SIMILARITY OF NEURAL ARCHITECTURES BASED ON INPUT GRADIENT TRANSFERABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we aim to design a quantitative similarity function between two neural architectures. Specifically, we define a model similarity using input gradient transferability. We generate adversarial samples of two networks and measure the average accuracy of the networks on adversarial samples of each other. If two networks are highly correlated, then the attack transferability will be high, resulting in high similarity. Using the similarity score, we investigate two topics: (1) Which network component contributes to the model diversity? (2) How does model diversity affect practical scenarios? We answer the first question by providing feature importance analysis and clustering analysis. The second question is validated by two different scenarios: model ensemble and knowledge distillation. Our findings show that model diversity takes a key role when interacting with different neural architectures. For example, we found that more diversity leads to better ensemble performance. We also observe that the relationship between teacher and student networks and distillation performance depends on the choice of the base architecture of the teacher and student networks. We expect our analysis tool helps a high-level understanding of differences between various neural architectures as well as practical guidance when using multiple architectures.

1 INTRODUCTION

The last couple of decades have seen the destructive success of deep neural networks (DNNs) in real-world applications, *e.g.*, image classification (He et al., 2016a) and natural language processing (Vaswani et al., 2017). The advances in the DNN architecture design have taken a key role in this success by making the learning process easier (*e.g.*, normalization methods (Ioffe & Szegedy, 2015; Wu & He, 2018; Ba et al., 2016) or skip connection (He et al., 2016a)) enforcing human inductive bias in the architecture (*e.g.*, convolutional neural networks (CNNs) (Krizhevsky et al., 2012; Simonyan & Zisserman, 2015)) or increasing model capability with the self-attention mechanism (*e.g.*, Transformers (Vaswani et al., 2017)). With different design principles and architectural elements, there have been proposed a number of different neural architectures; however, designing a distinguishable architecture is expensive and needs heavy expertise. One of the reasons for the difficulty is that there is only a little knowledge of the difference between two different neural architectures. Especially, if one can quantify the similarity between two models, then we can measure which design component actually contributes a lot to distinct properties. We also can utilize the quantity for new model design (*e.g.*, by neural architecture search (NAS) (Zoph & Le, 2017)).

In this paper, we aim to define the similarity between two networks to quantify the difference and diversity between neural architectures. Existing studies have focused on dissecting each network component layer-by-layer (Kornblith et al., 2019; Raghu et al., 2021) or providing a high-level understanding by visualization of loss surface (Dinh et al., 2017), input gradient (Springenberg et al., 2015; Smilkov et al., 2017), or decision boundary (Somepalli et al., 2022). On the other hand, we aim to design an architecture agnostic and quantitative score to measure the difference between the two architectures. We especially focus on the input gradients frequently utilized to model understanding (Sung, 1998; Simonyan et al., 2014; Springenberg et al., 2015; Smilkov et al., 2017; Sundararajan et al., 2017; Bansal et al., 2020; Choe et al., 2022). To quantify the impact of input gradients, we generate adversarial samples for each architecture and measure the *attack transferability* between different architectures. Consider two models A and B, and the input x. We generate adversarial samples x_A and x_B to A and B, respectively. Then we measure the accuracy of B for x_A ($\text{acc}_{A \to B}$)

and vice versa. If A and B are similar and assume an optimal adversary, then $\text{acc}_{A \to B}$ will be almost zero, while if A and B have distinct input gradients, then $\text{acc}_{A \to B}$ will not be dropped significantly. We define a neural architecture similarity based on attack transferability and analyze the existing neural architectures, *e.g.*, which network component affects the model diversity the most?

We measure the pairwise attack transferability-based network similarity of 69 different neural architectures trained on ImageNet (Russakovsky et al., 2015), provided by Wightman (2019). Our first goal is to understand the effect of each network module on model diversity. We first choose 13 basic components (*e.g.*, normalization layers, activation functions, the design choice for stem layers) that consist of neural architectures and list up the components of 69 networks. For example, we represent `ResNet` (He et al., 2016a) as $f_{\text{ResNet}} = [\text{Base architecture} = \text{CNN}, \text{Norm} = \text{BN}, \text{Activation} = \text{ReLU}, \ldots]$. We analyze the contribution of each network component to the model diversity by using the feature importance analysis by gradient boosting regression on model similarities, and the clustering analysis on model similarities. Our analyses show that the choice of base architecture (*e.g.*, CNN, Transformer) contributes most to the network diversity. Interestingly, our analysis shows that the design choice for the input-level layers (*e.g.*, stem layer, input resolution) determines the network diversity as much as the choice of core modules (*e.g.*, normalization layers, activation functions).

Our study is not only limited to the understanding of component-level architecture design, but we also focus on analyzing the effect of model diversity in practical scenarios. Particularly, we measure the *model ensemble performances* by controlling the diversity of the candidate models, *e.g.*, ensemble "similar" or "dissimilar" models by our similarity score. Here, we observe that when we ensemble more dissimilar models, the ensemble accuracy gets better; *more diversity leads to a better ensemble performance*. We observe that the diversity caused by different initialization, different hyper-parameter choice, and different training regimes is not as significant as the diversity caused by architecture change. We also observe that the ensemble of models from the same cluster performs worse than the ensemble of models from different clusters. Similarly, by choosing more diverse models, the number of wrong samples by all models is getting decreased.

As our third contribution, we provide a practical guideline for the choice of a teacher network for knowledge distillation (KD) (Hinton et al., 2015). We train 25 distilled `ViT-Ti` models with diverse teacher networks. Interestingly, our findings show that the performance of the distilled model is highly correlated to the similarity between the teacher and the student networks rather than the accuracy of the teacher network; if the student networks and the teacher network are based on the same architecture (*e.g.*, both are based on Transformer), then a similar teacher provides better knowledge; if the student and teacher networks are based on different architectures (*e.g.*, Transformer and CNN), then we observe that choosing a more dissimilar teacher will lead to a better distillation performance. Our findings are partially aligned with previous KD studies, *i.e.*, using a similar teacher network leads to a better KD performance (Jin et al., 2019; Mirzadeh et al., 2020). However, the existing studies only focus on the scenario when both teacher and student networks are based on the same architecture, exactly aligned with our findings. Our observation extends previous knowledge to the case of when the teacher and student networks significantly differ (*e.g.*, Transformer and CNN).

2 Network similarity by input gradient transferability

In this section, we propose a similarity measure between two networks based on adversarial attack transferability. Our interest lies in developing a practical toolbox to measure the architectural difference between the two models quantitatively. Existing studies have focused on comparing intermediate features (Kornblith et al., 2019; Raghu et al., 2021), comparing the wrong and correct predictions (Geirhos et al., 2018; 2020; Scimeca et al., 2022), understanding loss landscapes (Dinh et al., 2017; Li et al., 2018; Park & Kim, 2022), or decision boundary (Somepalli et al., 2022). However, their approaches cannot measure the distance between two different networks (Li et al., 2018; Kornblith et al., 2019) or need expensive computation costs for re-training networks multiple times (Somepalli et al., 2022). Input gradient (Sung, 1998; Simonyan et al., 2014) is another popular visualization tool for understanding DNNs by highlighting where a model attends to. An input gradient-based method has several advantages; it is computationally efficient, and no additional training is required; it can provide a visual understanding of the given input. Usually, an input gradient-based quantity measures how the input gradient of a model matches the actual foreground (Choe et al., 2022), *i.e.*, it usually measures how a model explains the input well. Because we are interested in comparing two

Table 1: **Overview of model elements.** We categorize each architecture with 13 different sub-components. The full feature list of each architecture is in Appendix A.

