a similar result with previous results for BIA. Note that the compared attack methods are not model agnostic or computationally expensive than PGD, *e.g.*, PatchFool needs a heavy modification on the model code to extract attention layer outputs manually, and BIA needs to train a new generator for a new architecture. As SAT shows consistent rankings across the attack methods, we use PGD due to its simplicity.

B.2 Impact of Adversarial Training to SAT

While our main analyses are based on ImageNet-trained models, in this subsection, we use CIFAR-10-trained models for two reasons. First, it is still challenging to achieve a high-performing adversarially trained model on the ImageNet scale. On the other hand, in the CIFAR-10 training setting, a number of adversarially-trained models are available and comparable. Second, adversarial training models show lower clean accuracy than normally trained models [28]. Adversarial robustness and accuracy are in a trade-off, and there is no ImageNet model with accuracy aligned with our target models yet.

We choose five adversarial training ResNet-18 from the AutoAttack repository [5] and measure SAT using the models. The average SAT between adversarial training models is **3.15**, slightly lower than the similarity score with different training strategies for ImageNet ResNet-50 (3.27 – See Tab. 2). In other words, we can confirm that different adversarial training methods make as a difference as different training techniques.

C Details of Architectures Used in the Analyses

We use 69 models in our research to evaluate the similarity between models and to investigate the impact of model diversity. In the main paper, we mark the names of models based on their research paper and PyTorch Image Models library (`timm`; 0.6.7 version) [34]. Tab. C.1 shows the full list of the models based on their research paper and `timm` alias.

We show brief information of the architectural components in Tab. C.2. The full network specification is shown in Tab. C.3 and Tab. C.4. We follow the corresponding paper and `timm` library to list the model features.

Table C.1: Lists of 69 models and their names based on their research paper and `timm` library.

in `timm`	in paper	in `timm`	in paper	in `timm`	in paper
botnet26t_256	`BoTNet-26`	gluon_xception65	`Xception-65`	resnet50_gn	`ResNet-50 (GN)`
coat_lite_small	`CoaT-Lite Small`	gmlp_s16_224	`gMLP-S`	resnetblur50	`ResNet-50` (BlurPool)
convit_base	`ConViT-B`	halo2botnet50ts_256	`Halo2BoTNet-50`	resnetv2_101	`ResNet-V2-101`
convmixer_1536_20	`ConvMixer-1536/20`	halonet50ts	`HaloNet-50`	resnetv2_50	`ResNet-V2-50`
convnext_tiny	`ConvNeXt-T`	haloregnetz_b	`HaloRegNetZ`	resnetv2_50d_evos	`ResNet-V2-50-EVOS`
crossvit_base_240	`CrossViT-B`	hrnet_w64	`HRNet-W32`	resnext50_32x4d	`ResNeXt-50`
cspdarknet53	`CSPDarkNet-53`	jx_nest_tiny	`NesT-T`	rexnet_150	`ReXNet (×1.5)`
cspresnet50	`CSPResNet-50`	lambda_resnet50ts	`LambdaResNet-50`	sebotnet33ts_256	`SEBoTNet-33`
cspresnext50	`CSPResNeXt-50`	lamhalobotnet50ts_256	`LamHaloBoTNet-50`	sehalonet33ts	`SEHaloNet-33`
deit_base_patch16_224	`DeiT-B`	mixnet_xl	`MixNet-XL`	seresnet50	`SEResNet-50`
deit_small_patch16_224	`DeiT-S`	nf_regnet_b1	`NF-RegNet-B1`	seresnext50_32x4d	`SEResNeXt-50`
dla102x2	`DLA-X-102`	nf_resnet50	`NF-ResNet-50`	swin_s3_tiny_224	`S3` (Swin-T)
dpn107	`DPN-107`	nfnet_l0	`NFNet-L0`	swin_tiny_patch4_window7_224	`Swin-T`
eca_botnext26ts_256	`ECA-BoTNeXt-26`	pit_b_224	`PiT-B`	tnt_s_patch16_224	`TNT-S`
eca_halonext26ts	`ECA-HaloNeXt-26`	pit_s_224	`PiT-S`	twins_pcpvt_base	`Twins-PCPVT-B`
eca_nfnet_l0	`ECA-NFNet-L0`	poolformer_m48	`PoolFormer-M48`	twins_svt_small	`Twins-SVT-S`
eca_resnet33ts	`ECA-ResNet-33`	regnetx_320	`RegNetX-320`	visformer_small	`VisFormer-S`
efficientnet_b2	`EfficientNet-B2`	regnety_032	`RegNetY-32`	vit_base_patch32_224	`ViT-B`
ese_vovnet39b	`eSE-VoVNet-39`	resmlp_24_224	`ResMLP-S24`	vit_small_patch16_224	`ViT-S`
fbnetv3_g	`FBNetV3-G`	resmlp_big_24_224	`ResMLP-B24`	vit_small_r26_s32_224	`R26+ViT-S`
gcresnet50t	`GCResNet-50`	resnest50d	`ResNeSt-50`	wide_resnet50_2	`Wide ResNet-50`
gcresnext50ts	`GCResNeXt-50`	resnet101	`ResNet-101`	xcit_medium_24_p16_224	`XCiT-M24`
gluon_resnet101_v1c	`ResNet-101-C`	resnet50	`ResNet-50`	xcit_tiny_12_p8_224	`XCiT-T12`

Table C.2: Overview of model elements. We categorize each architecture with 13 different architectural components. The full feature list is in the Appendix.

