Model Merging via Averaged Representational Similarity

Christopher Wang [*1] **Vighnesh Subramaniam** [*1] **Dan Gutfreund** [2] **Boris Katz** [1] **Phillip Isola** [1] **Brian Cheung** [1]

Abstract

If multiple artists are asked to draw a circle by hand, each one will produce something slightly imperfect. Yet, the average of their sketches can look strikingly close to ideal. We investigate whether knowledge from different models can be combined in the same way. We propose to average models based on their *kernel*: the matrix of all dot products between model embeddings per data sample. Compared to techniques such as weight-averaging, this has the advantage of allowing merging between models that have been trained separately, i.e., from different initializations. We take models that have been trained on disjoint, skewed sets of data and show that simple averaging produces a kernel that trends representationally towards that of a more accurate model. Empirically, we even find the similarity landscape with respect to teacher kernels to be convex. We then use a differentiable version of Mutual k-Nearest Neighbors (MKNN), to directly optimize a student network for representational similarity with the average kernel. We find that this provides consistent gains in performance. These findings open the door for a new type of model-merging that does not rely on weight-averaging, and is thus able to accommodate models that are trained from scratch independently. Going further, they hint at a more general framing for model-merging techniques, in which models can be thought to lie in the same loss basin with respect to their representations.

1. Introduction

Many techniques exist for merging knowledge from a population of models into a single system. Perhaps the simplest way this can be done is by ensembling model predictions at inference time. Prediction ensembling takes the output of many models and produces an ensemble output via majority-voting. Ensembling is reliable and widely used but incurs inference costs that scale linearly with the number of models. Model merging or model souping (White, 2016; Wortsman et al., 2022) avoids this by averaging the model weights together to obtain a singular model. However, because feed-forward networks are symmetric with respect to neuron permutation, weight-averaging requires that models be initialized from the same checkpoint, to ensure that they are aligned in parameter space. More sophisticated approaches try to explicitly find a matching between neurons of different models (Ainsworth et al., 2023), but this makes the restrictive assumption that such a matching exists, and moreover this technique does not work for merging models of different architectures.

We propose a model merging method based on the alignment of model *kernels*, which are the similarity structure of a dataset, given the embeddings produced by a model. More precisely, for a model f and dataset with elements x_i, the kernel is the matrix $\boldsymbol{K}$ of all dot products between model embeddings $\phi_i = f(x_i)$, such that $\boldsymbol{K}_{ij} = \langle \phi_i, \phi_j \rangle$. The similarity structures of different models can be compared using measures such as *Mutual k-Nearest Neighbors* (MKNN) or *Centralized Kernel Alignment* (CKA). Crucially, these comparisons can be made even if there is no alignment from one model's neurons to another.

Our approach is partly inspired by the Platonic Representation Hypothesis (PRH) (Huh et al., 2024), which posits that as models improve, they converge to the same 'platonic' representation $\boldsymbol{K}^*$. In PRH, the authors identify one method for obtaining platonic representations: gathering more data and training higher-quality, more accurate models. We propose an alternative method: ensembling the kernels of many different models. For illustration, imagine asking many people to draw a circle. They will each draw something imperfect, but the circles may be averaged together to smooth out individual mistakes and achieve something closer to the platonic ideal.

In this work, we study how averaging kernels from an ensemble of models – even models trained on different training distributions – can result in a better performance. Our theoretical motivation is the notion that each model can be seen

[1]CSAIL, MIT [2]MIT-IBM Watson AI Lab. Correspondence to: Christopher Wang <czw@mit.edu>, Vighnesh Subramaniam <vsub851@mit.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

as providing a noisy sample of the platonic kernel $\boldsymbol{K}^*$. We simulate this by training an 'oracle' model f^O to high performance, and separately training a population of relatively data-impoverished models. We empirically find that averaging the kernels of the weak models together results in a better kernel, which increases in representational similarity to $\boldsymbol{K}^O$ as N grows. We then go a step further and use differentiable versions of MKNN and CKA to optimize a student model towards the average kernel, which we find results in improved performance. Finally, we make the observation that, empirically, the landscape of similarity to f^O is often convex with respect to weightings between the ensemble's constituent kernels. This hints at a broader framing that could be given for model merging techniques: while prior works such as Git Re-Basin (Ainsworth et al., 2023) show that networks are in the same loss basin modulo permutation, our results support a view that many models could be in the same loss basin with respect to their similarity structures.

Concretely, our findings open the door for a new type of model merging based on optimization for representation similarity with the ensemble average kernel rather than relying solely on weight optimizations. This provides the inference-time savings of model-souping (Wortsman et al., 2022) as well as the knowledge pooling benefits of ensembling. Moreover, we find that these benefits are available, even when the student distribution is poor, i.e., small and heavily imbalanced. This is particularly useful in confidential federated learning settings. For example, separate hospitals may train classifiers, and may be willing to share model weights, but not patient data. Model merging via kernels would allow them to combine model knowledge to produce a superior model.

2. Prior works

Model merging Many of the earliest approaches for model-merging were simple weight-averaging methods (White, 2016; Wortsman et al., 2022; Matena & Raffel, 2022; Yadav et al., 2023). But naive weight-averaging is only possible when models start from the same pre-trained checkpoint, since feed-forward layers are symmetric with respect to neuron permutation. Some approaches attempt to resolve this by finding this permutation via explicit matching (Wang et al., 2020; Ainsworth et al., 2023). This makes the restrictive assumption that such a matching between neurons exists, which limits the representations that a student may learn, especially when a student's architecture differs from a teacher's. In this work, we explore whether knowledge transfer is possible without matching. Another way of doing ensembling is multiple distillation. Our work can be seen as a sub-species of this approach that does not require any alignment in embeddings space, c.f., (Liu et al., 2020).

Guidance Prior works have explored the notion of guiding networks using representational similarity as an auxiliary objective (He & Ozay, 2022; Subramaniam et al., 2025a;b). We extend Subramaniam et al. (2025b), since we focus on knowledge transfer from an *ensemble* of models rather than a single model. Compared to He & Ozay (2022), we offer a complementary explanation of why kernel averaging is effective as a strategy and explain when it fails. We also optimize for alignment based on MKNN, which avoids the pitfalls of raw kernel correlation, i.e., Centralized Kernel Alignment (CKA); e.g., bias from diagonal elements of the kernel. We share similar motivations to (Yu et al., 2025), although we not learn an adapter between models, and instead use representational similarity measures for direct alignment.