Sub-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 (Liu et al., 2021b)), No (*e.g.*, ViT (Dosovitskiy et al., 2021))
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.*, Bello (2021a), Srinivas et al. (2021b), Vaswani et al. (2021b)), No
Using channel-wise (CW) attention	Yes (*e.g.*, Hu et al. (2018a), Cao et al. (2019a), Wang et al. (2020c)), 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, ...

different models, we focus on how input gradients are similar between them. Especially, we generate an adversarial sample using a model and test the adversarial sample to the other architecture.

Formally, we generate adversarial samples x_A and x_B of model A and B for the given input x. Then, we measure the accuracy of model A using the adversarial sample for model B (called $\text{acc}_{B \to A}$). If A and B are the same, then $\text{acc}_{B \to A}$ will be zero (for simplicity, here, we may assume that the adversary can fool a model perfectly). On the other hand, if the input gradients of A and B differ significantly, then the performance drop will be neglectable because the adversarial sample is made almost similar to the original image (*i.e.*, $\|x - x_B\| \leq \varepsilon$). Let X_{AB} be the set of inputs where both A and B predict correctly, y be the ground truth label, and $\mathbb{I}(\cdot)$ be the indicator function. We use the following quantity to measure the similarity between two models:

$$s(A, B) = \log \left[\max \left\{ \varepsilon_s, 100 \times \frac{1}{2|X_{AB}|} \sum_{x \in X_{AB}} \{\mathbb{I}(A(x_B) \neq y) + \mathbb{I}(B(x_A) \neq y)\} \right\} \right], \quad (1)$$

where ε_s is a small scalar value to prevent the ill-posedness of $s(\cdot)$. If $A = B$ and we have an oracle adversary, then $s(A, A) = \log 100$. In practice, a strong adversary (*e.g.*, PGD (Madry et al., 2017) or AutoAttack (Croce & Hein, 2020)) can easily achieve a nearly-zero accuracy if a model is not trained by an adversarial attack-aware strategy, *e.g.*, adversarial training (Madry et al., 2017; Cohen et al., 2019). On the other hand, if the adversarial attacks on A are not transferable to B and vice versa, then $s(A, B) = \log \varepsilon_s$. We employ equation 1 as the similarity score between two networks. Now, we provide an extensive analysis of the existing networks using this quantity.

Settings. We use the projected gradient descent (PGD) (Madry et al., 2017) for the adversary. We set the iteration to 50, the learning rate to 0.1, and ε to 8/255 for the PGD attack. We select 69 neural architectures trained on ImageNet (Russakovsky et al., 2015) from the PyTorch Image Models library (Wightman, 2019)[1]. Notice that to reduce the unexpected effect of a significant accuracy gap, the chosen model candidates are limited to the models whose top-1 accuracy is between 79% and 83%. We also ignore the models with unusual training techniques, such as training on extra training datasets, using a small or large model input resolution (*e.g.*, less than 200 or larger than 300), or knowledge distillation. When A and B take different input resolutions, then we resize the attacked image from the source network for the target network. We also sub-sample 10% ImageNet validation images (*i.e.*, 5,000 images) to measure the similarity. This strategy makes our similarity score more computationally efficient. The full pairwise similarities of 69 models can be found in Appendix C.

3 MODEL ANALYSIS BY NETWORK SIMILARITY

We analyze how each model component affects the diversity between networks quantitatively by feature importance analysis and clustering analysis (Sec. 3.1). We also show that the diversity caused by different training strategies is not as significant as caused by different neural architectures (Sec. 3.2).

[1] https://github.com/rwightman/pytorch-image-models

3.1 WHICH ARCHITECTURAL COMPONENT CAUSES THE DIFFERENCE MOST?

Settings. We list 13 components of each architecture, *e.g.*, normalization (*e.g.*, BN (Ioffe & Szegedy, 2015) and LN (Ba et al., 2016)), activation functions (*e.g.*, ReLU (Krizhevsky et al., 2012) and GeLU (Ramachandran et al., 2017)) the existence of depthwise convolution, or stem layer (*e.g.*, 7×7 conv, 3×3 conv, or 16×16 conv with stride 16 – a.k.a. *"patchify"* stem (Liu et al., 2022b)). The list of sub-modules is shown in Tab. 1 and Appendix A. We then convert each architecture as a feature vector based on the listed sub-modules. For example, we convert ResNet as $f_{\text{ResNet}} = [\text{Base arch} = \text{CNN}, \text{Norm} = \text{BN}, \text{Activation} = \text{ReLU}, \ldots]$.

Feature important analysis. The full list of components of 69 architectures is in Appendix A. Now, we measure the feature importance by fitting a gradient boosting regressor (Friedman, 2001) on the feature difference (*e.g.*, $f_{\text{ResNet-50}} - f_{\text{DeiT-base}}$) measured by Hamming distance and the corresponding similarity (*e.g.*, 2.3). The details of the regressor are in Appendix A. We use the permutation importance (Breiman, 2001); We measure how the trained regression model changes the predicted similarity by randomly changing each feature (*e.g.*, normalization). The feature importance of each architecture component is shown in Fig. 1. We first observe that the choice of base architecture (*e.g.*, CNN (Krizhevsky et al., 2012), Transformer (Vaswani et al., 2017), and MLP-Mixer (Tolstikhin et al., 2021)) contributes to the similarity most significantly. Fig. 1 also shows that the design choice of the input layer (*i.e.*, stem layer design choice or input resolution) affects the similarity as much as the choice of basic components such as normalization layers, activation functions, and the existence of attention layers. On the other hand, we observe that the modified efficiency-aware convolution operations, such as depth-wise convolution (Chollet, 2017a) or group convolution (Xie et al., 2017b), are not effective for model diversity.

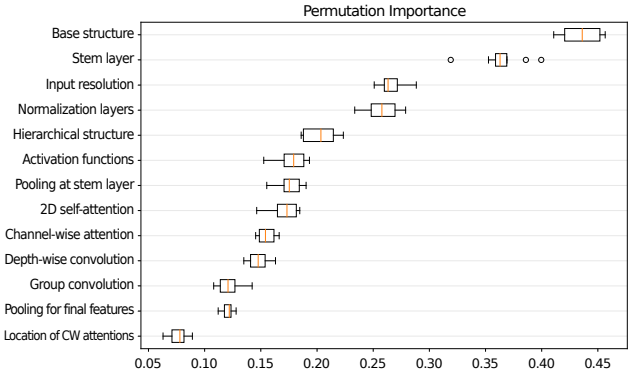


Figure 1: **Importance of each architecture component to network similarity.** 13 components (Tab. 1) are sorted by the contribution to the network similarities. The larger feature importance means the component contributes more to the network similarity.