Components	Elements
Base architecture	CNN, Transformer, MLP-Mixer, Hybrid (CNN + Transformer), NAS-Net
Stem layer	7×7 conv with stride 2, 3×3 conv with stride 2, 16×16 conv with stride 16, ...
Input resolution	224×224, 256×256, 240×240, 299×299
Normalization layer	BN, GN, LN, LN + GN, LN + BN, Normalization-free, ...
Using hierarchical structure	Yes (*e.g.*, CNNs, Swin [18]), No (*e.g.*, ViT [8])
Activation functions	ReLU, HardSwish, SiLU, GeLU, ReLU + GeLU, ReLU + SiLU or GeLU ...
Using pooling at stem	Yes, No
Using 2D self-attention	Yes (*e.g.*, [2], [24], [29]), No
Using channel-wise (CW) attention	Yes (*e.g.*, [15], [3], [31]), No
Using depth-wise convolution	Yes, No
Using group convolution	Yes, No
Type of pooling for final feature	Classification (CLS) token, Global Average Pool (GAP)
Location of CW attentions	At the end of each block, in the middle of each block, ...

Table C.3: Description of features of 69 models. “s" in “Stem layer" indicates the stride of a layer in the stem, and the number before and after “s" are a kernel size and size of stride, respectively. For example, “3s2/3/3" means that the stem is composed of the first layer having 3 × 3 kernel with stride 2, the second layer having 3 × 3 kernel with stride 1, and the last layer having 3 × 3 with stride 1.