3. Ensembling via averaging

Here, we provide a brief theoretical motivation for why it can be expected that ensembling kernels should lead to better performance.

Consider a dataset of M elements. A trained model f gives an embedding for the dataset elements $f(i) = \phi_i$. This induces a distance between elements via the inner-product between model embeddings: $\boldsymbol{K}^f_{i,j} := \langle \phi_i, \phi_j \rangle$. For some chosen k, let $\mathcal{S}^f_k$ be indices of the k elements nearest to x_i under this distance.

Definition: The *mutual k nearest neighbors* distance between two models f and g is defined as the overlap between their nearest neighbor sets under each model's distance function: $m_{nn}(f, g) := \frac{1}{M} \left| \mathcal{S}^f_k \cap \mathcal{S}^g_k \right|$.

Observation: Let $\mathcal{S}^*(i)$ be the nearest neighbors of x_i under the platonic representation $\phi^* = f^*(x_i)$. Assume each model learns a noisy version of the platonic representation, and errors are not correlated between models. For a particular model, let the distance $\boldsymbol{K}_{i,j}$ be a random variable with mean $\boldsymbol{K}^*_{i,j} := \left\langle \phi^*_i, \phi^*_j \right\rangle$. Given N samples $\boldsymbol{K}^k_{i,j} \sim \boldsymbol{K}_{i,j}$, let $\overline{\boldsymbol{K}}_{i,j} = \frac{1}{N} \sum_k \boldsymbol{K}^k_{i,j}$. Let f^* and $\bar{f}$ be models that induce distances $\boldsymbol{K}^*$ and $\overline{\boldsymbol{K}}$ respectively. Then $\lim_{N \to \infty} P\left(m_{nn}(\bar{f}, f^*) = 1\right) = 1$. See Section A.4 for proof.

4. Approach

Given an ensemble of models and a dataset, each model induces neighborhoods over examples in embedding space. We average these induced kernels and train a student whose internal representations reproduce the resulting average neighborhood; see Figure 1.

Formally, given a dataset $\mathcal{D} = \{x_1, ..., x_M\}$ of M elements, and an ensemble of N models $\{f_1, ..., f_N\}$, each model, f_k, induces a distance between dataset elements via the dot product between model embeddings: $\boldsymbol{K}^{f_k,l}_{ij} = \langle \phi^{k,l}_i, \phi^{k,l}_j \rangle$,

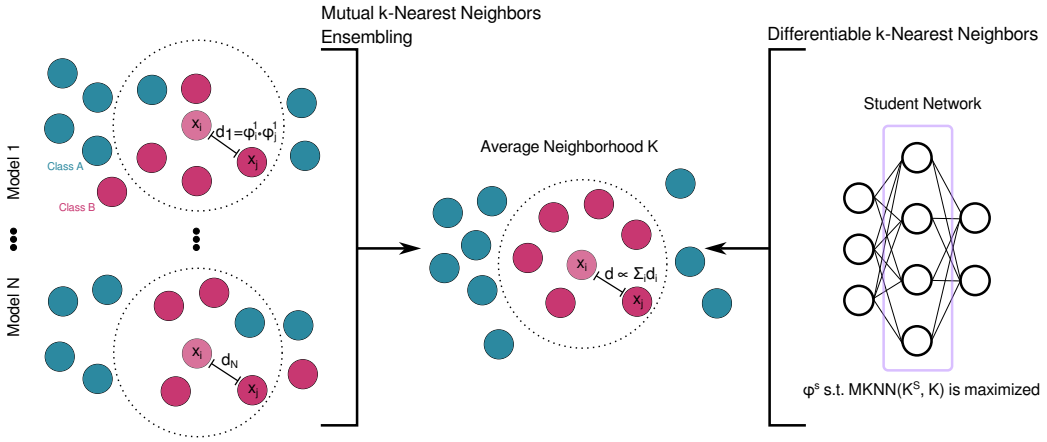

Figure 1. **Schematic overview of our approach**: (*Left*) Given an ensemble of models, each model gives rise to a neighborhood around every dataset element x_i. For individual models, neighborhoods may be imperfect; that is, the closest elements to x_i may be semantically incorrect. (*Center*) However, averaging kernels across models can smooth out these mistakes and produce distances that give rise to more correct neighborhoods. (*Right*) We can then use measures such as Differentiable k-Nearest Neighbors (DiffKNN) to train student models such that their representations ϕ are optimized towards producing this average neighborhood.

where $\phi_i^{k,l} = f_k(x_i)_l$ is the output of f_k at layer l. Elsewhere, our notation will omit the index by layer l, and it can be assumed that results are averaged across layers. Under the distance defined by this dot product, let $\mathcal{S}_n^k(i)$ be the indices of the n closest elements to ϕ_i. We average distances across models and obtain an average neighborhood.

Guidance To optimize for representational similarity with the ensemble kernel, we build on the work of Subramaniam et al. (2025b). During a forward pass, we obtain the final logits as well as the intermediate outputs of each layer of the network. The final logits are used to compute the task loss, and the intermediate outputs are concatenated per layer, and tuned to match the average layer kernel of the teachers. We train student models using representational similarity metrics, to optimize towards the average kernel. Namely, we use a differentiable version of MKNN (DiffKNN) (Subramaniam et al., 2025a), as well as Centralized Kernal Alignment (CKA) (Kornblith et al., 2019), and Unbiased Centralized Kernal Alignment (UCKA) (Song et al., 2007). Below, we give the objective function according to DiffKNN.

DiffKNN Briefly, for f_a and f_b, the differentiable mutual k nearest neighbors (DiffKNN) is given as: $\mathcal{L}_{\text{MKNN}} = \frac{1}{M}\sum_{i=1}^{M} D_{KL}\Big(P_{i\setminus *} \Big\| Q_{i\setminus *}\Big)$ where $P_{ij} = \frac{\exp(\boldsymbol{K}_{ij}^{f_a})\,\mathbf{1}\{j \in \mathcal{S}_k^{f_a}(i)\}}{\sum_{j' \in \mathcal{S}_k^{f_a}(i)} \exp(\boldsymbol{K}_{ij'}^{f_a})}$ and Q_{ij} is defined similarly for f_b. See Subramaniam et al. (2025b) or Appendix A.2 for full description.