Clustering analysis. We additionally provide a clustering analysis based on the architecture similarities. We construct a pairwise similarity graph A between all 69 architectures where its vertex denotes an architecture, and its edge denotes the similarity between two networks. We perform the spectral clustering algorithm (Ng et al., 2001) on A where the number of clusters K is set to 10: We compute the Laplacian matrix of A, $L = D - A$ where D is the diagonal matrix and its i-th component is $\sum_j A_{ij}$. Then, we perform K-means clustering on the K-largest eigenvectors of L, where K is the number of clusters (*i.e.*, $K = 10$). More details of spectral clustering are in Appendix B.

Tab. 2 shows the clustering results on 69 networks and the top-5 keywords for each cluster based on term frequency-inverse document frequency (TF-IDF) analysis. Specifically, we treat each model feature (Tab. 1) as a word and compute TF and IDF by treating each architecture as a document. Then we compute the average TF-IDF for each cluster and report top-5 keywords. Similar to Fig. 1, the base architecture (*e.g.*, CNN in Cluster 5, 6, 10 and Transformer in Cluster 2, 3) and the design choice for the stem layer (*e.g.*, Cluster 1, 2, 4, 5, 6, 7, 8, 10) repeatedly appear at the top keywords. Especially, we can observe that the differences in base architecture significantly cause the diversity in model similarities, *e.g.*, non-hierarchical Transformers (Cluster 1), hierarchical networks with the patchified stem (Cluster 2), hierarchical Transformers (Cluster 3), CNNs with 2D self-attention (Cluster 4, 5), ResNet-based architectures (Cluster 6), and NAS-based architectures (Cluster 7).

3.2 TRAINING STRATEGY AND SIMILARITY SCORE

The architectural difference is not the only cause of the model diversity. We compare the effect by different architecture choices (*e.g.*, ResNet and Vision Transformer) and by different training strate-

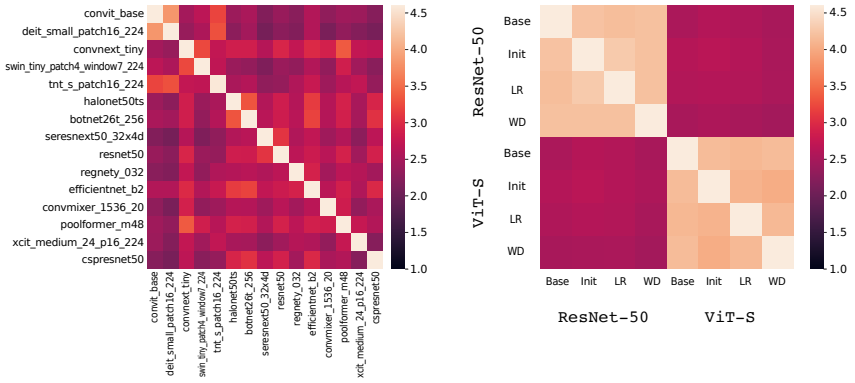
Table 2: **Clustering by architecture similarities.** We perform a clustering analysis on 69 neural architectures. All the architectures here are denoted by the aliases defined in their respective papers. We show the top-5 keywords for each cluster based on TF-IDF. InRes, SA, and CWA denote input resolution, self-attention, and channel-wise attention, respectively. The customized model details are described in the footnote†.

No.	Top-5 Keywords	Architecture
1	Stem layer: 16×16 conv w/ s16, No Hierarchical, GeLU, LN, Final GAP	`ConViT-B` (d'Ascoli et al., 2021), `CrossViT-B` (Chen et al., 2021a), `DeiT-B` (Touvron et al., 2021), `DeiT-S` (Touvron et al., 2021), `ViT-S (patch size 16)` (Dosovitskiy et al., 2021), `ResMLP-S24` (Touvron et al., 2022a), `gMLP-S` (Liu et al., 2021a)
2	Stem layer: 4×4 conv w/ s4, LN, GeLU, Transformer, No pooling at stem	`Twins-PCPVT-B` (Chu et al., 2021), `Twins-SVT-S` (Chu et al., 2021), `CoaT-Lite Small` (Dai et al., 2021b), `NesT-T` (Zhang et al., 2022b), `Swin-T` (Liu et al., 2021b), `S3 (Swin-T)` (Chen et al., 2021b), `ConvNeXt-T` (Liu et al., 2022a), `ResMLP-B24` (Touvron et al., 2022a)
3	Transformer, Final GAP, GeLU, Pooling at stem, InRes: 224	`XCiT-M24` (Ali et al., 2021), `XCiT-T12` (Ali et al., 2021), `HaloRegNetZ-B`†, `TNT-S` (Han et al., 2021b), `Visformer-S` (Chen et al., 2021d), `PiT-S` (Heo et al., 2021), `PiT-B` (Heo et al., 2021)
4	Stem layer: stack of 3×3 conv, 2D SA, InRes: 256, Pooling at stem, SiLU	`HaloNet-50` (Vaswani et al., 2021a), `LambdaResNet-50` (Bello, 2021b), `BoTNeT-26` (Srinivas et al., 2021a), `GC-ResNeXt-50` (Cao et al., 2019b), `ECAHaloNeXt-50`†, `ECA-BoTNeXt-26`†
5	Stem layer: stack of 3×3 convs, InRes: 256, 2D SA, CWA: middle of blocks, CNN	`LamHaloBoTNet-50`†, `SE-BoTNet-33`†, `SE-HaloNet-33`†, `Halo2BoTNet-50`†, `GC-ResNet-50` (Cao et al., 2019b), `ECA-Net-33` (Wang et al., 2020c)
6	Stem layer: 7×7 conv w/ s2, ReLU, Pooling at stem, CNN, BN	`ResNet-50` (He et al., 2016a), `ResNet-101` (He et al., 2016a), `ResNeXt-50` (Xie et al., 2017a), `Wide ResNet-50` (Zagoruyko & Komodakis, 2016), `SE-ResNet-50` (Hu et al., 2018b), `SE-ResNeXt-50` (Hu et al., 2018b), `ResNet-V2-50` (He et al., 2016b), `ResNet-V2-101` (He et al., 2016b), `ResNet-50 (GN)` (Wu & He, 2018), `ResNet-50 (BlurPool)` (Zhang, 2019), `DPN-107` (Chen et al., 2017), `Xception-65` (Chollet, 2017b)
7	NAS, Stem layer: 3×3 conv w/ s2 CWA: middle of blocks, CWA, DW Conv	`EfficientNet-B2` (Tan & Le, 2019a), `FBNetV3-G` (Dai et al., 2021a), `ReXNet` (×1.5) (Han et al., 2021a), `RegNetY-32` (Radosavovic et al., 2020), `MixNet-XL` (Tan & Le, 2019b), `NF-RegNet-B1` (Brock et al., 2021a)
8	Input resolution: 224, Stem layer: stack of 3×3 convs, Group Conv, Final GAP, 2D SA	`NFNet-L0`†, `ECA-NFNet-L0`†, `PoolFormer-M48` (Yu et al., 2022), `ResNeSt-50` (Zhang et al., 2022a), `ResNet-V2-50-D-EVOS`†, `ConvMixer-1536/20` (Trockman & Kolter, 2022)
9	ReLU, Input resolution: 224, DW Conv, BN 2D self-attention	`ViT-B (patch size 32)` (Dosovitskiy et al., 2021), `R26+ViT-S` (Steiner et al., 2022), `DLA-X-102` (Yu et al., 2018), `eSE-VoVNet-39` (Lee & Park, 2020), `ResNet-101-C` (He et al., 2019), `RegNetX-320` (Radosavovic et al., 2020) `HRNet-W32` (Wang et al., 2020b)
10	ReLU + Leaky ReLU, InRes: 256, Stem layer: 7×7 conv, CNN, Pooling at stem	`CSPResNet-50` (Wang et al., 2020a), `CSPResNeXt-50` (Wang et al., 2020a), `CSPDarkNet-53` (Bochkovskiy et al., 2020), `NF-ResNet-50` (Brock et al., 2021a)