Model name	Base architecture	Hierarchical structure	Stem layer	Input resolution	Normalization	Activation
botnet26t_256	CNN	Yes	3s2/3/3	256 × 256	BN	ReLU
convmixer_1536_20	CNN	Yes	7s7	224 × 224	BN	GeLU
convnext_tiny	CNN	Yes	4s4	224 × 224	LN	GeLU
cspdarknet53	CNN	Yes	3s1	256 × 256	BN	Leaky ReLU
cspresnet50	CNN	Yes	7s2	256 × 256	BN	Leaky ReLU
cspresnext50	CNN	Yes	7s2	256 × 256	BN	Leaky ReLU
dla102x2	CNN	Yes	7s1	224 × 224	BN	ReLU
dpn107	CNN	Yes	7s2	224 × 224	BN	ReLU
eca_botnext26ts_256	CNN	Yes	3s2/3/3	256 × 256	BN	SiLU
eca_halonext26ts	CNN	Yes	3s2/3/3	256 × 256	BN	SiLU
eca_nfnet_l0	CNN	Yes	3s2/3/3/3s2	224 × 224	Norm-free	SiLU
eca_resnet33ts	CNN	Yes	3s2/3/3s2	256 × 256	BN	SiLU
ese_vovnet39b	CNN	Yes	3s2/3/3s2	224 × 224	BN	ReLU
gcresnet50t	CNN	Yes	3s2/3/3s2	256 × 256	LN + BN	ReLU
gcresnext50ts	CNN	Yes	3s2/3/3	256 × 256	LN + BN	ReLU + SiLU
gluon_resnet101_v1c	CNN	Yes	3s2/3/3	224 × 224	BN	ReLU
gluon_xception65	CNN	Yes	3s2/3	299 × 299	BN	ReLU
halo2botnet50ts_256	CNN	Yes	3s2/3/3s2	256 × 256	BN	SiLU
halonet50ts	CNN	Yes	3s2/3/3	256 × 256	BN	SiLU
hrnet_w64	CNN	Yes	3s2/3s2	224 × 224	BN	ReLU
lambda_resnet50ts	CNN	Yes	3s2/3/3	256 × 256	BN	SiLU
lamhalobotnet50ts_256	CNN	Yes	3s2/3/3s2	256 × 256	BN	SiLU
nf_resnet50	CNN	Yes	7s2	256 × 256	Norm-free	ReLU
nfnet_l0	CNN	Yes	3s2/3/3/3s2	224 × 224	Norm-free	ReLU + SiLU
poolformer_m48	CNN	Yes	7s4	224 × 224	GN	GeLU
resnest50d	CNN	Yes	3s2/3/3	224 × 224	BN	ReLU
resnet101	CNN	Yes	7s2	224 × 224	BN	ReLU
resnet50	CNN	Yes	7s2	224 × 224	BN	ReLU
resnet50_gn	CNN	Yes	7s2	224 × 224	GN	ReLU
resnetblur50	CNN	Yes	7s2	224 × 224	BN	ReLU
resnetv2_101	CNN	Yes	7s2	224 × 224	BN	ReLU
resnetv2_50	CNN	Yes	7s2	224 × 224	BN	ReLU
resnetv2_50d_evos	CNN	Yes	3s2/3/3	224 × 224	EvoNorm	-
resnext50_32x4d	CNN	Yes	7s2	224 × 224	BN	ReLU
sebotnet33ts_256	CNN	Yes	3s2/3/3s2	256 × 256	BN	ReLU + SiLU
sehalonet33ts	CNN	Yes	3s2/3/3s2	256 × 256	BN	ReLU + SiLU
seresnet50	CNN	Yes	7s2	224 × 224	BN	ReLU
seresnext50_32x4d	CNN	Yes	7s2	224 × 224	BN	ReLU
wide_resnet50_2	CNN	Yes	7s2	224 × 224	BN	ReLU
convit_base	Transformer	No	16s16	224 × 224	LN	GeLU
crossvit_base_240	Transformer	Yes	16s16	240 × 240	LN	GeLU
deit_base_patch16_224	Transformer	No	16s16	224 × 224	LN	GeLU
deit_small_patch16_224	Transformer	No	16s16	224 × 224	LN	GeLU
jx_nest_tiny	Transformer	Yes	4s4	224 × 224	LN	GeLU
pit_s_224	Transformer	Yes	16s8	224 × 224	LN	GeLU
swin_tiny_patch4_window7_224	Transformer	Yes	4s4	224 × 224	LN	GeLU
tnt_s_patch16_224	Transformer	Yes	7s4	224 × 224	LN	GeLU
vit_base_patch32_224	Transformer	No	32s32	224 × 224	LN	GeLU
vit_small_patch16_224	Transformer	No	16s16	224 × 224	LN	GeLU
gmlp_s16_224	MLP-Mixer	Yes	16s16	224 × 224	LN	GeLU
resmlp_24_224	MLP-Mixer	No	16s16	224 × 224	Affine transform	GeLU
resmlp_big_24_224	MLP-Mixer	Yes	8s8	224 × 224	Affine transform	GeLU
swin_s3_tiny_224	NAS (TFM)	Yes	4s4	224 × 224	LN	GeLU
efficientnet_b2	NAS (CNN)	Yes	3s2	256 × 256	BN	SiLU
fbnetv3_g	NAS (CNN)	Yes	3s2	240 × 240	BN	HardSwish
haloregnetz_b	NAS (CNN)	Yes	3s2	224 × 224	BN	ReLU + SiLU
mixnet_xl	NAS (CNN)	Yes	3s2	224 × 224	BN	ReLU + SiLU
nf_regnet_b1	NAS (CNN)	Yes	3s2	256 × 256	Norm-free	ReLU + SiLU
regnetx_320	NAS (CNN)	Yes	3s2	224 × 224	BN	ReLU
regnety_032	NAS (CNN)	Yes	3s2	224 × 224	BN	ReLU
rexnet_150	NAS (CNN)	Yes	3s2	224 × 224	BN	ReLU + SiLU + ReLU6
coat_lite_small	Hybrid	Yes	4s4	224 × 224	LN	GeLU
pit_b_224	Hybrid	Yes	14s7	224 × 224	LN	GeLU
twins_pcpvt_base	Hybrid	Yes	4s4	224 × 224	LN	GeLU
twins_svt_small	Hybrid	Yes	4s4	224 × 224	LN	GeLU
visformer_small	Hybrid	Yes	7s2	224 × 224	BN	GeLU + ReLU
vit_small_r26_s32_224	Hybrid	No	7s2	224 × 224	LN + GN	GeLU + ReLU
xcit_medium_24_p16_224	Hybrid	No	3s2/3s2/3s2/3s2	224 × 224	LN + BN	GeLU
xcit_tiny_12_p8_224	Hybrid	No	3s2/3s2/3s2	224 × 224	LN + BN	GeLU

Table C.4: Description of features of 69 models. “Pooling (stem)" and “Pooling (final)" denote “Pooling at the stem" and “Pooling for final feature", respectively. “SA", “CW", and “DW" means “Self-attention", “Channel-wise", and “Depth-wise", respectively.