Training In the case of any given measure, $\mathcal{L}_{CKA}$, $\mathcal{L}_{UCKA}$, $\mathcal{L}_{MKNN}$, we average the representational similarity across layers and teachers to obtain $\mathcal{L}_{\text{sim}}$. We train jointly on the task objective and the similarity objective, which are computed by batch. The final objective is then $\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{sim}$ During training, we take the log of the CKA and UCKA loss so that it has the same range as the cross-entropy component.

5. Methodology

Experiments We evaluate two questions. First, does averaging kernels from many weak teachers produce a kernel closer to that of a strong oracle? Second, can a student trained to match this average kernel achieve better task performance? To test this, we train impoverished teachers on small or skewed data subsets, compare their averaged kernels to an oracle trained on substantially more data, and then train students with and without representational guidance.

Data For each setting, we run two groups of experiments. A small-scale experiment is run on CIFAR10 (Krizhevsky et al., 2009), for which we train ResNet20 models (He et al., 2016), as implemented by Idelbayev. We also run experiments in which we train ResNet18 models (He et al., 2016) on a subset of ImageNet (Krizhevsky et al., 2012), with 100 classes selected according to Van Gansbeke et al. (2020). For the purpose of the experiment, we take the test set from the ImageNet validation set and the CIFAR10 test set. The test set is common to all splits. The validation set is different per split, and follows the distribution of the train set. Teacher kernels are taken from the checkpoint that achieves the best validation loss.

6. Results

Averaging kernels In short, we find that kernels can be averaged to obtain closer representational similarity to stronger models. We observe that as we average together more of these kernels, we obtain a kernel that is closer to the oracle's

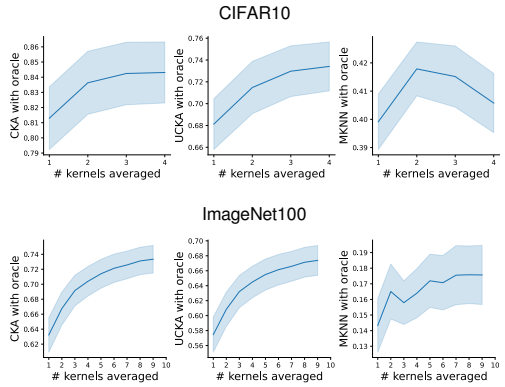

Figure 2. **Averaging kernels improves representational similarity with a more accurate model.** Each specialist model has been trained on a disjoint, skewed dataset. The oracle has been trained on the full dataset, with a balanced distribution. As more teachers are added to the ensemble, the similarity to the oracle increases, as measured by MKNN, CKA and UCKA. Metrics are computed from among 1024 examples. Shading shows standard error.

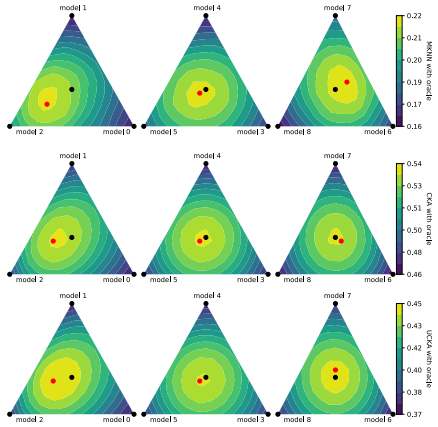

Figure 3. **Similarity landscape appears convex with respect to constituent kernels**. Each vertex of each triangle represents a specialist model's kernel. Each interior point of a triangle shows the oracle similarity with the kernel of a convex weighted average of constituent kernels. The weighting between three teacher models that yields the highest similarity with the oracle (red) is close to the barycenter (black), and the landscape appears highly convex. Therefore, this is similar with findings in weight averaging.

kernel, as measured by CKA, UCKA, and MKNN (see Figure 2). However, the incremental value of each kernel does diminish as the number of kernels in the ensemble increases.

Moreover, we observe something stronger. It appears that we can take a weighted average among kernels and observe that a uniform weighting produces the optimal kernel (see Figure 3). Precisely, for convex weights, $\sum_i w_i = 1$, we measure the similarity between the oracle kernel and $\sum_i w_i \cdot \boldsymbol{K}_i$, and we find that the landscape of this similarity appears convex.

Student training Moreover, we can train a student model to jointly optimize for a classification objective and representational similarity with an ensemble of models. In short, we find that training with guidance from an ensemble of

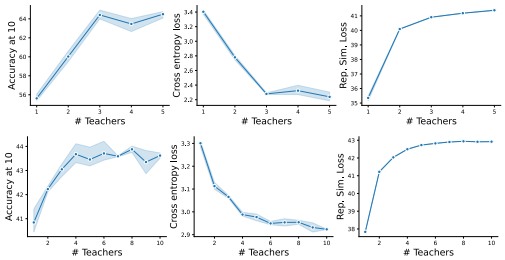

Figure 4. **Optimizing for representational similarity with larger ensembles results in better performance.** Ensembling more teachers leads to better student accuracy (Left) and better performance on the task loss (Center). Shading shows standard error ($n = 3$). Here, teachers consist of models that were trained on the specialist distributions of data. The student was trained on a single specialist distribution of data. (Right) As the student is guided by a larger ensemble, it becomes more difficult to fit the representational similarity of all the teachers at once. This is shown both for CIFAR10 (top) and ImageNet100 (bottom).

teachers results in stronger test performance. As teachers are added to the ensemble, we find improvement on the task accuracy and loss to be roughly monotonic (see Figure 4). See supplementary Table 1 1 for complete results. However, as was the case with measuring kernel similarity, the marginal value of each additional teacher diminishes as the ensemble size increases. Crucially, it should be noted that these improvements occur even when the student data is impoverished or skewed.

7. Conclusion

Broadly, our findings suggest a broader framing to the story behind model soup (Wortsman et al., 2022), and Git Rebasin (Ainsworth et al., 2023). Whereas the existence of model soups suggests that fine-tuned weights of the same checkpoint may all exist in the same loss basin, Git Re-Basin shows that many models, even models trained from different initializations, may exist in the same basin, modulo permutation of neurons. We extend this. For models that have been trained on different data distributions, our findings hint that these models can be considered to be in the same basin, modulo representation. In future investigations, it can be explored whether this holds for models trained on different tasks or different architectures, or both. We could also analyze even bolder settings, such as investigating models that operate with very different data, perhaps different modalities.

