Deceiving the CKA Similarity Measure in Deep Learning

Stefan Horoi [2,3] * **MohammadReza Davari** [1,3] * **Amine Natik** [2,3]
Guillaume Lajoie [2,3] **Guy Wolf** [2,3] † **Eugene Belilovsky** [1,3] †

[1] Concordia University [2] Université de Montréal [3] Mila – Quebec AI Institute
{mohammadreza.davari, eugene.belilovsky}@concordia.ca
{stefan.horoi, amine.natik, guillaume.lajoie, guy.wolf}@umontreal.ca

Abstract

Understanding the behaviour of trained deep neural networks is a critical step in allowing reliable deployment of these networks in critical applications. One direction for obtaining insights on neural networks is through comparison of their internal representations. Comparing neural representations in neural networks is thus a challenging but important problem, which has been approached in different ways. The Centered Kernel Alignment (CKA) similarity metric, particularly its linear variant, has recently become a popular approach and has been widely used to compare representations of a network's different layers, of architecturally similar networks trained differently, or of models with different architectures trained on the same data. A wide variety of conclusions about similarity and dissimilarity of these various representations have been made using CKA. In this work we present an analysis that formally characterizes CKA sensitivity to a large class of simple transformations, which can naturally occur in the context of modern machine learning. This provides a concrete explanation of CKA sensitivity to outliers and to transformations that preserve the linear separability of the data, an important generalization attribute. Finally we propose an optimization-based approach for modifying representations to maintain functional behaviour while changing the CKA value. Our results illustrate that, in many cases, the CKA value can be easily manipulated without substantial changes to the functional behaviour of the models, and call for caution when leveraging activation alignment metrics.

1 Introduction

A helpful framework for thinking about deep learning models is that of *representation learning*, where we view artificial neural networks (ANNs) as learning increasingly complex internal representations as we go deeper through their layers. In practice, it is often of interest to analyze and compare the representations of multiple ANNs. However, the typical high dimensionality of ANN internal representation spaces makes this a fundamentally difficult task.

To address this problem, the machine learning community has tried finding meaningful ways to compare ANN internal representations and various *representation (dis)similarity measures* have been proposed (Li et al., 2015; Wang et al., 2018; Raghu et al., 2017; Morcos et al., 2018). Recently, Centered Kernel Alignment (CKA) (Kornblith et al., 2019) was proposed and shown to be able to reliably identify correspondences between representations in architecturally similar networks trained on the same dataset but from different initializations, unlike past methods such as linear regression or CCA based methods (Raghu et al., 2017; Morcos et al., 2018). While CKA can capture different notions of similarity between points in representation space by using different kernel functions, it was empirically shown in the original work that there are no real benefits to using CKA with a nonlinear

*Equal contribution, name order randomized. †Equal senior-author contribution

ML Safety Workshop, 36th Conference on Neural Information Processing Systems (NeurIPS 2022).

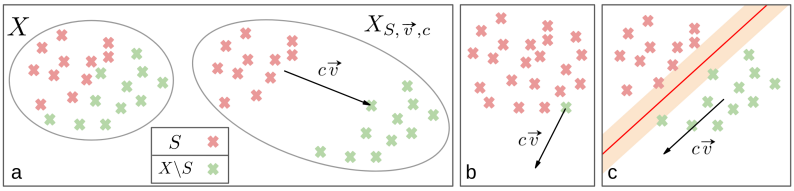

Figure 1: Visual representations of the transformations considered in the theoretical results. (a) Thm. 1: The original set of neural representations X contains subsets S (red) and $X \setminus S$ (green). We can then build $X_{S,\vec{v},c}$ as a copy of X, where the points in $X \setminus S$ are translated a distance c in direction $\vec{v}$. The linear CKA value between X and $X_{S,\vec{v},c}$ is then computed. (b) Cor. 3: X and $X_{S,\vec{v},c}$ differ by a single point, which has been translated by $c\vec{v}$ in $X_{S,\vec{v},c}$. (c) Cor. 4: S and $X \setminus S$ are linearly separable (red line with orange margins), the transformation made to obtain $X_{S,\vec{v},c}$ preserves the linear separability of the data as well as the margins.