Model name	Pooling (stem)	Pooling (final)	2D SA	CW attention	Location of CW attention	DW conv	Group conv
botnet26t_256	Yes	GAP	Yes (BoT)	No		No	No
convmixer_1536_20	No	GAP	No	No		Yes	No
convnext_tiny	No	GAP	No	No		Yes	No
cspdarknet53	No	GAP	No	No		No	No
cspresnet50	Yes	GAP	No	No		No	No
cspresnext50	Yes	GAP	No	No		No	Yes
dla102x2	No	GAP	No	No		No	Yes
dpn107	Yes	GAP	No	No		No	Yes
eca_botnext26ts_256	Yes	GAP	Yes (BoT)	Yes (ECA)	Middle	No	Yes
eca_halonext26ts	Yes	GAP	Yes (Halo)	Yes (ECA)	Middle	No	Yes
eca_nfnet_l0	No	GAP	No	Yes (ECA)	End	No	Yes
eca_resnet33ts	No	GAP	No	Yes (ECA)	Middle	No	No
ese_vovnet39b	No	GAP	No	Yes (ESE)	End	No	No
gcresnet50t	No	GAP	No	Yes (GCA)	Middle	Yes	No
gcresnext50ts	Yes	GAP	No	Yes (GCA)	Middle	Yes	Yes
gluon_resnet101_v1c	Yes	GAP	No	No		No	No
gluon_xception65	No	GAP	No	No		Yes	No
halo2botnet50ts_256	No	GAP	Yes (Halo, BoT)	No		No	No
halonet50ts	Yes	GAP	Yes (Halo)	No		No	No
hrnet_w64	No	GAP	No	No		No	No
lambda_resnet50ts	Yes	GAP	Yes (Lambda)	No		No	No
lamhalobotnet50ts_256	No	GAP	Yes (Lambda, Halo, BoT)	No		No	No
nf_resnet50	Yes	GAP	No	No		No	No
nfnet_l0	No	GAP	No	Yes (SE)	End	No	Yes
poolformer_m48	No	GAP	No	No		No	No
resnest50d	Yes	GAP	No	Yes		No	Yes
resnet101	Yes	GAP	No	No		No	No
resnet50	Yes	GAP	No	No		No	No
resnet50_gn	Yes	GAP	No	No		No	No
resnetblur50	Yes	GAP	No	No		No	No
resnetv2_101	Yes	GAP	No	No		No	No
resnetv2_50	Yes	GAP	No	No		No	No
resnetv2_50d_evos	Yes	GAP	No	No		No	No
resnext50_32x4d	Yes	GAP	No	No		No	Yes
sebotnet33ts_256	No	GAP	Yes (BoT)	Yes (SE)	Middle	No	No
sehalonet33ts	No	GAP	Yes (Halo)	Yes (SE)	Middle	No	No
seresnet50	Yes	GAP	No	Yes (SE)	End	No	No
seresnext50_32x4d	Yes	GAP	No	Yes (SE)	End	No	Yes
wide_resnet50_2	Yes	GAP	No	No		No	No
convit_base	No	CLS token	No	No		No	No
crossvit_base_240	No	CLS token	No	No		No	No
deit_base_patch16_224	No	CLS token	No	No		No	No
deit_small_patch16_224	No	CLS token	No	No		No	No
jx_nest_tiny	No	GAP	No	No		No	No
pit_s_224	No	CLS token	No	No		Yes	No
swin_tiny_patch4_window7_224	No	GAP	No	No		No	No
tnt_s_patch16_224	No	CLS token	No	No		No	No
vit_base_patch32_224	No	CLS token	No	No		No	No
vit_small_patch16_224	No	CLS token	No	No		No	No
gmlp_s16_224	No	GAP	No	No		No	No
resmlp_24_224	No	GAP	No	No		No	No
resmlp_big_24_224	No	GAP	No	No		No	No
swin_s3_tiny_224	No	GAP	No	No		No	No
efficientnet_b2	No	GAP	No	Yes (SE)	Middle	Yes	No
fbnetv3_g	No	GAP	No	Yes (SE)	Middle	Yes	No
haloregnetz_b	No	GAP	Yes (Halo)	Yes (SE)	Middle	No	Yes
mixnet_xl	No	GAP	No	Yes (SE)	Middle	Yes	No
nf_regnet_b1	No	GAP	No	Yes (SE)	Middle	Yes	Yes
regnetx_320	No	GAP	No	No		No	Yes
regnety_032	No	GAP	No	Yes (SE)	Middle	No	Yes
rexnet_150	No	GAP	No	Yes (SE)	Middle	Yes	No
coat_lite_small	No	CLS token	No	No		Yes	No
pit_b_224	No	CLS token	No	No		Yes	No
twins_pcpvt_base	No	GAP	No	No		Yes	No
twins_svt_small	No	GAP	No	No		Yes	No
visformer_small	No	GAP	No	No		No	Yes
vit_small_r26_s32_224	Yes	CLS token	No	No		No	No
xcit_medium_24_p16_224	No	CLS token	No	No		Yes	No
xcit_tiny_12_p8_224	No	CLS token	No	No		Yes	No

D Feature Importance and Clustering Details

D.1 Feature Importance Analysis Details

We fit a gradient boosting regressor [9] based on `Scikit-learn` [22] and report the permutation importance of each architectural component in Fig. 3 of the main paper. The number of boosting stages, maximum depth, minimum number of samples, and learning rate are set to 500, 12, 4, and 0.02, respectively. Permutation importance is computed by permuting a feature 10 times.