8. Acknowledgments

This work was supported by the DARPA Mathematics for the DIscovery of ALgorithms and Architectures (DIAL) program, the DARPA Knowledge Management at Scale and Speed (KMASS) program, the NSF award 2124052, the MIT-IBM Watson AI Lab, the Department of the Air Force Artificial Intelligence Accelerator under Cooperative Agreement Number FA8750-19-2-1000, and the Air Force Office of Scientific Research (AFOSR) under award number FA9550-21-1-0399. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

Ainsworth, S., Hayase, J., and Srinivasa, S. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*, 2023.

Alshammari, S. N., Hershey, J. R., Feldmann, A., Freeman, W. T., and Hamilton, M. I-con: A unifying framework for representation learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Cortes, C., Mohri, M., and Rostamizadeh, A. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828, 2012.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. On kernel-target alignment. *Advances in neural information processing systems*, 14, 2001.

He, B. and Ozay, M. Feature kernel distillation. In *International Conference on Learning Representations*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huh, M., Cheung, B., Wang, T., and Isola, P. The platonic representation hypothesis. *ICML*, 2024.

Idelbayev, Y. Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch. https://github.com/akamaster/pytorch_resnet_cifar10. Accessed: 2026-01-15.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Krizhevsky, A. et al. Learning multiple layers of features from tiny images. 2009.

Liu, Y., Zhang, W., and Wang, J. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415: 106–113, 2020.

Matena, M. S. and Raffel, C. A. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Song, L., Smola, A., Gretton, A., Borgwardt, K. M., and Bedo, J. Supervised feature selection via dependence estimation. In *Proceedings of the 24th international conference on Machine learning*, pp. 823–830, 2007.

Subramaniam, V., Conwell, C., Katz, B., Barbu, A., and Cheung, B. Network of theseus (like the ship). *arXiv preprint arXiv:2512.04198*, 2025a.

Subramaniam, V., Mayo, D., Conwell, C., Poggio, T. A., Katz, B., Cheung, B., and Barbu, A. Training the untrainable: Introducing inductive bias via representational alignment. *NeurIPS*, 2025b.

Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., and Van Gool, L. Scan: Learning to classify images without labels. In *European conference on computer vision*, pp. 268–285. Springer, 2020.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020.

White, T. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

Yu, S., Kwak, S., Jang, H., Jeong, J., Huang, J., Shin, J., and Xie, S. Representation alignment for generation:

Training diffusion transformers is easier than you think. In *The Thirteenth International Conference on Learning Representations*, 2025.

A. Appendix

A.1. Additional Figures and Tables

Specialist Training

Training	CIFAR10		ImageNet100	
	Acc@1	Acc@5	Acc@1	Acc@5
Baseline	55.68 ± 0.68	94.37 ± 0.11	39.92 ± 0.02	65.99 ± 0.22
DiffKNN	$\mathbf{64.49 \pm 0.21}$	$\mathbf{96.44 \pm 0.05}$	$\mathbf{43.62 \pm 0.11}$	$\mathbf{70.00 \pm 0.31}$
CKA	60.35 ± 0.35	95.66 ± 0.05	40.27 ± 0.20	66.44 ± 0.12
UCKA	59.51 ± 0.64	95.24 ± 0.12	39.86 ± 0.17	66.47 ± 0.10
Prediction Ensemble	67.19 ± 2.99	-	45.83 ± 4.17	-
Oracle	91.50 ± 0.09	99.67 ± 0.02	84.08 ± 0.10	95.61 ± 0.03

Balanced Training

Training	CIFAR10		ImageNet100	
	Acc@1	Acc@5	Acc@1	Acc@5
Baseline	81.52 ± 0.53	98.78 ± 0.08	52.98 ± 0.26	76.95 ± 0.32
DiffKNN	$\mathbf{82.91 \pm 0.01}$	98.92 ± 0.17	$\mathbf{56.51 \pm 0.24}$	$\mathbf{80.43 \pm 0.08}$
CKA	82.62 ± 0.11	$\mathbf{98.98 \pm 0.08}$	53.16 ± 0.32	77.97 ± 0.20
UCKA	81.99 ± 0.21	98.93 ± 0.06	54.00 ± 0.20	78.55 ± 0.21
Prediction Ensemble	90.62 ± 1.80	-	58.33 ± 4.17	-
Oracle	91.50 ± 0.09	99.67 ± 0.02	84.08 ± 0.10	95.61 ± 0.03

Table 1. **Test performance of students** Student test performance on specialist (top) and balanced (bottom) datasets, reported as accuracy on the top 1 and top 5 predicted classes. The baseline (first row) is trained on a single data subset and not optimized for representational similarity with any teachers. For DiffKNN, CKA, and UCKA, performance is reported for a student trained to optimize for representational similarity with all teachers, $n = 10$ for ImageNet100 and $n = 5$ for CIFAR10. Accuracies are reported for the final model checkpoint. The similarity measure that results in the best performance is shown in bold. For comparison, we also report the performance of a prediction ensembling approach; i.e., running each model and selecting an output based on majority vote. Despite requiring $N\times$ less parameters at inference time, the single student is still able to achieve competitive performance with prediction ensembling. We also report performance for the oracle, which is trained on the full, balanced distribution. Error is standard error among seeds ($n = 3$).