kernel over its linear counterpart (Kornblith et al., 2019). As a result, linear CKA has been the preferred representation similarity measure of the machine learning community in recent years and other similarity measures (including nonlinear CKA) are seldomly used. CKA has been utilized in a number of works to derive conclusions regarding the similarity between different models and their behaviours such as wide versus deep ANNs (Nguyen et al., 2021) and Transformer (Vaswani et al., 2017) versus CNN based ANNs (Raghu et al., 2021). Moreover, this similarity metric has been used to draw conclusions about transfer learning (Neyshabur et al., 2020) and catastrophic forgetting (Ramasesh et al., 2021) of ANNs. For a more detailed background on ANN representation comparison and CKA as well as a review of recent related work please see Appendix A.1. Due to this widespread use, it is important to understand how reliable the CKA similarity measure is and in what cases it fails to provide meaningful results. Furthermore, we are interested to know how easily a model can be designed to intentionally deceive the CKA similarity. In this paper, we study CKA sensitivity to a class of simple transformations and show how CKA similarity values can be directly manipulated without noticeable changes in the model final output behaviour. In particular our contributions are as follows: (1) In Sec. 2 and with Thm. 1 we characterize CKA sensitivity to a large class of simple transformations, which can naturally occur in ANNs. With Cor. 3 and 4 we extend our theoretical results to cover CKA sensitivity to outliers, which has been empirically observed in previous work (Nguyen et al., 2021; Ding et al., 2021; Nguyen et al., 2022), and to transformations preserving linear separability of data, an important characteristic for generalization. Concretely, our theoretical contributions show how the CKA value between two copies of the same set of representations can be significantly decreased through simple, functionality preserving transformations of one of the two copies. (2) In Sec. 3 we present a general optimization procedure that allows the CKA value to be heavily manipulated to be either high or low without significant changes to the functional behaviour of the underlying ANNs. We use this to revisit previous findings (Nguyen et al., 2021; Kornblith et al., 2019).

2 CKA sensitivity to subset translation

In this section, we theoretically characterize CKA sensitivity to a wide class of simple transformations, namely the translation of a subset of the representations. We also justify why this class of transformations and the special cases it contains are important in the context of predictive tasks that are solved using neural networks. Our main theoretical result, Thm. 1, shows that any set of internal neural representations X (e.g., from hidden layers of a network) can be manipulated with simple transformations (translations of a subset, see Fig. 1.a) to significantly reduce the CKA between the original and manipulated set. We note that our theoretical results are entirely class and direction agnostic (except for Cor. 4 which is not direction agnostic).

Theorem 1. *Consider a set of n internal representations in p dimensions $X \in \mathbb{R}^{n \times p}$ that have been centered column-wise, let $S \subset X$ such that $|S| = \frac{|X|}{2}$ and $\vec{v}$ such that $\|\vec{v}\| = 1$. We define $X_{S,\vec{v},c} = S \cup \{x + c\vec{v} : x \in X \setminus S\}$. Then we have:*

$$\lim_{c \to \infty} CKA_{lin}(X; X_{S,\vec{v},c}) = (\rho) \frac{\|\mathbb{E}_{x \in S}[x]\|^2}{\mathbb{E}_{x \in X}[\|x\|^2]} \sqrt{\dim_{PR}(X)} \tag{1}$$

where $(\rho) = \frac{\rho}{1-\rho} \in (0; 1]$, and $\dim_{PR}(X) \triangleq \frac{(\sum_i \lambda_i)^2}{\sum_i \lambda_i^2} \in [1; p]$ the dimensionality estimate provided by the participation ratio of eigenvalues $\{\lambda_i\}$ of the covariance of X.

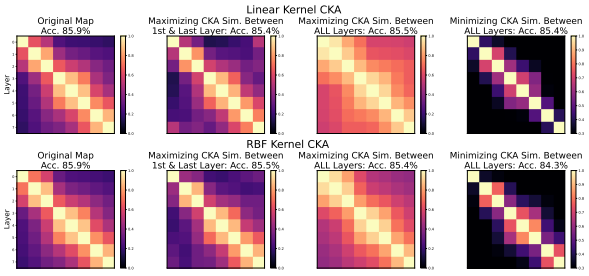

Figure 2: Original Map is the test set CKA map of a network trained on CIFAR10. We manipulate this network to produce CKA maps which: (1) maximizes the CKA measure between the 1st and last layer, (2) maximizes the CKA measure between all layers, and (3) minimizes the CKA measure between all layers. In cases (1) and (2), the network experiences only a slight loss in performance, which counters previous findings by achieving a strong CKA similarity between early and late layers. We find similar results are easily achieved in the kernel CKA case.

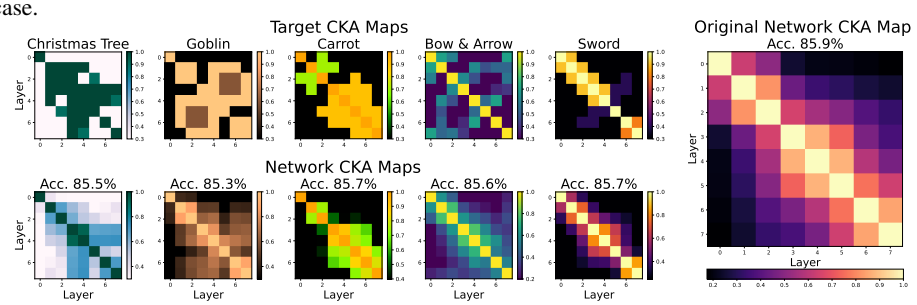

Figure 3: The comical target CKA maps (first row) are used as the objective for the CKA map loss in Eq. 2, while prioritizing network performance (small tolerance for changes in accuracy $_{\text{acc}}$). The second row shows the test set CKA map produced by the network.