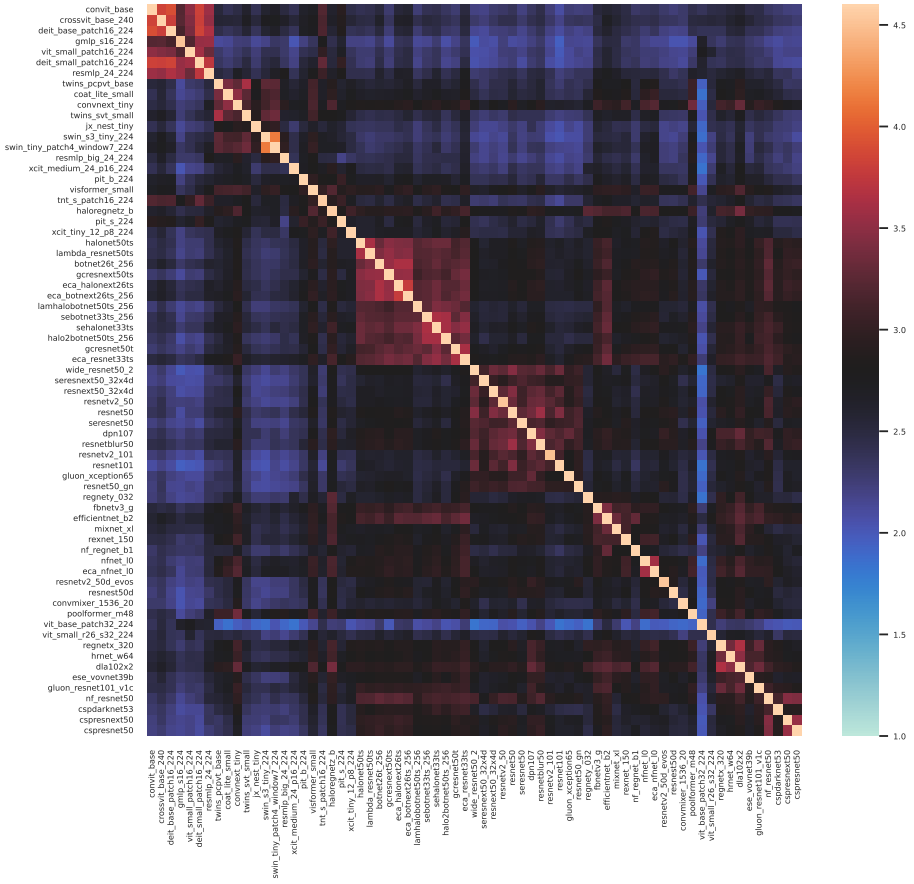


Fig. D.1: Pairwise similarity among 69 models. Rows and columns are sorted by the clustering index in Tab. 2. (n, n) component of pairwise similarity is close to 4.6 ($\log 100$) because the attack success rate is almost 100% when a model used for generating adversarial perturbation and attacked model are the same.

D.2 Pairwise Similarities

Fig. D.1 indicates the pairwise similarity among 69 models. We can observe a weak block pattern around clusters, as also revealed in Fig. D.2.

D.3 Spectral Clustering Details

We use the normalized Laplacian matrix to compute the Laplacian matrix. We also run K-means clustering 100 times and choose the clustering result with the best final objective function to reduce the randomness by the K-means clustering.