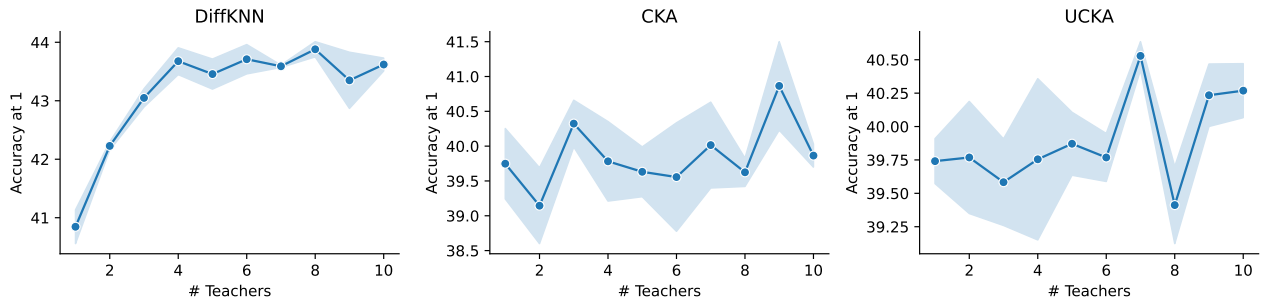

Figure 5. **Accuracy as a function of ensemble size for specialist ImageNet100 teachers** Same results as Figure 4, but for teachers with specialist distributions on ImageNET100. Bands show standard error across seeds ($n = 3$).

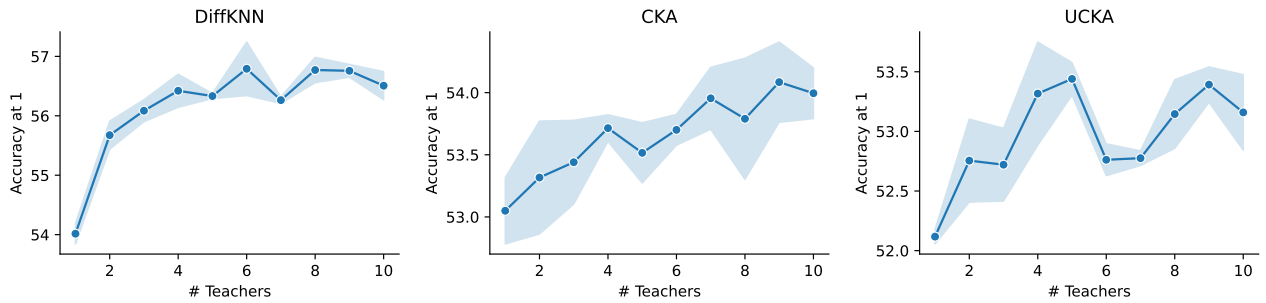

Figure 6. **Accuracy as a function of ensemble size for balanced ImageNet100 teachers** Same results as Figure 4, but for teachers with balanced distributions on ImageNET100. Bands show standard error across seeds ($n = 3$).

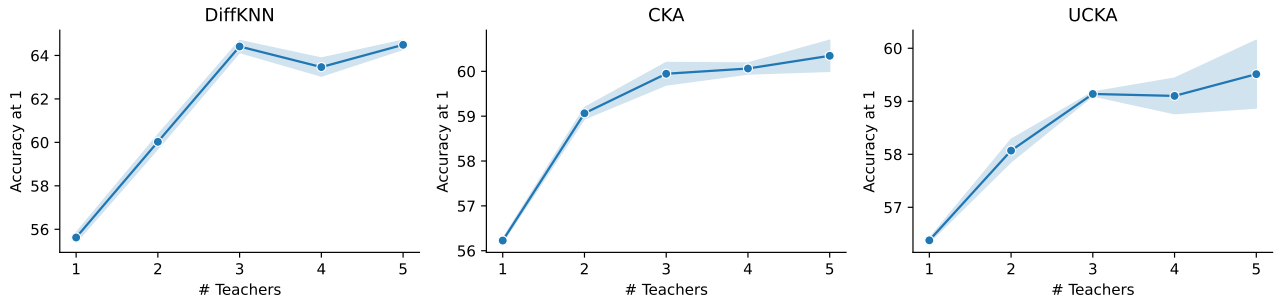

Figure 7. **Accuracy as a function of ensemble size for specialist CIFAR10 teachers** Same results as Figure 4, but for teachers with specialist distributions on CIFAR10. Bands show standard error across seeds ($n = 3$).

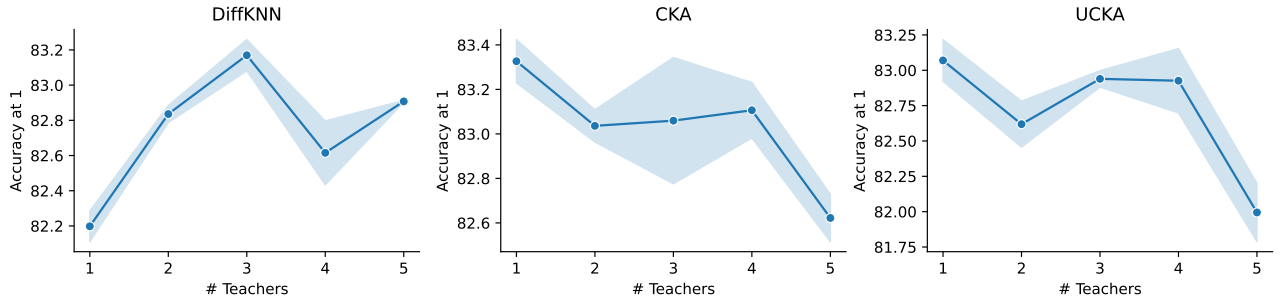

Figure 8. **Accuracy as a function of ensemble size for balanced CIFAR10 teachers** Same results as Figure 4, but for teachers with balanced distributions on CIFAR10. Bands show standard error across seeds ($n = 3$).

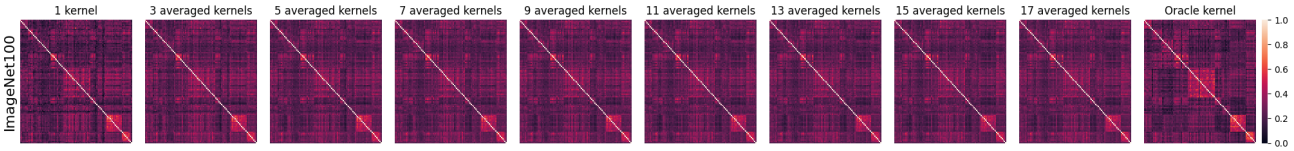

Figure 9. **The progression of averaging all 10 specialist teachers.** Teachers were trained on ImageNet100.