We consider a copy of X to which we apply a transformation where the representations of a subset $X \cap S$ of the data is moved a distance c along direction $\hat{v}$, resulting in the modified representation set $X_{S,\hat{v},c}$. A closed form solution is found for the limit of the linear CKA value between X and $X_{S,\hat{v},c}$ as c tends to infinity. We note that up to orthogonal transformations (which CKA is invariant to) the transformation $X \to X_{S,\hat{v},c}$ is not difficult to implement when transforming representations between hidden layers in neural networks. More importantly, it is also easy to eliminate or ignore in a single layer transformation, as long as the weight vectors associated with the neurons in the subsequent layer are orthogonal to $\hat{v}$. Therefore, our results show that from a theoretical perspective, CKA can easily provide misleading information by capturing representation differences introduced by a shift of the form $X \to X_{S,\hat{v},c}$ (especially with high magnitude c), which would have no impact on network operations or their effective task-oriented data processing. For a more detailed analysis of Thm. 1, corollaries discussing the symmetry of Thm.1 (Cor. 2), and CKA sensitivity to outliers (Cor. 3) and to transformations that preserve the linear separability of the representations (Cor. 4) please see Appendix A.2. In the appendix we also discuss possible extensions of these results to nonlinear CKA and provide proofs.

3 An Adversarial Attack on the CKA Map

We now study a generic approach inspired by the adversarial attack literature (Goodfellow et al., 2014) that tries to modify a model such that it can provide any desired interpretation using CKA, while keeping the observed functional behavior of the model the same. Although not explicitly tied to our theory in the Appendix C.7 we illustrate empirically that our theoretical results serve as a clear basis for understanding of how these attacks can be so successful.

The CKA map, commonly used to analyze network architectures (Kornblith et al., 2019) and their inter-layers similarities, is a matrix M, where $M[i, j]$ is the CKA value between the activations of layers i, and j of a network. In many works (Nguyen et al., 2021; Raghu et al., 2021; Nguyen et al., 2022) these maps are used explicitly to obtain insights on how different models behave, compared to one another. However, as seen in our analysis so far it is possible to manipulate the CKA similarity value, decreasing and increasing it without changing the behaviour of the model on a target task. In this section we set to directly manipulate the CKA map of a trained network f_θ , by adding the desired CKA map, M_{target}, to its optimization objective, while maintaining its original outputs via distillation loss (Hinton et al., 2015). The goal is to determine if the CKA map can be changed while

keeping the model performance the same, suggesting the behaviour of the network can be maintained while changing the CKA measurements. To accomplish this we optimize f_θ over the training set $(X; Y)$ (note however, that the results are shown on the test set) via the following objective:

$$^{*}_{new} = \text{argmin}_\theta \left(L_{distill} \left(f_\theta \ (X); f_\theta(X) \right) + \ L_{map} \left(M_{f \ (X)}; M_{target} \right) \right) \tag{2}$$

where $L_{map} \left(M_{f \ (X)}; M_{target} \right) = \sum_{i,j} \ln \cosh \left(M[i; j]_{f \ (X)} \quad M[i; j]_{target} \right)$. The multiplier in Eq. 2 is the weight that balances the two losses. Making large will favour the agreements between the target and network CKA maps over preservation of the network outputs. In our experiments is allowed to change dynamically at every optimization step. Using the validation set accuracy as a surrogate metric for how well the network's representations are preserved, is then modulated to learn maps. If the difference between the original accuracy of the network and the current validation accuracy is above a certain threshold ($_{\text{acc}}$) we scale down to emphasize the alignment of the network output with the outputs of f_θ , otherwise we scale it up to encourage finer agreement between the target and network CKA maps (see Appendix B.4 for the pseudo code).

Fig. 2 shows the test set CKA map of f_θ along with the test set CKA map of three scenarios we investigated: (1) maximizing the CKA similarity between the 1st and last layer, (2) maximizing the CKA similarity between all layers, and (3) minimizing the CKA similarity between all layers (for network architecture and training details see Appendix B.3). In cases (1) and (2), the network performance is barely hindered by the manipulations of its CKA map. This is surprising and contradictory to the previous findings (Kornblith et al., 2019; Raghu et al., 2021) as it suggests that it is possible to achieve a strong CKA similarity between early and later layers of a well-trained network without noticeably changing the model behaviour. Similarly, we observe that for the RBF kernel based CKA (Kornblith et al., 2019) we can obtain manipulated results using the same procedure. The bandwidth for the RBF kernel CKA is set to 0.8 of the median Euclidean distance between representations (Kornblith et al., 2019). In the Appendix C.3 we also show similar analysis on other values. We further experiment with manipulating the CKA map of f_θ to produce a series of comical CKA maps (Fig. 3) while maintaining similar model accuracy. Although the network CKA maps seen in Fig. 3 closely resemble their respective targets, it should be noted that we prioritized maintaining the network outputs, and ultimately its accuracy by choosing small $_{\text{acc}}$. Higher thresholds of accuracy result in stronger agreements between the target and network CKA maps at the cost of performance.

See Appendix for further experiments on CKA map optimization with wider (vs. narrower) networks as well as an analysis of how these adversarial attacks suggest the mechanism of action found by the direct optimization utilizes an approach similar to the theoretical construction.