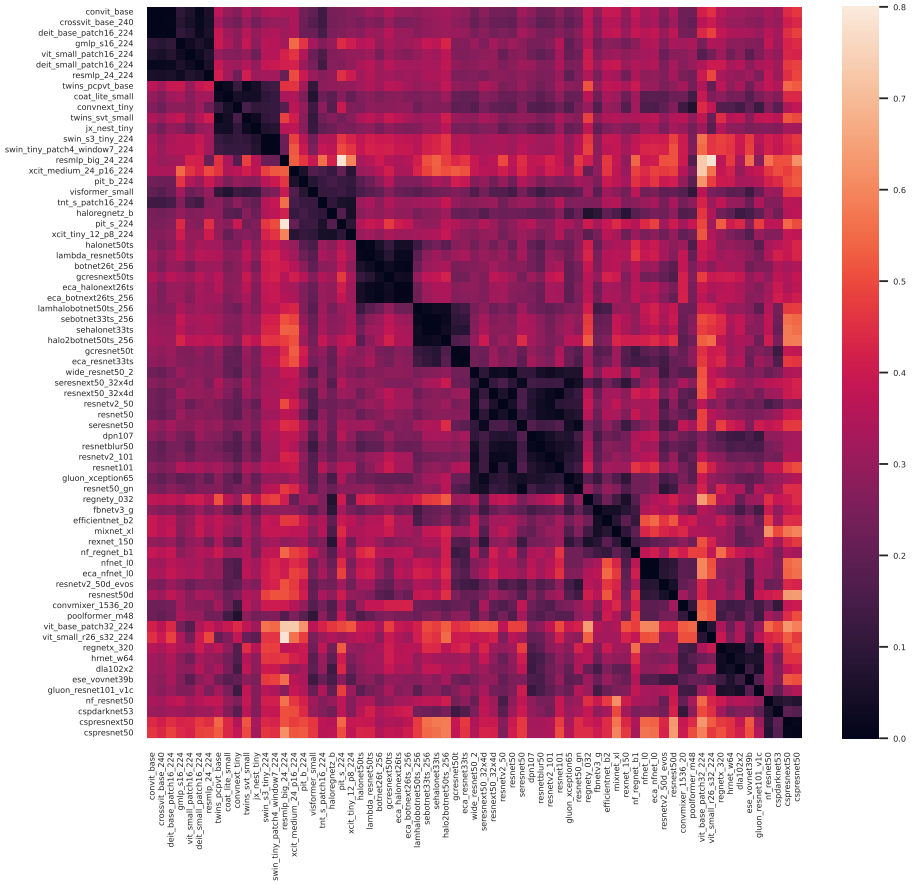


Fig. D.2: Spectral features of 69 architectures. The K-th largest eigenvectors of the Laplacian matrix of the pairwise similarity graph of 69 architectures are shown ($K = 10$ in this figure). Rows and columns are sorted by the clustering index in Tab. 2. We denote the model name in `timm` for each row and column.

We visualize the pairwise distances of the spectral features (*i.e.*, K-largest eigenvectors of L) of 69 architectures with their model names in Fig. D.2. This figure is an extension of Fig. 4, now including the model names. Note that rows and columns of Fig. D.2 are sorted by the clustering results. Fig. D.2 shows block diagonal patterns, *i.e.*, in-cluster similarities are large while between-cluster similarities are small.

E Training Settings for Models Used in Analyses

Models with various training methods for Sec. 4.1. We train 21 `ResNet-50` models and 16 `ViT-S` from scratch individually by initializing each network with different random seeds. We further train 28 `ResNet-50` models by randomly choosing learning rate ($\times 0.1$, $\times 0.2$, $\times 0.5$, $\times 1$, $\times 2$, and $\times 5$ where the base learning rate is 0.1), weight decay ($\times 0.1$, $\times 0.2$, $\times 0.5$, $\times 1$, $\times 2$, and $\times 5$ where the base weight decay is 1e-4), and learning rate scheduler (step decay or cosine decay). Similarly, we train 9 `ViT-S` models by randomly choosing learning rate ($\times 0.2$, $\times 0.4$, and $\times 1$ where the base learning rate is 5e-4) and weight decay ($\times 0.2$, $\times 0.4$, and $\times 1$ where the base weight decay is 0.05). Note that the DeiT training is unstable when we use a larger learning rate or weight decay than the base values. Finally, we collect 22 `ResNet-50` models with different training regimes: 1 model with standard training by PyTorch [21]; 4 models trained by GluonCV [13][1]; a semi-supervised model and semi-weakly supervised model on billion-scale unlabeled images by Yalniz *et al.* [36][2]; 5 models trained by different augmentation methods (Cutout [6], Mixup [40], manifold Mixup [30], CutMix [38], and feature CutMix[3]; 10 optimized ResNet models by [35][4]. We also collect 7 `ViT-S` models with different training regimes, including the original ViT training setup [8][5], a stronger data augmentation setup in the Deit paper [25]-3[5], the training setup with distillation [25]-3[5], an improved DeiT training setup [26]-3[5] , and self-supervised training fashions by MoCo v3 [4][6], MAE [14][7] and BYOL [11][8]. We do not use adversarially-trained networks because the adversarial training usually drops the standard accuracy significantly [28].

Distillation models for Sec. 5. We train `ViT-Ti` student models with 25 different teacher models using hard distillation strategy. We follow the distillation training setting of DeiT official repo[9], only changing the teacher model. Note that we

[1] gluon_resnet50_v1b, gluon_resnet50_v1c, gluon_resnet50_v1d, and gluon_resnet50_v1s from `timm` library.

[2] ssl_resnet50 and swsl_resnet50 from `timm` library.

[3] We use the official weights provided by `https://github.com/clovaai/CutMix-PyTorch`.

[4] We use the official weights provided by `https://github.com/rwightman/pytorch-image-models/releases/tag/v0.1-rsb-weights`

[5] deit_small_patch16_224, vit_small_patch16_224, deit_small_distilled_patch16_224, and deit3_small_patch16_224 from `timm` library.

[6] We train the `ViT-S` model by following `https://github.com/facebookresearch/moco-v3`

[7] We train the `ViT-S` model by following `https://github.com/facebookresearch/mae`

[8] We train the `ViT-S` model by following `https://github.com/lucidrains/byol-pytorch`

[9] `https://github.com/facebookresearch/deit`.

resize the input images to the input size the teacher model requires if the input sizes of student and teacher models differ. If a teacher model needs a different input resolution, such as 240×240 and 256×256, we resize the input image for distilling it. Because `DeiT-Ti` has low classification accuracy compared to teacher models, the similarity score is calculated between `DeiT-S` and 25 models. The 25 teacher models are as follows: `BoTNet-26`, `CoaT-Lite Small`, `ConViT-B`, `ConvNeXt-B`, `CrossViT-B`, `CSPDarkNet-53`, `CSPResNeXt-50`, `DLA-X-102`, `DPN-107`, `EfficientNet-B2`, `FBNetV3-G`, `GC-ResNet-50`, `gMLP-S`, `HaloRegNetZ`, `MixNet-XL`, `NFNet-L0`, `PiT-S`, `RegNetY-032`, `ResMLP-24`, `ResNet-50`, `SEHaloNet33`, `Swin-T`, `TNT-S`, `VisFormer-S`, and `XCiT-T12`.