gies, while fixing the model architecture, as follows: **Different initializations** can affect the model training by the nature of the stochasticity of the training procedure. For example, Somepalli et al. (2022) showed that the decision boundary of each architecture can vary by different initializations. We also consider **different optimization hyper-parameters** (*e.g.*, learning rate, weight decay). Finally, we study the effect of **different training regimes** (*e.g.*, augmentations, type of supervision). For example, the choice of data augmentation (*e.g.*, Mixup (Zhang et al., 2018), or CutMix (Yun et al., 2019)) or the optimization techniques (*e.g.*, (Szegedy et al., 2016)) can theoretically or empirically affect adversarial robustness (Zhang et al., 2021; Park et al., 2022; Shafahi et al., 2019; Chun et al., 2019). Similarly, we investigate the effect of different types of supervision, such as self-supervision (*e.g.*, MOCO (Chen et al., 2021c), MAE (He et al., 2022) and BYOL (Grill et al., 2020)) or semi-weakly supervised learning (Yalniz et al., 2019).

Settings. **Different initializations:** We train 21 `ResNet-50` models and 16 `ViT-S` from scratch individually by initializing each network with different random seeds. **Different hyper-parameters:** We train 28 `ResNet-50` models and 9 `ViT-S` by choosing different learning rates, weight decays, and learning rate schedulers. The full list of hyper-parameters is shown in Appendix D. **Different training regimes:** We collect 23 `ResNet-50` models and 7 `ViT-S` models with different training regimes. 22 `ResNet-50` models include 1 model trained by PyTorch (Paszke et al., 2019), 4 models

† Customized models by Wightman (2019): `HaloRegNetZ` = `HaloNet` + `RegNetZ`; `ECA-BoTNeXt` = `ECA-Net` + `HaloNet` + `ResNeXt`; `ECA-BoTNeXt` = `ECA-Net` + `BoTNet` + `ResNeXt`; `LamHaloBoTNet` = `LambdaNet` + `HaloNet` + `BoTNet`; `SE-BoTNet` = `SENet` + `BoTNet`; `SE-HaloNet` = `SENet` + `HaloNet`; `Halo2BoTNet` = `HaloNet` + `BoTNet`; `NFNet-L0` = an efficient `NFNet-F0` (Brock et al., 2021b); `ECA-NFNet-L0` = `ECA-Net` + `NFNet-L0`; `ResNet-V2-D-EVOS` = `ResNet-V2` + `EvoNorms` (Liu et al., 2020).



(a) Similarity between different networks. (b) Similarity within `ResNet-50` and `ViT-S` with different optimization settings.

Figure 2: **Similarity scores between different architectures vs. within the same architecture.** (a) Subsampled pairwise similarity scores of 69 neural architectures. (b) Similarity between 4 `ResNet-50` models and 4 `ViT-S` models with different optimization settings. `Init`, `LR`, and `WD` are randomly chosen from models trained with different settings of initialization, learning rate, and weight decay in Tab. 3, respectively.

trained by GluonCV (Guo et al., 2020), a semi-supervised and semi-weakly supervised models on billion-scale unlabeled images (Yalniz et al., 2019), 5 models trained by different augmentations (Cutout (DeVries & Taylor, 2017), Mixup (Zhang et al., 2018), manifold Mixup (Verma et al., 2019), CutMix (Yun et al., 2019), and feature CutMix); 11 optimized models by Wightman et al. (2021). 7 `ViT-S` models contain the original ViT training setup (Dosovitskiy et al., 2021), a stronger data augmentation setup (Touvron et al., 2021), the training setup with distillation (Touvron et al., 2021), an improved DeiT training setup (Touvron et al., 2022b), and self-supervised training fashions including MoCo v3 (Chen et al., 2021c), MAE (He et al., 2022) and BYOL (Grill et al., 2020).

Tab. 3 shows the comparison of similarity scores between the same architecture but different learning methods (a smaller similarity means more diversity). First, we observe that using different random initialization or different optimization hyper-parameters shows high correlations with each other (almost ≥ 4.2) while the average similarity score between various neural architectures is 2.73. In other words, the difference in initializations or optimization hyper-parameters does not significantly contribute to the model diversity. Second, we observe that using different learning technique (*e.g.*, different augmentation methods, different types of supervision) remarkably affect the similarity score (3.27 for ResNet and 3.44 for ViT), but not significant as much as the architectural difference (2.73). Fig. 2 shows that the similarities between different architectures (Fig. 2a and the off-diagonal elements of Fig. 2b) are smaller than the similarities within the same architecture with different initializations or hyper-parameters. Furthermore, we discover that change of similarity caused by different initializations or hyper-parameters is less marked than the change caused by different architecture.

Table 3: **Similarity within the same architecture.** We compare the average similarity within the same network architecture but trained with different procedures, *e.g.*, different random initializations, different optimization hyper-parameters, and different training regimes. We report the similarities of two neural architectures, `ResNet-50` and `ViT-S`, where we use the DeiT training setting (Touvron et al., 2021) for ViT training. "All" denotes the average similarity of 69 neural architectures in Tab. 2.

Architecture	Diversity by	Similarity
`ResNet-50`	Initialization	4.23
	Hyper-parameter	4.05
	Training regime	3.27
`ViT-S`	Initialization	4.21
	Hyper-parameter	4.22
	Training regime	3.44
All		2.73

In practice, learning the same network with significantly different learning techniques is challenging and time-consuming because the hyper-parameter search space is tremendously large (Bergstra & Bengio, 2012). Especially, a novel optimization technique (*e.g.*, using more data points for semi- or self-supervised learning, using different augmentation methods, or adopting advanced optimization settings) sometimes needs a completely different optimization setting compared to the vanilla setting. On the other hand, learning a new architecture is easier than using a different learning paradigm

in terms of hyper-parameter selection and implementation. Thus, we can conclude that if we need multiple diverse models, then using multiple architectures is more effective and easier to optimize.