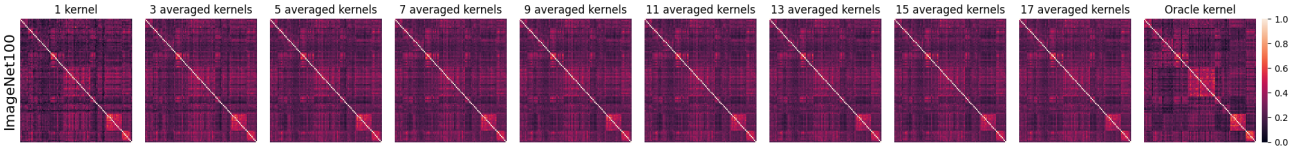

Figure 10. **The progression of averaging all 10 balanced teachers.** Teachers were trained on ImageNet100.

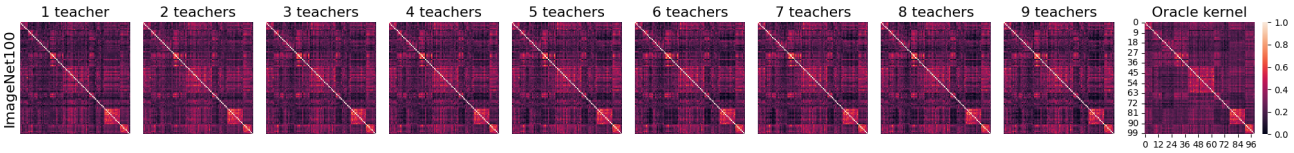

Figure 11. **The progression of a student kernel, progressively trained using all 10 specialist teachers.** Teachers were trained on ImageNet100.

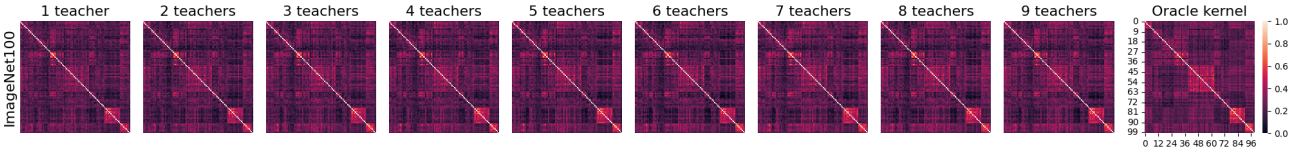

Figure 12. **The progression of a student kernel, progressively trained using all 10 balanced teachers.** Teachers were trained on ImageNet100.

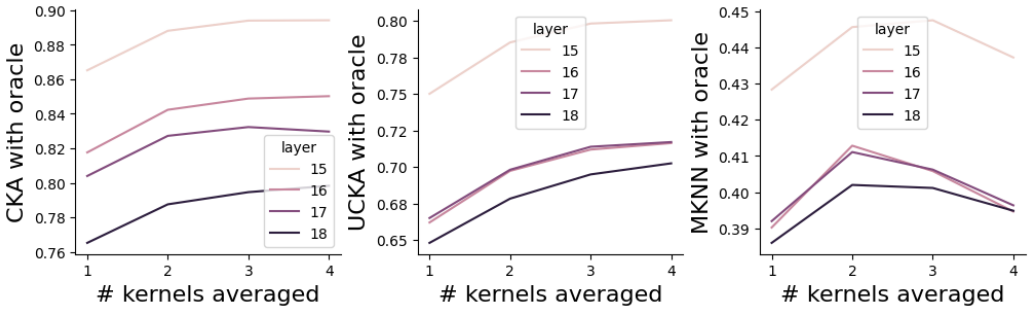

Figure 13. **CIFAR10 similarity trends for specialist teachers.** Trends shown for last 4 network layers.

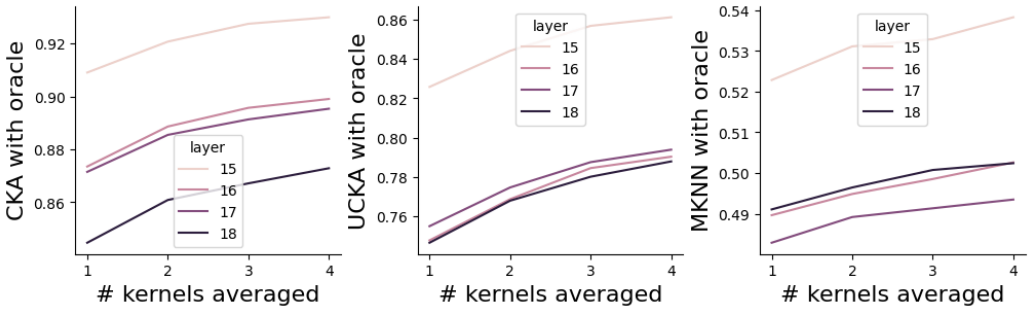

Figure 14. **CIFAR10 similarity trends for balanced teachers.** Trends shown for last 4 network layers.

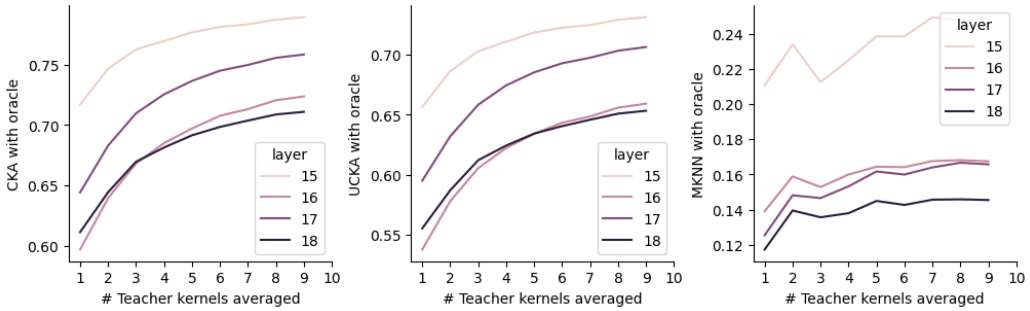

Figure 15. **ImageNet100 similarity trends for specialist teachers.** Trends shown for last 4 network layers.