F Knowledge Distillation

F.1 Teacher Accuracy and Distillation Performance

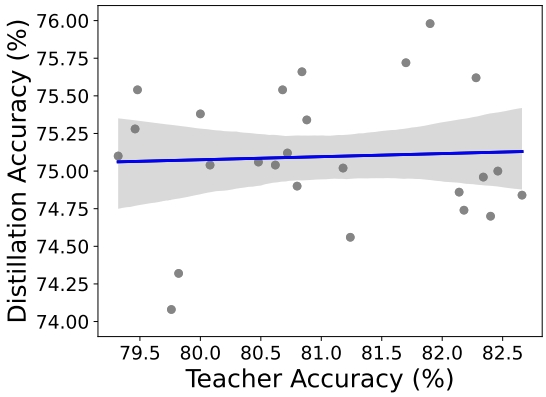
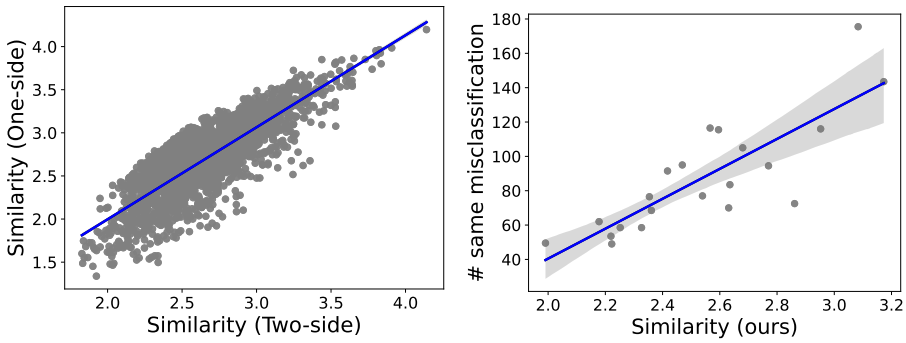


Fig. F.1: Teacher network accuracy and distillation performance. There is no significant correlation between teacher accuracy and distillation performance.

The similarity between teacher and student networks may not be the only factor contributing to distillation. For example, a stronger teacher can lead to better distillation performance [39]. In Fig. F.1, we observe that if the range of the teacher accuracy is not significantly large enough (*e.g.*, between 79.5 and 82.5), then the correlation between teacher network accuracy and distillation performance is not significant; 0.049 Pearson correlation coefficient with 0.82 p-value. In this case, we can conclude that the teacher and student networks similarity contributes more to the distillation performance than the teacher performance.



(a) **Approximated (one-side) SAT vs. original (two-side).** (b) **SAT vs. The number of same misclassification.**

Fig. G.1: Additional Analysis for SAT.

G Limtations and Discussions

G.1 Efficient Approximation of SAT for a Novel Model

We can use our toolbox for designing a new model; we can measure SAT between a novel network and existing N architectures; a novel network can be assigned to clusters (Tab. 1) to understand how it works. However, it requires generating adversarial samples for all $N+1$ models (*e.g.*, 70 in our case), which is computationally inefficient. Instead, we propose the approximation of Eq. 1 by omitting to compute the accuracy of the novel network on the adversarial samples of the existing networks. It will break the symmetricity of SAT, but we found that the approximated score and the original score have high similarity – 0.82 Pearson coefficient with almost 0 p-value – as shown in Fig. G.1a.

As an example, we tested `Gluon-ResNeXt-50` [13] and the distilled version of `DeiT-S` [25]. As observed in Tab. 2 and Fig. 5, models with the same architecture have high similarity compared to models with different architectures; hence, we expect that `Gluon-ResNeXt-50` is assigned to the same cluster with `ResNeXt-50`, and distilled `DeiT-S` is assigned to the same cluster with `DeiT-S`. As we expected, each network is assigned to the desired cluster. Therefore, we suggest using our efficient approximation for analyzing a novel network with our analysis toolbox.

G.2 Adversarial Attack Transferability and Direction of Missclassification

Waseda et al. [33] showed that adversarial attack transferability is highly related to the direction of the misclassification. We examine if SAT is related to the misclassification. Fig. G.1b shows the relationship between SAT and the number of the same misclassification by the attack. We observe that they are highly correlated, *i.e.*, we confirmed that SAT is also related to the misclassification.

G.3 Change of SAT During Training

We check the adversarial attack transferability between the fully trained model and less trained models on CIFAR-10 with 180 training epochs. A model trained with only 20 epochs shows high similarity over different initializations (4.23 in Tab. 2 of the main paper). Note that SAT considers models having similar clean accuracy; namely, there is room to explore this further in future work.