4 PRACTICAL APPLICATIONS OF THE NETWORK SIMILARITY SCORE

In this section, we show the practical usage of our similarity score. First, we show that using more diverse models will lead to better ensemble performance (Sec. 4.1). Second, we suggest a similarity-based guideline for the choice of a teacher model when distilling to a specific architecture. Different from several existing studies that show a more similar teacher is better, our findings show that their observation is valid only if the teacher and student networks are based on the same architecture (*e.g.*, Transformer). On the other hand, when the teacher and student networks are based on different architectures, a more dissimilar teacher is better (Sec. 4.2).

4.1 MODEL DIVERSITY AND ENSEMBLE PERFORMANCE

The model ensemble is a practical technique to achieve high performances in practice, however, there is still not enough study on the choice of the models for the ensemble. We report the relationship between the model similarity and the ensemble performance based on the unweighted average method (Ju et al., 2018) (*i.e.*, averaging the logit values of the ensembled models). Because the ensemble performance is sensitive to the original model performances, we define Error Reduction Rate (ERR) as $1 - \frac{\text{Err}_{\text{ens}}(M)}{\frac{1}{|M|}\sum_{m \in M} \text{Err}(m)}$, where M is the set of the ensembled models, ERR(m) denotes the top-1 ImageNet validation error of model m, and $\text{Err}_{\text{ens}}(\cdot)$ denotes the top-1 error of the ensemble.

We first measure the 2-ensemble performances among the 69 neural architectures in Tab. 2 (*i.e.*, the number of ensembles is $\binom{69}{2} = 2346$). We plot the relationship between the model similarity and ERR in Fig. 3. We observe that there exists a strong negative correlation between the model similarity and the ensemble performance (Pearson correlation coefficient -0.31 with p-value 1.74×10^{-54}), *i.e.*, *more diversity leads to better ensemble performance*.

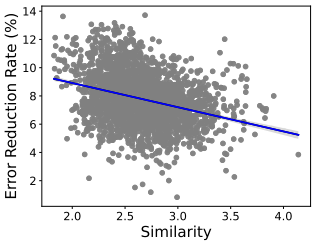
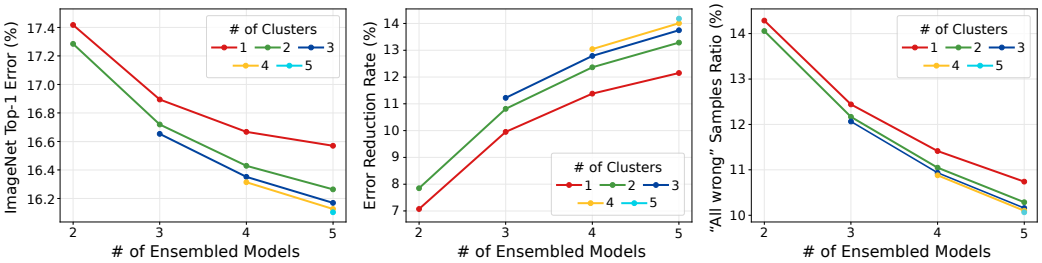


Figure 3: **Correlation between model similarity and ensemble performance.** The trend line and its 90% confidence interval are shown in the blue line.

We also conduct N-ensemble experiments with $N \geq 2$ based on the clustering results in Tab. 2. We evaluate the average ERR of the ensemble of models from k clusters, *i.e.*, if $N = 5$ and $k = 3$, the ensembled models are only sampled from the selected 3 clusters while ignoring the other 7 clusters. We investigate the effect of model diversity and ensemble performance by examining $k = 1 \ldots N$ (*i.e.*, larger k denotes more diverse ensembled models). We report three evaluation metrics in Fig. 4: ImageNet top-1 error of the ensemble (lower is better), error reduction rate (ERR) (higher is better), and the number of wrong samples by all ensembled models (lower is better). In all metrics, we observe that the ensemble of more diverse models (*i.e.*, by increasing the number of clusters for the model selection) shows better ensemble performance. Interestingly, in Fig. 4a, we observe that when

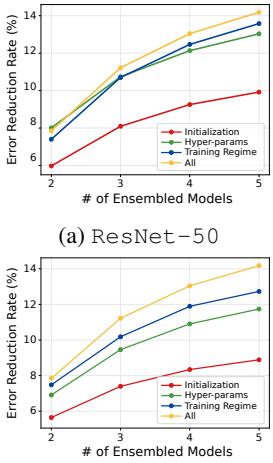


(a) Top-1 Error (b) Error reduction rate (c) "All wrong" samples ratio

Figure 4: **Model diversity by architecture difference and ensemble performance.** We report ensemble performances by varying the number of ensembled models (N) and the diversity of the ensembled models. The diversity is controlled by choosing the models from k different clusters (larger k denotes more diversity).

the number of clusters for the model selection (k) is decreased, the ensemble performance by the number of ensembled models (N) quickly reaches saturation. Similarly, in Fig. 4c, we observe that the number of wrong samples by all models is decreased by selecting more diverse models. In other words, the different architecture choices can lead to different model decision boundaries.

In Sec. 3.2, we showed that the different training strategies are not as effective as different architectures in terms of diversity. To examine the effect of this observation on the ensemble scenario, we report the ensemble results of different training strategies, *i.e.*, the same `ResNet-50` and `ViT-S` models in Tab. 3. For comparison with different architectures, we also report the ensemble of different architectures where all ensembled models are from different clusters (*i.e.*, when $N = k$ in Fig. 4). Fig. 5 shows that although using diverse training regimes (blue lines) improves ensemble performance most among other training techniques (red and green lines), the improvements by using different architectures (yellow lines) are more significant than the improvements by using different training regimes (blue lines) with large gaps.



(a) `ResNet-50`

(b) `ViT-S`

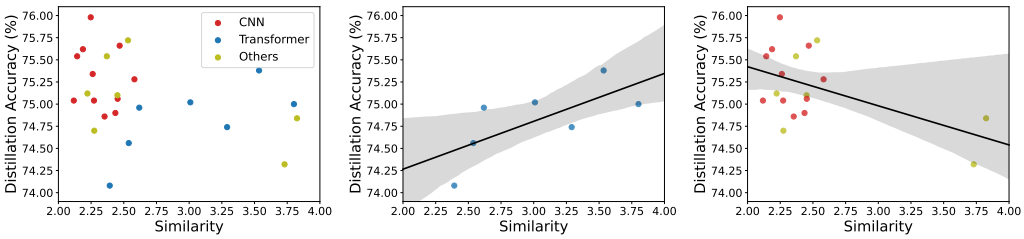
Figure 5: **Model diversity by training techniques and ensemble performance.** The same metrics to Fig. 4 for `ResNet-50` and `ViT-S` models in Tab. 3 are shown. "All" means N=k in Fig. 4 (*i.e.*, # of ensembled models = # of Clusters).

4.2 Model diversity and knowledge distillation

Knowledge distillation (Hinton et al., 2015) is a training method for transferring rich knowledge of a well-trained teacher network. Intuitively, distillation performance affects a lot by choice of the teacher network, however, the relationship between similarity and distillation performance has not yet been explored enough, especially for ViT. In this subsection, we investigate how the similarity between teacher and student networks contributes to the distillation performance. There are several studies showing two contradictory conclusions; Jin et al. (2019); Mirzadeh et al. (2020) showed that a similar teacher leads to better distillation performance; Touvron et al. (2021) reports that distillation from a substantially different teacher is beneficial for ViT training.