4 Discussion and Conclusion

We have first presented a formal characterization of CKA's sensitivity to translations of a subset of the representations, a simple yet large class of transformations that is highly meaningful in the context of deep learning. This characterization has provided a theoretical explanation to phenomena observed in practice, namely CKA sensitivity to outliers and to directions of high variance. Moreover, our theoretical analysis shows how the CKA value between two sets of representations can diminish even if they share local structure and are linearly separable by the same hyperplanes, with the same margins. This meaningful way in which two sets of representations can be similar, as justified by classical machine learning theory and seminal deep learning results, is therefore not captured by linear CKA. Secondly, we show an optimization framework that can manipulate CKA in networks to result in arbitrarily low/high values while preserving functional behaviour, which we use to revisit previous findings (Nguyen et al., 2021; Kornblith et al., 2019). Our theoretical results allow us to understand how such a framework can easily achieve a desired result (further explored in the Appendix).

Some of the problematic transformations we identify are not necessarily encountered in many applications. However, given the popularity of this method and the exclusive way it has been applied to compare representations in recent years, we believe it is necessary to better understand its sensitivities and the ways in which it can be manipulated. We particularly note that such manipulations can be undertaken by malicious actors aiming to mislead users of standard analysis tools like CKA. Our results call for caution when leveraging linear CKA, as well as other representations similarity measures, and especially when the procedure used to produce the model is not known, consistent, or controlled. An example of such a scenario is the increasingly popular use of open-sourced pre-trained models. Finally, our adversarial optimization framework can provide a basis for interrogating improved similarity metrics, potentially leading to more robust approaches through an iterative process of creating metrics more robust to adversarial attack.

5 Acknowledgements

This work was partially funded by OpenPhilanthropy [M.D., E.B.]; NSERC CGS D, FRQNT B1X & UdeM A scholarships [S.H.]; NSERC Discovery Grant RGPIN-2018-04821 & Samsung Research Support [G.L.]; and Canada CIFAR AI Chairs [G.L., G.W.]. This work is also supported by resources from Compute Canada and Calcul Quebec. The content is solely the responsibility of the authors and does not necessarily represent the views of the funding agencies.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2016. URL https://arxiv.org/abs/1610.01644.

Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 6679–6687, 2021.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SyK00v5xx.

Peter Bartlett and John Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. *Advances in Kernel methods—support vector learning*, pp. 43–54, 1999.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to ImageNet. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 583–593. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/belilovsky19a.html.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

MohammadReza Davari, Leila Kosseim, and Tien Bui. TIMBERT: Toponym identifier for the medical domain based on BERT. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 662–668, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.58. URL https://aclanthology.org/2020.coling-main.58.

MohammadReza Davari, Nader Asadi, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. Probing representation forgetting in supervised and unsupervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16712–16721, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding representation similarity through statistical testing. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=_kwj6V53ZqB.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Shimon Edelman. Representation is representation of similarities. *Behavioral and Brain Sciences*, 21 (4):449–467, 1998. doi: 10.1017/S0140525X98001253.

Farhood Farahnak, Elham Mohammadi, MohammadReza Davari, and Leila Kosseim. Semantic similarity matching using contextualized representations. In *Proceedings of the 34th Canadian Conference on Artificial Intelligence (CanAI 2021)*, Vancouver, Canada (Online), June 2021.

Matthew Farrell, Stefano Recanatesi, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Recurrent neural networks learn robust representations by dynamically balancing compression and expansion. *bioRxiv*, 2019. doi: 10.1101/564476. URL https://www.biorxiv.org/content/early/2019/12/18/564476.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *Lecture Notes in Computer Science*, pp. 63–77. Springer Berlin Heidelberg, 2005. doi: 10.1007/11564089_7. URL https://doi.org/10.1007/11564089_7.

Tom George Grigg, Dan Busbridge, Jason Ramapuram, and Russ Webb. Do self-supervised and supervised methods learn similar visual representations?, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

Stefan Horoi, Victor Geadah, Guy Wolf, and Guillaume Lajoie. Low-dimensional dynamics of encoding and learning in recurrent neural networks. In Cyril Goutte and Xiaodan Zhu (eds.), *Advances in Artificial Intelligence*, pp. 276–282, Cham, 2020. Springer International Publishing. ISBN 978-3-030-47358-7.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Jörn-Henrik Jacobsen, Arnold W.M. Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJsjkMb0Z.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Simon Kornblith, Ting Chen, Honglak Lee, and Mohammad Norouzi. Why do better loss functions lead to less transferable features? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=8twKpG5s8Qh.

Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2, 2008. ISSN 1662-5137. doi: 10.3389/neuro.06.004.2008.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, MIT & NYU, 2009.

Aarre Laakso and Garrison Cottrell. Content and cluster analysis: Assessing representational similarity in neural systems. *Philosophical Psychology*, 13(1):47–76, 2000. doi: 10.1080/09515080050002726. URL https://doi.org/10.1080/09515080050002726.

Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. Lower Bounds on the VC Dimension of Smoothly Parameterized Function Classes. *Neural Computation*, 7(5):1040–1053, 09 1995. ISSN 0899-7667. doi: 10.1162/neco.1995.7.5.1040. URL https://doi.org/10.1162/neco.1995.7.5.1040.

Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. *International Journal of Computer Vision*, 127(5):456–476, May 2018. doi: 10.1007/s11263-018-1098-y. URL https://doi.org/10.1007/s11263-018-1098-y.

Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? In Dmitry Storcheus, Afshin Rostamizadeh, and Sanjiv Kumar (eds.), *Proceedings of the 1st International Workshop on Feature Extraction: Modern Questions and Challenges at NIPS 2015*, volume 44 of *Proceedings of Machine Learning Research*, pp. 196–212, Montreal, Canada, 11 Dec 2015. PMLR. URL https://proceedings.mlr.press/v44/li15convergent.html.

Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and L.F. Abbott. Optimal degrees of synaptic connectivity. *Neuron*, 93(5):1153–1164.e7, 2017. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2017.01.030. URL https://www.sciencedirect.com/science/article/pii/S0896627317300545.

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*, pp. 10012–10022, 2021.

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

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Isabel I.C. Low, Alex H. Williams, Malcolm G. Campbell, Scott W. Linderman, and Lisa M. Giocomo. Dynamic and reversible remapping of network representations in an unchanging environment. *Neuron*, 109(18):2967–2980.e11, 2021. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2021.07.005. URL https://www.sciencedirect.com/science/article/pii/S0896627321005043.

Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/5f5d472067f77b5c88f69f1bcfda1e08-Paper.pdf.

Luca Mazzucato, Alfredo Fontanini, and Giancarlo La Camera. Stimuli reduce the dimensionality of cortical activity. *Frontiers in Systems Neuroscience*, 10, 2016. ISSN 1662-5137. doi: 10.3389/fnsys.2016.00011. URL https://www.frontiersin.org/article/10.3389/fnsys.2016.00011.

PhD Mingzhou Ding and PhD Dennis Glanzman. *The Dynamic Brain*. Oxford University Press, January 2011. doi: 10.1093/acprof:oso/9780195393798.001.0001. URL https://doi.org/10.1093/acprof:oso/9780195393798.001.0001.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/a7a3d70c6d17a73140918996d03c014f-Paper.pdf.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 512–523. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/0607f4c705595b911a4f3e7a127b44e0-Paper.pdf.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=KJNcAkY8tY4.

Thao Nguyen, Maithra Raghu, and Simon Kornblith. On the origins of the block structure phenomenon in neural network representations. *arXiv preprint arXiv:2202.07184*, 2022.

Edouard Oyallon. Building a regular decision boundary with deep networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1886–1894, 2017. doi: 10.1109/CVPR.2017.204.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34, 2021.

Vinay Venkatesh Ramasesh, Ethan Dyer, and Maithra Raghu. Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LhY8QdUGSuw.

Le Song, Alex Smola, Arthur Gretton, Justin Bedo, and Karsten Borgwardt. Feature selection via dependence maximization. *Journal of Machine Learning Research*, 13(5), 2012.

David Sussillo and Omri Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 25(3):626–649, 03 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00409. URL https://doi.org/10.1162/NECO_a_00409.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant CNNs for digital pathology. *arXiv preprint arXiv:1806.03962*, June 2018.

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 Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6438–6447. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/verma19a.html.

Liwei Wang, Lunjia Hu, Jiayuan Gu, Zhiqiang Hu, Yue Wu, Kun He, and John Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/5fc34ed307aac159a30d81181c99847e-Paper.pdf.

Alex H Williams, Erin Kunz, Simon Kornblith, and Scott Linderman. Generalized shape metrics on neural representations. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=L9JM-pxQOl.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pp. 818–833. Springer International Publishing, 2014a. doi: 10.1007/978-3-319-10590-1_53. URL https://doi.org/10.1007/978-3-319-10590-1_53.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014b.

Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

A Complements to The Main Text

A.1 Background on CKA and Related Work

Comparing Representations Let $X \in \mathbb{R}^{n \times d_1}$ denote a set of ANN internal representations, i.e., the neural activations of a specific layer with d_1 neurons in a network, in response to $n \in \mathbb{N}$ input examples. Let $Y \in \mathbb{R}^{n \times d_2}$ be another set of such representations generated by the same input examples but possibly at a different layer of the same, or different, deep learning model. It is standard practice to center these representations column-wise (feature or "neuron" wise) before analyzing them. We are interested in representation similarity measures, which try to capture a certain notion of similarity between X and Y.

Quantifying similarity Li et al. (2015) have considered one-to-one, many-to-one and many-to-many mappings between neurons from different neural networks, found through activation correlation maximization. Wang et al. (2018) extended that work by providing a rigorous theory of neuron activation subspace match and algorithms to compute such matches between neurons. Alternatively, Raghu et al. (2017) introduced SVCCA where singular value decomposition is used to identify the most important directions in activation space. Canonical correlation analysis (CCA) is then applied to find maximally correlated singular vectors from the two sets of representations and the mean of the correlation coefficients is used as a similarity measure. In order to give less importance to directions corresponding to noise, Morcos et al. (2018) introduced projection weighted CCA (PWCCA). The PWCCA similarity measure corresponds to the weighted sum of the correlation coefficients, assigning more importance to directions in representation space contributing more to the output of the layer. Many other representation similarity measures have been proposed based on linear classifying probes (Alain & Bengio, 2016; Davari et al., 2022), fixed points topology of internal dynamics in recurrent neural networks (Sussillo & Barak, 2013; Maheswaranathan et al., 2019), solving the orthogonal Procrustes problem between sets of representations (Ding et al., 2021; Williams et al., 2021) and many more (Laakso & Cottrell, 2000; Lenc & Vedaldi, 2018; Arora et al., 2017). We also note that a large body of neuroscience research has focused on comparing neural activation patterns in biological neural networks (Edelman, 1998; Kriegeskorte et al., 2008; Williams et al., 2021; Low et al., 2021).