Table G.1: Change of SAT between fully-trained model (epoch 180) and models on various epochs.

Epoch	0	20	40	60	80	100	120	140	160	180
SAT	2.84	4.46	4.56	4.58	4.59	4.59	4.60	4.60	4.60	4.60

G.4 More Possible Applications Requiring Diverse Models

In the main paper, we introduce several applications with multiple models, such as model ensemble, knowledge distillation, and novel model development. As another example, we employ SAT-based diverse model selection for improving the dataset distillation (DD) task with random network selection [41]. DD task [17, 37, 41, 43, 44] aims to synthesize a small (usually less than 5 images per class) but informative dataset that prevents a significant drop from the original performance. Acc-DD [41] employs multiple random networks for DD, where each network is randomly selected during the training. In this study, we show that a more diverse network selection can help synthesize more informative and diverse condensed images. We replace the random selection of Acc-DD (`Rand`) with the selection by the probability proportional to (1) the similarity ($\texttt{P}_{\texttt{sim}}$) or (2) the inverse of similarity ($\texttt{P}_{\texttt{sim}^{-1}}$). More specifically, we first (a) select a network randomly and (b) select the next network by (1) or (2) with the current network. We repeat (b) similar to K-means++ [1]. We report the CIFAR-10 results by setting images per class as 1 using 50 CNNs. `Rand` shows 48.6 top-1 accuracy, while $\texttt{P}_{\texttt{sim}}$ and $\texttt{P}_{\texttt{sim}^{-1}}$ show 48.7 and **49.4**, respectively. Namely, a more diverse network selection ($\texttt{P}_{\texttt{sim}^{-1}}$) helps Acc-DD.

References

[1] David Arthur, Sergei Vassilvitskii, et al. k-means++: The advantages of careful seeding. In *Soda*, pages 1027–1035, 2007.

[2] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *Int. Conf. Learn. Represent.*, 2021.

[3] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Int. Conf. Comput. Vis. Worksh.*, 2019.

[4] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *Int. Conf. Comput. Vis.*, 2021.

[5] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Int. Conf. Mach. Learn.*, 2020.

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

[7] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. Learn. Represent.*, 2021.

[9] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[10] Yonggan F Fu, Shang Wu, Yingyan Lin, et al. Patch-fool: Are vision transformers always robust against adversarial perturbations? *Int. Conf. Learn. Represent.*, 2022.

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *Adv. Neural Inform. Process. Syst.*, 2020.

[12] Chuan Guo, Jared S Frank, and Kilian Q Weinberger. Low frequency adversarial perturbation. *UAI*, 2019.

[13] Jian Guo, He He, Tong He, Leonard Lausen, Mu Li, Haibin Lin, Xingjian Shi, Chenguang Wang, Junyuan Xie, Sheng Zha, Aston Zhang, Hang Zhang, Zhi Zhang, Zhongyue Zhang, Shuai Zheng, and Yi Zhu. Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. *Journal of Machine Learning Research*, 21(23):1–7, 2020.

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[17] Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. Dataset condensation with contrastive signals. In *International Conference on*

Machine Learning (ICML), 2022.

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis.*, 2021.

[19] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Int. Conf. Learn. Represent.*, 2018.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Adv. Neural Inform. Process. Syst.*, 2019.

[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, and Tom Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[24] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[25] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Int. Conf. Mach. Learn.*, 2021.

[26] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. *arXiv preprint arXiv:2204.07118*, 2022.

[27] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[28] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *Int. Conf. Learn. Represent.*, 2019.

[29] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[30] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *Int. Conf. Mach. Learn.*, 2019.

[31] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[32] Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1924–1933, 2021.

[33] Futa Waseda, Sosuke Nishikawa, Trung-Nghia Le, Huy H Nguyen, and Isao Echizen. Closer look at the transferability of adversarial examples: How they fool

different models differently. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023.
[34] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.
[35] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. In *Adv. Neural Inform. Process. Syst. Worksh.*, 2021.
[36] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.
[37] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *arXiv preprint arXiv:2301.07014*, 2023.
[38] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Int. Conf. Comput. Vis.*, 2019.
[39] Sangdoo Yun, Seong Joon Oh, Byeongho Heo, Dongyoon Han, Junsuk Choe, and Sanghyuk Chun. Re-labeling imagenet: from single to multi-labels, from global to localized labels. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.
[40] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Int. Conf. Learn. Represent.*, 2018.
[41] Lei Zhang, Jie Zhang, Bowen Lei, Subhabrata Mukherjee, Xiang Pan, Bo Zhao, Caiwen Ding, Yao Li, and Dongkuan Xu. Accelerating dataset distillation via model augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11950–11959, 2023.
[42] Qilong Zhang, Xiaodan Li, Yuefeng Chen, Jingkuan Song, Lianli Gao, Yuan He, and Hui Xue. Beyond imagenet attack: Towards crafting adversarial examples for black-box domains. *arXiv preprint arXiv:2201.11528*, 2022.
[43] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*, pages 12674–12685. PMLR, 2021.
[44] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.