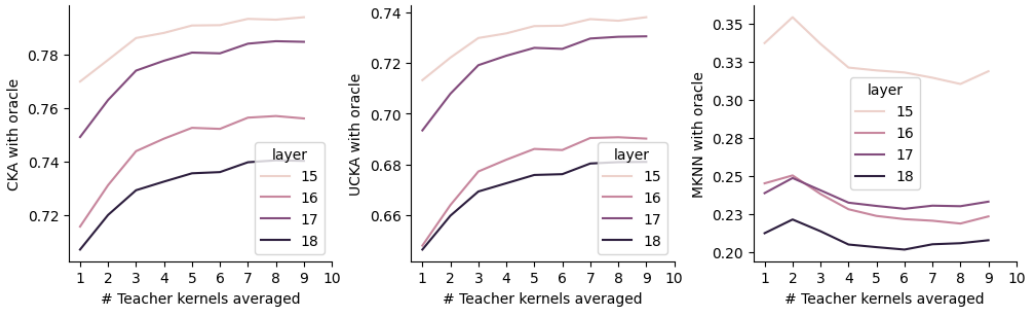

Figure 16. **ImageNet100 similarity trends for balanced teachers.** Trends shown for last 4 network layers.

A.2. Differentiable Mutual Nearest Neighbors

We adapt the appendices of (Subramaniam et al., 2025a) and present them here for ease of reference.

For Differentiable MNN (D-MNN), we use Huh et al. (2024) and Alshammari et al. (2025) to reformulate nearest neighbors into a differentiable form.

Let b be the mini-batch size and anchors $i \in \{1, \ldots, b\}$. Two encoders f_1, f_2 produce features $\phi_i = f_1(x_i) \in \mathbb{R}^d$, $\psi_i = f_2(y_i) \in \mathbb{R}^d$. Define $\mathcal{J}_i = \{1, \ldots, b\} \setminus \{i\}$. For $k \in \mathbb{N}$, let $s_{f_1}(i), s_{f_2}(i) \subset \mathcal{J}_i$ be the index sets of the k nearest neighbors of i under $\{\phi_j\}$ and, $\{\psi_j\}$, respectively.

k-Nearest Neighbors: As defined by Huh et al. (2024), the overlap score for anchor i is

$$g_{NN}(\phi_i, \psi_i) = \frac{|s_{f_1}(i), s_{f_2}(i)|}{k} \tag{1}$$

and the batch score is the average over i.

From overlap to probabilities: We introduce hard conditional distributions uniform on the kNN sets:

$$p_{f_1}(j \mid i) = \frac{\mathbf{1}\{j \in s_{f_1}(i)\}}{k}, \qquad q_{f_2}(j \mid i) = \frac{\mathbf{1}\{j \in s_{f_2}(i)\}}{k}, \quad j \in \mathcal{J}_i. \tag{2}$$

Then

$$\sum_{j \in \mathcal{J}_i} p_{f_1}(j \mid i)\, q_{f_2}(j \mid i) = \frac{|s_{f_1}(i) \cap s_{f_2}(i)|}{k^2}, \quad \Rightarrow \quad g_{NN}(\phi_i, \psi_i) = k \sum_{j \in \mathcal{J}_i} p_{f_1}(j \mid i)\, q_{f_2}(j \mid i). \tag{3}$$

Differentiable conditional neighborhoods: Let $\tau > 0$ be a temperature. Define pairwise similarities either by

$$s^f_{ij} = -\frac{\|\phi_i - \phi_j\|_2^2}{\tau} \quad \text{or} \quad s^f_{ij} = \frac{\phi_i^\top \phi_j}{\tau}, \qquad j \in \mathcal{J}_i, \tag{4}$$

and analogously s^g_{ij} from $\{\psi_j\}$. Let $T^f_k(i) \subset \mathcal{J}_i$ and $T^g_k(i) \subset \mathcal{J}_i$ denote top-k index selections under s^f_{ij} and s^g_{ij}, respectively. We define masked, normalized conditionals over the top-k:

$$P_{ij} \equiv p_{f_1}(j \mid i) = \frac{\exp(s^{f_1}_{ij})\, \mathbf{1}\{j \in T^{f_1}_k(i)\}}{\sum_{j' \in T^{f_1}_k(i)} \exp(s^{f_1}_{ij'})}, \qquad Q_{ij} \equiv q_{f_2}(j \mid i) = \frac{\exp(s^{f_2}_{ij})\, \mathbf{1}\{j \in T^{f_2}_k(i)\}}{\sum_{j' \in T^{f_2}_k(i)} \exp(s^{f_2}_{ij'})}. \tag{5}$$

(Any differentiable soft-top-k operator may be used; with an exact mask and $\tau \to 0$, P_{ij} and Q_{ij} converge to the hard uniforms.)

D-MNN alignment and limit: We define the differentiable alignment for anchor i as the inner product

$$g_{\text{soft}}(\phi_i, \psi_i) \;=\; \sum_{j \in \mathcal{J}_i} P_{ij}\, Q_{ij}, \qquad \bar{g}_{\text{soft}} \;=\; \frac{1}{b} \sum_{i=1}^{b} m_{\text{soft}}(\phi_i, \psi_i). \tag{6}$$

Under the exact top-k mask and $\tau \to 0$, $P_{ij}, Q_{ij} \to \frac{1}{k}\mathbf{1}\{j \in s_{f_1}(i)\}, \frac{1}{k}\mathbf{1}\{j \in s_{f_2}(i)\}$, so that $k\, g_{\text{soft}}(\phi_i, \psi_i) \to g_{NN}(\phi_i, \psi_i)$ by equation 3.

Training objective: Excluding the anchor, we write $P_{i\setminus *} = \{P_{ij}\}_{j \in \mathcal{J}_i}$ and similarly for Q. When P (from f) guides Q (from g), we minimize

$$\Delta_{\text{D-MNN}} \;:=\; \mathcal{L}_{KL} \;=\; \frac{1}{b} \sum_{i=1}^{b} D_{KL}\Big(P_{i\setminus *} \,\Big\|\, Q_{i\setminus *}\Big). \tag{7}$$

This KL aligns conditional neighborhood distributions. Most importantly, before computing g, rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ are ℓ_2-normalized. For our setting in this paper, we have to tune the temperature for D-MNN.

A.3. Centered Kernel Alignment

To compare representations, we use a representation similarity metric, $\mathcal{M}$, which corresponds to centered kernel alignment (CKA) (Kornblith et al., 2019; Cortes et al., 2012; Cristianini et al., 2001) in our setting. We specifically consider linear CKA.