CKA Centered Kernel Alignment (CKA) (Kornblith et al., 2019) is another such similarity measure based on the Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005) that was presented as a means to evaluate independence between random variables in a non-parametric way. For $K_{i,j} = k(x_i; x_j)$ and $L_{i,j} = l(y_i; y_j)$ where $k; l$ are kernels and for $H = I \quad \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ the centering matrix, HSIC can be written as: $\text{HSIC}(K; L) = \frac{1}{(n-1)^2} tr(KHLH)$. CKA can then be computed as:

$$\text{CKA}(K; L) = \frac{\text{HSIC}(K; L)}{\sqrt{\text{HSIC}(K; K)\text{HSIC}(L; L)}} \tag{3}$$

In the linear case k and l are both the inner product so $K = XX^\top$, $L = YY^\top$ and we use the notation $CKA(X; Y) = CKA(XX^\top; YY^\top)$. Intuitively, HSIC computes the similarity structures of X and Y, as measured by the kernel matrices K and L, and then compares these similarity structures (after centering) by computing their alignment through the trace of $KHLH$.

Recent CKA Results CKA has been used in recent years to make various claims about neural network representations. Nguyen et al. (2021) used CKA to establish that parameter initialization drastically impact feature similarity and that the last layers of overparameterized (very wide or deep) models learn representations that are very similar, characterized by a visible "block structure" in the networks CKA heatmap. CKA has also been used to compare vision transformers with convolutional neural networks and to find striking differences between the representations learned by the two architectures, such as vision transformers having more uniform representations across all layers (Raghu et al., 2021). Ramasesh et al. (2021) have used CKA to find that deeper layers are especially responsible for forgetting in continual learning settings.

Most closely related to our work, Ding et al. (2021) demonstrated that CKA lacks *sensitivity* to the removal of low variance principal components from the analyzed representations even when this removal significantly decreases probing accuracy. Moreover, Nguyen et al. (2022) found that the previously observed high CKA similarity between representations of later layers in large capacity models (so-called block structure) is actually caused by a few dominant data points that share similar

characteristics. Williams et al. (2021) discussed how CKA does not respect the triangle inequality, which makes it problematic to use CKA values as a similarity measure in downstream analysis tasks. We distinguish ourselves from these papers by providing theoretical justifications to CKA sensitivity to outliers and to directions of high variance which were only empirically observed in Ding et al. (2021); Nguyen et al. (2021). Furthermore, we do not only present situations in which CKA gives unexpected results but we also show *how* CKA values can be manipulated to take on arbitrary values.

Nonlinear CKA The original CKA paper (Kornblith et al., 2019) stated that, in practice, CKA with a nonlinear kernel achieves similar results as linear CKA across the considered experiments. Potentially as a result of this, all subsequent papers which used CKA as a neural representation similarity measure have used linear CKA (Maheswaranathan et al., 2019; Neyshabur et al., 2020; Nguyen et al., 2021; Raghu et al., 2021; Ramasesh et al., 2021; Ding et al., 2021; Williams et al., 2021; Kornblith et al., 2021), and to the best of our knowledge, no published work besides Kornblith et al. (2019); Nguyen et al. (2022) has used CKA with a nonlinear kernel. Consequently, we largely focus our analysis on linear CKA which is the most popular method and the one actually used in practice. However, our empirical results suggest that many of the observed problems hold for CKA with an RBF kernel and we discuss a possible way of extending our theoretical results to the nonlinear case.

A.2 Complement to The Theoretical Results

Corollary 2. *Thm. 1 holds even if S is taken such that* $= \frac{|S|}{|X|}$ 2 $(0.5; 1)$.

The terms in Eq. 1 can each be analyzed individually. $(\ )$ depends entirely on , the proportion of points in $X_{S,\sharp,c}$ that have not been translated i.e. that are exactly at the same place as in X. Its value is between 0 and 1 and it tends towards 0 for small sizes of S. The participation ratio, with values in $[1; p]$, is used as an effective dimensionality estimate for internal representations (Mingzhou Ding & Dennis Glanzman, 2011; Mazzucato et al., 2016; Litwin-Kumar et al., 2017). It has long been observed that the effective dimensionality of internal representations in neural networks is far smaller than the actual number of dimensions of the representation space (Farrell et al., 2019; Horoi et al., 2020). $\mathbb{E}_{x\in X}[kxk^2]$ and $k\mathbb{E}_{x\in S}[x]k^2$ are respectively the average squared norms of all representations in X and the squared norm of the mean of S, the subset of representations that are not being translated. Since most neural networks are trained using weight decay, the network parameters, and hence the resulting representations as well as these two quantities are biased towards small values in practice.