Settings. We use the same distillation setting by Touvron et al. (2021); We train `ViT-Ti` models with the hard distillation strategy (Hinton et al., 2015) (*i.e.*, using the predicted label by the teacher network as the ground truth label). We train 25 `ViT-Ti` models with different teacher networks from Tab. 2. We describe the detailed training settings in Appendix E.1.

Fig. 6a illustrates the relationship between the teacher-student similarity and the distillation performance. Here, we exclude the top-2 best and worst results to reduce the effect of outliers. Fig. 6a tends to show a not significant negative correlation between teacher-student similarity and distillation performance (-0.32 Pearson correlation coefficient with 0.12 p-value). However, if we only focus on when the teacher and student networks are based on the same architecture (*i.e.*, Transformer), we can observe a strong positive correlation (Fig. 6b) – 0.70 Pearson correlation coefficient with 0.078 p-value. In this case, our observation is aligned with Jin et al. (2019); Mirzadeh et al. (2020): a



(a) Results on various teachers (b) Transformer teachers (c) CNN & other teachers

Figure 6: **Model diversity and distillation performance.** (a) We show the relationship between teacher-student similarity and distillation performance of 25 `DeiT-S` models distilled by various teacher networks. We show the relationship when the teacher and student networks are based on (b) Transformer and (c) otherwise.

teacher similar to the student improves distillation performance. However, when the teacher and student networks are based on different architectures (*e.g.*, CNN), then we can observe a stronger negative correlation (Fig. 6c) with -0.51 Pearson correlation coefficient and 0.030 p-value. In this case, a more dissimilar teacher leads to better distillation performance. We also test other factors that can affect distillation performance in Appendix E.2; We observe that distillation performance is not correlated to teacher accuracy in our experiments.

Why we observe contradictory results for Transformer teachers (Fig. 6b) and other teachers (Fig. 6c)? Here, we conjecture that when the teacher and student networks differ significantly, distillation works as a strong regularizer. In this case, using a more dissimilar teacher can be considered as a stronger regularizer (Fig. 6c). On the other hand, we conjecture that if two networks are similar, then distillation works as easy-to-follow supervision for the student network. In this case, we can assume that a more similar teacher will work better because a more similar teacher will provide more easy-to-follow supervision for the student network (Fig. 6b). In our experiments, we found that the regularization effect improves distillation performance better than easy-to-follow supervision (*i.e.*, the best performing distillation result is by a CNN teacher). Therefore, in practice, we recommend using a significantly different teacher network for achieving better distillation performance (*e.g.*, using RegNet (Radosavovic et al., 2020) teacher for ViT student as Touvron et al. (2021)).

5 Limitations and discussions

An efficient approximation for a new model. We can use our toolbox for designing a new model; we can measure the similarity between a novel network and existing N architectures; a novel network can be assigned to clusters (Tab. 2) 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 equation 1 by omitting computing the accuracy of the novel network on the adversarial samples of the existing networks. It will break the symmetricity of our score, but it is computationally efficient because we only need to generate adversarial samples for the novel network. As an example, we tested `Gluon-ResNeXt-50` (Guo et al., 2020), which has the same architecture with `ResNeXt-50` (Xie et al., 2017a) in 69 models, and the distilled version of `DeiT-S` model (Touvron et al., 2021). Because the difference is not significant as we observed in Tab. 3 and Fig. 5, 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`. We observe that, 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.

Other possible applications. We can use our similarity to develop a novel neural architecture or training technique by measuring how the proposed method can learn different representations compared to existing methods. Our similarity also can be used for analyzing the hyper-parameter sensitivity. Instead of reporting the performance changes by different hyper-parameters, one can report the average similarity of the trained models with different hyper-parameters. If the average similarity is significantly low, then we can argue that the method is sensitive to hyper-parameter selection. We expect this scenario is especially useful for the task, sensitive to the hyper-parameter selection (Choe et al., 2022; Gulrajani & Lopez-Paz, 2021). Finally, our similarity can be used for the model selection criteria for selecting diverse models, *e.g.*, constructing a new dataset with diverse pre-trained models for annotation candidates (Kuznetsova et al., 2020; Chun et al., 2022).

6 Conclusion

We have explored similarities between image classification models to investigate what makes the model diverse and whether developing and using diverse models are required. For quantitative and model-agnostic assessment of the similarity, we have suggested a new score based on attack transferability demonstrating differences in input gradients. We have shown that macroscopic architectural properties, such as base architecture and stem architecture, have a greater impact on similarity than microscopic operations, such as types of used convolution, with numerical analysis. Furthermore, the effectiveness of the training strategy is minor compared to model architectural design-related features. Finally, we have discovered the advantages of using diverse models in ensemble or distillation strategies, which aim to achieve high performance.

REFERENCES

Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Naman Bansal, Chirag Agarwal, and Anh Nguyen. Sam: The sensitivity of attribution methods to hyperparameters. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *arXiv preprint arXiv:2102.08602*, 2021a.

Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *International Conference on Learning Representations*, 2021b.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. In *International Conference on Learning Representations*, 2021a.

Andy Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *Int. Conf. Mach. Learn.*, 2021b.

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.*, pp. 0–0, 2019a.

Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019b.

Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021a.

Minghao Chen, Kan Wu, Bolin Ni, Houwen Peng, Bei Liu, Jianlong Fu, Hongyang Chao, and Haibin Ling. Searching the search space of vision transformer. *Advances in Neural Information Processing Systems*, 2021b.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021c.

Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. *Advances in Neural Information Processing Systems*, 2017.

Zhengsu Chen, Lingxi Xie, Jianwei Niu, Xuefeng Liu, Longhui Wei, and Qi Tian. Visformer: The vision-friendly transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021d.

Junsuk Choe, Seong Joon Oh, Sanghyuk Chun, and Hyunjung Akata, Zeynepand Shim. Evaluation for weakly supervised object localization: Protocol, metrics, and datasets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017a.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017b.

Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 2021.

Sanghyuk Chun, Seong Joon Oh, Sangdoo Yun, Dongyoon Han, Junsuk Choe, and Youngjoon Yoo. An empirical evaluation on robustness and uncertainty of regularization methods. In *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019.

Sanghyuk Chun, Wonjae Kim, Song Park, Minsuk Chang Chang, and Seong Joon Oh. Eccv caption: Correcting false negatives by collecting machine-and-human-verified image-caption associations for ms-coco. In *Eur. Conf. Comput. Vis.*, 2022.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020.

Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using predictor pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021a.

Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 2021b.

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

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Int. Conf. Mach. Learn.*, pp. 1019–1028. PMLR, 2017.

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. URL https://openreview.net/forum?id=YicbFdNTTy.

Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, 2021.

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

Robert Geirhos, Carlos RM Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. In *Adv. Neural Inform. Process. Syst.*, volume 31, 2018.