CKA uses kernel functions on mean-centered representations to compute representational similarity matrices, which are then compared via the Hilbert-Schmidt Independence Criterion (HSIC). More specifically, suppose we have two sets of representations $\boldsymbol{A} \in \mathbb{R}^{b \times d_1}$ and $\boldsymbol{B} \in \mathbb{R}^{b \times d_2}$. We first compute the Gram matrices for each set of representations

$$\boldsymbol{K} = \boldsymbol{A}\boldsymbol{A}^T, \boldsymbol{L} = \boldsymbol{B}'\boldsymbol{B}'^T \tag{8}$$

We center the Gram matrices by introducing a matrix, H, where $H = \boldsymbol{I}_b - \frac{1}{b}\mathbf{1}\mathbf{1}^T$.

$$\tilde{\boldsymbol{K}} = \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}, \tilde{\boldsymbol{L}} = \boldsymbol{H}\boldsymbol{L}\boldsymbol{H} \tag{9}$$

We compute the HSIC on the Gram matrices.

$$HSIC(\boldsymbol{K}, \boldsymbol{L}) = \text{tr}(\tilde{\boldsymbol{K}}, \tilde{\boldsymbol{L}}) \tag{10}$$

Finally, we define our linear CKA metric as:

$$g_{\text{CKA}}(\boldsymbol{A}, \boldsymbol{B}') := \text{CKA}(\boldsymbol{K}, \boldsymbol{L}) = \frac{HSIC(\boldsymbol{K}, \boldsymbol{L})}{\sqrt{HSIC(\boldsymbol{K}, \boldsymbol{K}) * HSIC(\boldsymbol{L}, \boldsymbol{L})}} \tag{11}$$

In our setting, we consider representational *dissimilarity* and aim to minimize the dissimilarity between representations from our target network and guide network. We define this as:

$$\Delta_{\text{CKA}}(\boldsymbol{A}, \boldsymbol{B}) = 1 - g(\boldsymbol{A}, \boldsymbol{B}) \tag{12}$$

Linear CKA ranges from 0 to 1 (very different representations). Because of this, we take the complement by subtracting the linear CKA from 1 to represent dissimilarity. Most importantly, before computing g, rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ are ℓ_2-normalized.

A.3.1. Unbiased CKA

We cover the unbiased HSIC (Song et al., 2007) here.

We start with our Gram matrices, $\boldsymbol{K}, \boldsymbol{L}$. An unbiased estimator for HSIC then involves forming zero-diagonal copies of both matrices.

$$\tilde{\boldsymbol{K}} \coloneqq \boldsymbol{K} - \text{diag}(\boldsymbol{K}), \qquad \tilde{\boldsymbol{L}} \coloneqq \boldsymbol{L} - \text{diag}(\boldsymbol{L}), \tag{13}$$

so that $\tilde{\boldsymbol{K}}_{ii} = \tilde{\boldsymbol{L}}_{ii} = 0$ and $\tilde{\boldsymbol{K}}_{ij} = \boldsymbol{K}_{ij}$, $\tilde{\boldsymbol{L}}_{ij} = L_{ij}$ for $i \neq j$. Let $\mathbf{1} \in \mathbb{R}^m$ be the all-ones vector. The unbiased estimator of HSIC is

$$HSIC_U(K, L) = \frac{1}{m(m-3)} \left[\text{tr}(\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}) + \frac{\text{tr}(\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}})}{(m-1)(m-2)} - \frac{2}{m-2}\ \text{tr}(\tilde{\boldsymbol{K}}\tilde{\boldsymbol{L}}) \right], \tag{14}$$

which is well-defined for $m \geq 4$. The last term corresponds to the unbiased correction.

A.4. Proof of observation 1

For ease of discussion, consider a single element of the dataset x_i. Let $\boldsymbol{K}_{min} := \min_{j \neq k}(\boldsymbol{K}^*_{i,j} - \boldsymbol{K}^*_{i,k})$ be the smallest gap between distances to x_i as given by the platonic representations. Let $r_{\boldsymbol{K}}(j)$ be the ranking of element x_j among the neighbors of x_i, ordered by distance, so that $r_{\boldsymbol{K}}(j) < r_{\boldsymbol{K}}(k)$ iff $\boldsymbol{K}_{i,j} < \boldsymbol{K}_{i,k}$. By the law of large numbers, $\lim_{N\to\infty} P(|\boldsymbol{K}_{i,j} - \overline{\boldsymbol{K}}_{i,j}| < \boldsymbol{K}_{min}) = 1$. Then, by the union bound, $\lim_{N\to\infty} P(r_{\overline{\boldsymbol{K}}}(i) = r_{\boldsymbol{K}}(i)) = 1$, which implies that the neighborhoods will be equal. □

A.5. Specialist Datasets

Classes in CIFAR10 specialist splits

Split	Classes
1	airplane, automobile
2	bird, cat
3	deer, dog
4	frog, horse
5	ship, truck

Classes in ImageNet100 specialist splits

Split	Classes
1	cougar, giant schnauzer, dalmatian, Staffordshire bullterrier, coyote, grey fox, schipperke, guenon, leopard, Norfolk terrier
2	toy terrier, cairn, proboscis monkey, guinea pig, Scotch terrier, Doberman, collie, Border collie, flat-coated retriever, Arctic fox
3	American black bear, Shetland sheepdog, bluetick, wood rabbit, Irish setter, Border terrier, tusker, trimaran, wreck, bicycle-built-for-two
4	recreational vehicle, unicycle, table lamp, file, banana, rapeseed, organ, upright, electric guitar, oboe
5	hatchet, plane, lawn mower, robin, water ouzel, goose, American coot, oystercatcher, albatross, box turtle
6	sea snake, diamondback, sidewinder, wing, hand blower, microphone, analog clock, computer keyboard, thresher, printer
7	vending machine, neck brace, scorpion, ringlet, washer, steel arch bridge, stupa, planetarium, obelisk, totem pole
8	shoe shop, tobacco shop, dam, picket fence, honeycomb, mashed potato, shower curtain, ashcan, manhole cover, bagel
9	knee pad, crash helmet, pop bottle, trifle, guacamole, bucket, broom, mitten, envelope, cowboy hat
10	gown, maze, pickelhaube, Windsor tie, purse, sleeping bag, stretcher, maillot, stole, dumbbell

Table 2. Classes in specialist splits