CKA Sensitivity to Outliers As mentioned in Appendix A.1, it was recently found that the block structure in CKA heatmaps of high capacity models was caused by a few dominant data points that share similar characteristics (Nguyen et al., 2022). Other works have empirically highlighted CKA's sensitivity to directions of high variance, failing to detect important, function altering changes that occur in all but the top principal components (Ding et al., 2021). Cor. 3 provides a concrete explanation to these phenomena by treating the special case of Thm. 1 where only a single point, $\hat{x}$ is moved and thus has a different position in $X_{S,\sharp,c}$ with respect to X, see Fig. 1.b for an illustration. We note that the term "subset translation" was coined by us and was not used in the past works. However, all the papers referenced in this paragraph and later in this section present naturally occurring examples of subset translations in a set of representations relative to another, comparable set.

Corollary 3. *Thm. 1 holds in the special case where $S = f\hat{x}g$ is a single point, i.e. an outlier.*

Cor. 3 exposes a key weakness of linear CKA: its sensitivity to outliers. Consider two sets of representations that are identical in all aspects except for the fact that one of them contains an outlier, i.e. a representation further away from the others. Cor. 3 then states that as the difference between the outlier's position in the two sets of representations becomes large the CKA value between the two sets drops dramatically, indicating high dissimilarity. Indeed, as previously noted, $\frac{\|\mathbb{E}_{x\,2\,S}[x]\|^2}{\mathbb{E}_{x\,2\,X}[\|x\|^2]}\sqrt{\dim_{PR}(X)}$ will be of relatively small value in practice so the whole expression in Eq. 1 will be dominated by $(\ ) = \frac{\rho}{1-\rho}$. In the outlier case $(\ )$ $= \frac{1}{\#\text{ representations}}$ which will be extremely small since for most modern deep learning datasets the number of examples in both the training and test sets is in the tens of thousands or more. This will drastically lower the CKA value between the two considered representations despite their obvious similarity.

CKA Sensitivity to Transformations Preserving Linear Separability Classical machine learning theory highlights the importance of data separability and of margin size for predictive models generalization (Lee et al., 1995; Bartlett & Shawe-Taylor, 1999). Large margins, i.e. regions surrounding the separating hyperplane containing no data points, are associated with less overfitting, better generalization and greater robustness to outliers and to noise. The same concepts naturally arise in the study of ANNs with past work establishing that internal representations become almost perfectly linearly separable by the network's last layer (Zeiler & Fergus, 2014a; Oyallon, 2017; Jacobsen et al., 2018; Belilovsky et al., 2019). Furthermore, the quality of the separability, the margin size and the decision boundary smoothness have all been linked to generalization in neural networks (Verma et al., 2019). Given the theoretical and practical importance of these concepts and their natural prevalence in deep learning models it is reasonable to assume that a meaningful way in which two sets of representations can be "similar" is if they are linearly separable by the same hyperplanes in representation space and if their margins are equally as large. This would suggest that the exact same linear classifier could accurately classify both sets of representations. Cor. 4 treats this exact scenario as a special case of Thm. 1, see Fig. 1.c for an illustration. If X contains two linear separable subsets, S and $X \setminus S$, we can create $X_{S,\vec{v},c}$ by translating one of the subsets in a direction that preserves the linear separability of the representations and the size of the margins while simultaneously decreasing the CKA between the original and the transformed representations, counterintuitively indicating a low similarity between representations.

Corollary 4. *Assume S and $X \setminus S$ are linearly separably i.e. $\exists w \in \mathbb{R}^p$, the separating hyperplane's normal vector, and $k \in \mathbb{R}$ such that for every representation $x \in X$ we have: $x \in S \implies \langle w, x \rangle < k$ and $x \in X \setminus S \implies \langle w, x \rangle > k$. We can then pick $\vec{v}$ such that S and $\{x + c\vec{v} : x \in X \setminus S\}$ are linearly separable by the exact same hyperplane and with the exact same margins as S and $X \setminus S$ for any value of $c \in \mathbb{R}_{\geq 0}$ and Thm. 1 still holds.*

Extensions to Nonlinear CKA As previously noted in Appendix A.1, given the popularity of linear CKA, it is outside the scope of our work to theoretically analyze nonlinear kernel CKA. However one can consider extending our theoretical results to the nonlinear CKA case with symmetric, positive definite kernels. Indeed we know from reproducing kernel Hilbert space (RKHS) theory that we can write such a kernel as an inner product in an implicit Hilbert space. While directly translating points in the representations space would likely not drive CKA values down as in the linear case, it would suffice to find/learn which transformations in representations space correspond to translations in the implicit Hilbert space. Our results should hold if we apply the found transformations, instead of translations, to a subset of the representations. Although practically harder to implement than simple translations, we hypothesize that it would be possible to learn such transformations.