Robert Geirhos, Kristof Meding, and Felix A Wichmann. Beyond accuracy: quantifying trial-by-trial behaviour of cnns and humans by measuring error consistency. In *Adv. Neural Inform. Process. Syst.*, volume 33, pp. 13890–13902, 2020.

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. *Advances in neural information processing systems*, 33: 21271–21284, 2020.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *Int. Conf. Learn. Represent.*, 2021.

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. URL http://jmlr.org/papers/v21/19-429.html.

Dongyoon Han, Sangdoo Yun, Byeongho Heo, and YoungJoon Yoo. Rethinking channel dimensions for efficient model design. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021a.

Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 2021b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Eur. Conf. Comput. Vis.*, 2016b.

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.*, pp. 16000–16009, 2022.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 7132–7141, 2018a.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018b.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1345–1354, 2019.

Cheng Ju, Aurélien Bibaut, and Mark van der Laan. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics*, 45(15):2800–2818, 2018.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Int. Conf. Mach. Learn.*, pp. 3519–3529. PMLR, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Adv. Neural Inform. Process. Syst.*, volume 25, pp. 1097–1105, 2012.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *Int. J. Comput. Vis.*, 128(7):1956–1981, 2020.

Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Adv. Neural Inform. Process. Syst.*, volume 31, 2018.

Hanxiao Liu, Andy Brock, Karen Simonyan, and Quoc Le. Evolving normalization-activation layers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, 2020.

Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021a.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021b.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022a.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022b.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5191–5198, 2020.

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

Chanwoo Park, Sangdoo Yun, and Sanghyuk Chun. A unified analysis of mixed sample data augmentation: A loss function perspective. In *Neural Information Processing Systems (NeurIPS)*, 2022.

Namuk Park and Songkuk Kim. How do vision transformers work? In *Int. Conf. Learn. Represent.*, 2022.

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. *Advances in neural information processing systems*, 32, 2019.

Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing network design spaces. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? In *Adv. Neural Inform. Process. Syst.*, volume 34, pp. 12116–12128, 2021.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

Luca Scimeca, Seong Joon Oh, Sanghyuk Chun, Michael Poli, and Sangdoo Yun. Which shortcut cues will dnns choose? a study from the parameter-space perspective. In *Int. Conf. Learn. Represent.*, 2022.

Ali Shafahi, Amin Ghiasi, Furong Huang, and Tom Goldstein. Label smoothing and logit squeezing: a replacement for adversarial training? *arXiv preprint arXiv:1910.11585*, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. Learn. Represent.*, 2015.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR (workshop track)*, 2014.

D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. SmoothGrad: removing noise by adding noise. *ICML workshop on visualization for deep learning*, 2017.

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.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021a.

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.*, pp. 16519–16529, 2021b.

Andreas Peter Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. 2022.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Int. Conf. Mach. Learn.*, pp. 3319–3328, 2017.

AH Sung. Ranking importance of input parameters of neural networks. *Expert Systems with Applications*, 15 (3-4):405–411, 1998.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pp. 2818–2826, 2016.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, 2019a.

Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. In *Brit. Mach. Vis. Conf.*, 2019b.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

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.*, pp. 10347–10357. PMLR, 2021.

Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022a.

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

Asher Trockman and J. Zico Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst.*, 2017.

Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021a.

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.*, pp. 12894–12904, 2021b.

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.*, pp. 6438–6447. PMLR, 2019.

Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020a.

Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3349–3364, 2020b.

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.*, 2020c.

Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017a.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017b.

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.

Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

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.

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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2022a.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Int. Conf. Learn. Represent.*, 2018.

Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? In *Int. Conf. Learn. Represent.*, 2021.

Richard Zhang. Making convolutional networks shift-invariant again. In *Int. Conf. Mach. Learn.*, 2019.

Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, Sercan Ö Arik, and Tomas Pfister. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022b.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *Int. Conf. Learn. Represent.*, 2017.

Table A.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

A Details of networks used for our analysis

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) (Wightman, 2019). Tab. A.1 shows the full list of the models based on their research paper and `timm` alias.

In Tab. A.2 and Tab. A.3, We show the full network specification defined by Tab. 1. We follow the corresponding paper and `timm` library to list the model features.

B 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 algorithm.

We visualize the pairwise distances of the spectral features (*i.e.*, K-largest eigenvectors of L) of 69 architectures in Fig. B.1. Note that rows and columns of Fig. B.1 are sorted by the clustering results. Fig. B.1 shows block diagonal patterns, *i.e.*, in-cluster similarities are large while between-cluster similarities are small.

C Pairwise similarities

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

D Training setting details for Sec. 3.2

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 (Paszke et al.,

Table A.2: 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 $\times$ 3 kernel with stride 2, the second layer having 3 $\times$ 3 kernel with stride 1, and the last layer having 3 $\times$ 3 with stride 1.

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

Table A.3: 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

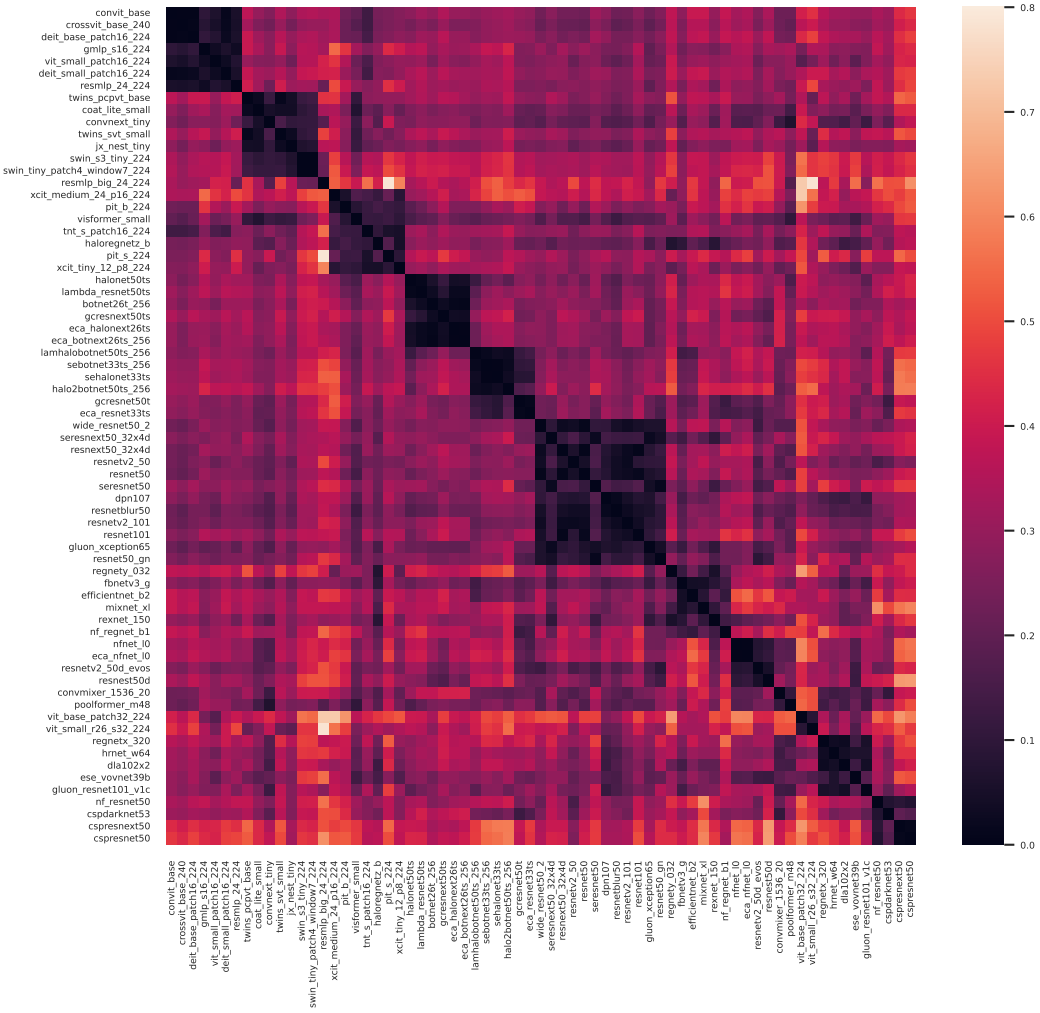


Figure B.1: **Spectral features of 69 architectures.** The K-th largest eigenvectors of the Laplacian matrix of the pairwise similarity graph of 69 architectures are shown ($K = 10$ in this figure). Rows and columns are sorted by the clustering index in Tab. 2. We denote the model name in `timm` for each row and column.

2019); 4 models trained by GluonCV (Guo et al., 2020)[2]; a semi-supervised model and semi-weakly supervised model on billion-scale unlabeled images by Yalniz et al. (2019)[3]; 5 models trained by different augmentation methods (Cutout (DeVries & Taylor, 2017), Mixup (Zhang et al., 2018), manifold Mixup (Verma et al., 2019), CutMix (Yun et al., 2019), and feature CutMix[4]; 10 optimized ResNet models by (Wightman et al., 2021)[5]. We also collect 7 `ViT-S` models with different training regimes, including the original ViT training setup (Dosovitskiy et al., 2021)[6], a stronger data augmentation setup in the Deit paper (Touvron et al., 2021)-3[6], the training setup with distillation (Touvron et al., 2021)-3[6], an improved DeiT training setup (Touvron et al., 2022b)-3[6] , and self-supervised training fashions by MoCo v3 (Chen et al., 2021c)[7], MAE (He et al., 2022)[8] and

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

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

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

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

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

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

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

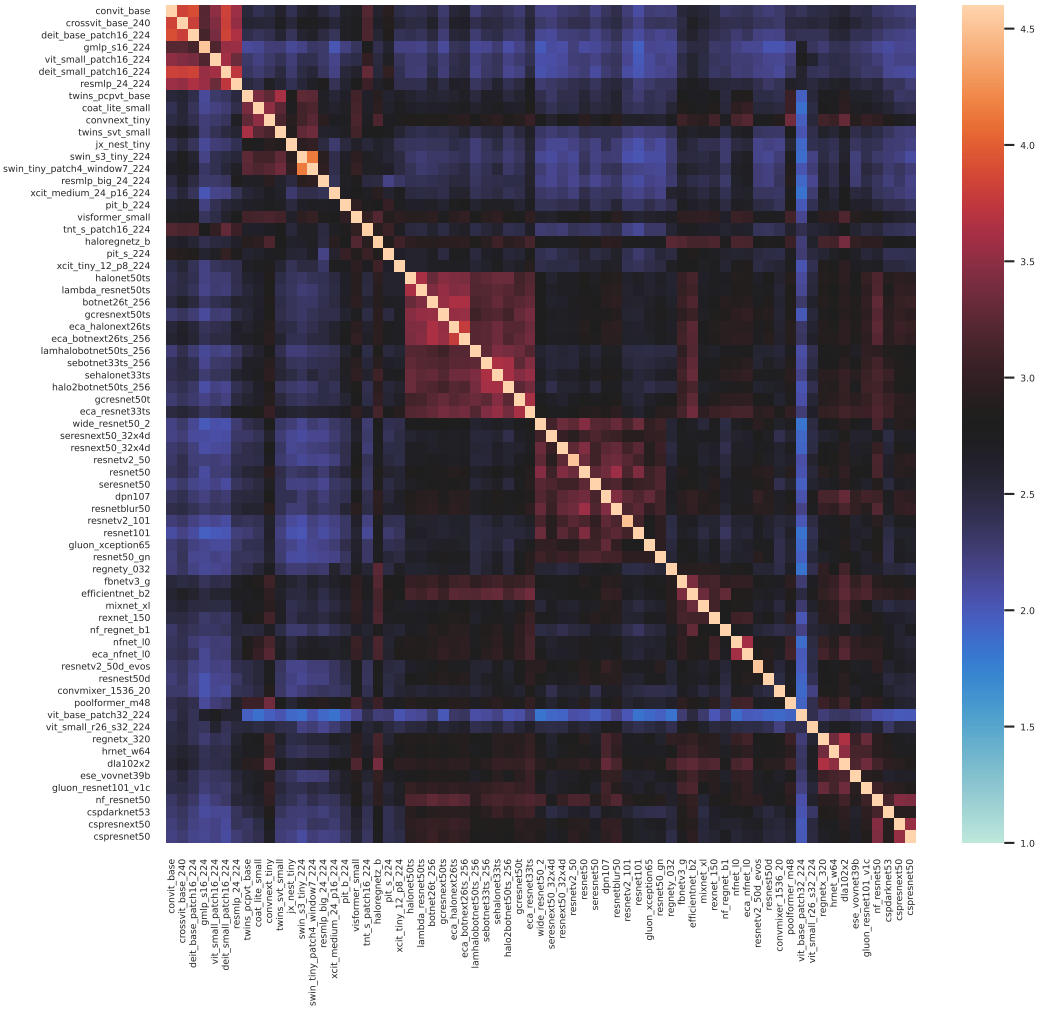


Figure C.1: Pairwise similarity among 69 models. Rows and columns are sorted by the clustering index in Tab. 2. (n, n) component of pairwise similarity is close to 4.6 ($\log 100$) because the attack success rate is almost 100% when attacked model and a model used to generate are the same.

BYOL (Grill et al., 2020)[9]. We do not use adversarially-trained networks because the adversarial training usually drops the standard accuracy significantly (Tsipras et al., 2019).

E KNOWLEDGE DISTILLATION

E.1 TRAINING SETTING DETAILS FOR SEC. 4.2

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[10], only changing the teacher model. Note that we 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`,

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

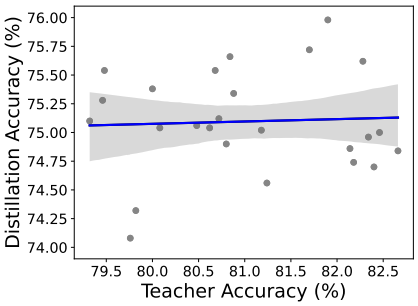


Figure E.1: **Teacher network accuracy and distillation performance.** There is no significant correlation between teacher accuracy and distillation performance.

`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`.

E.2 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 (Yun et al., 2021). In Fig. E.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.