A.3 Practical Implications of The Theoretical Results

Here we empirically test the behaviour of linear and RBF CKA in situations inspired by our theoretical analysis, first in an artificial setting, then in a more realistic one. We begin with artificially generated representations $X \in \mathbb{R}^{n \times d}$ to which we apply subset translations to obtain $Y \in \mathbb{R}^{n \times d}$, similar to what is described in Thm. 1. We generate X by sampling 10K points uniformly from the 1K-dimensional unit cube centered at the origin and 10K points from a similar cube centered at $(1.1, 0, 0, \ldots, 0)$, so the points from the two cubes are linearly separable along the first dimension. We translate the representations from the second cube in a random direction sampled from the d-dimensional ball and we plot the CKA values between X and Y as a function of the translation distance in Fig. 4.a. This transformation entirely preserves the topological structure of the representations as well as their local geometry since the points sampled from each cube have not moved with respect to the other points sampled from the same cube and the two cubes are still separated, only the distance between them has been changed. Despite these multiple notions of "similarity" between X and Y being preserved, the CKA values quickly drop below 0.2 for both linear and RBF CKA. While our theoretical results (Thm. 1) predicted this drop for linear CKA, it seems that RBF CKA is also highly sensitive to translations of a subset of the representations. Furthermore, it is surprising to see that the drop in CKA value occurs even for relatively small translation distances. We note that RBF CKA with σ equal to 0.2 times the median distance between examples is unperturbed by the considered transformation. However, as we observe ourselves (RBF CKA experiments in the supplement) and as was found in the original CKA paper (see Table 2 of Kornblith et al. (2019)), RBF CKA with $\sigma = 0.2$ median is significantly less informative than RBF CKA with higher values of σ. With small values of σ, RBF CKA only captures very local, possibly trivial, relationships.

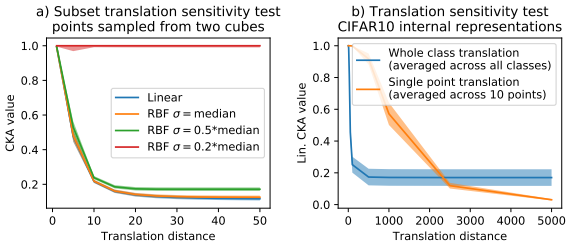

Figure 4: (a) Linear and RBF CKA values between the artificial representations X and the subset translated version Y as a function of the translation distance. (b) CKA value between a CNN's internal representations of the CIFAR10 training set and modified versions where either a class or a single point is translated as functions of the translation distance.

In a more realistic setting we test the practical implications of linear CKA sensitivity to outliers (see Cor. 3) and to transformations that preserve the linear separability of the data as well as the margins (see Cor. 4). We consider the 9 layers CNN presented in Sec. 6.1 of Kornblith et al. (2019) trained on CIFAR10. As argued in Appendix A.1, when trained on classification tasks, ANNs tend to learn increasingly complex representations of the input data that can be almost perfectly linearly separated into classes by the last layer of the network. Therefore a meaningful way in which two sets of representations can be "similar" in practice is if they are linearly separable by the same hyperplanes in parameter space, with the same margins. Given X, the network's internal representations of 10k training images at the last layer before the output we can use an SVM classifier to extract the hyperplanes in parameter space which best separate the data (with approx. 91% success rate). We then create Y by translating a subset of the representations in a direction which won't cross these hyperplanes, and won't affect the linear separability of the representations. We plot the CKA values between X and Y according to the translation distance in Fig. 4.b. The CKA values quickly drop to 0, despite the existence of a linear classifier that can classify both sets of representations into the correct classes with $> 90\%$ accuracy. In Fig. 4.b we also examine linear CKA's sensitivity to outliers. Plotted are the CKA values between the set of training image representations and the same representations but with a single point being translated from its original location. While the translation distance needed to achieve low CKA values is relatively high, the fact that the position of a *single* point out of *tens of thousands* can so drastically influence the CKA value raises doubts about CKA's reliability as a similarity metric.

We note that our main theoretical results, namely Thm. 1, Cor. 2 and Cor. 3 are entirely class and direction agnostic. The empirical results presented in this section are simply examples that we deemed particularly important in the context of ML but the same results would hold with any subset of the representations and translation direction, even randomly chosen ones. This is important since the application of CKA is not restricted to cases where labels are available, for example it can also be used in unsupervised learning settings (Grigg et al., 2021). Furthermore, the subset translations presented here were added manually to be able to run the experiments in a controlled fashion but these transformations can naturally occur in ANNs, as discussed in Sec. 2, and one would not necessarily know that they have occurred. We also run experiments to evaluate CKA sensitivity to invertible linear transformations, see the Appendix for justification and results.

B Experimental details

B.1 Minibatch CKA

In our experiments (with the exception of Appendix A.3), in order to reduce memory consumption, we use the minibatch implementation of the CKA similarity Nguyen et al. (2021, 2022). More precisely, let be a kernel function (we experiment with linear and RBF kernel), and $X_b \in \mathbb{R}^{m \times n_1}$ and $Y_b \in \mathbb{R}^{m \times n_2}$ be the minibatches of m samples from two network layers containing n_1 and n_2 neurons respectively. We estimate the value of CKA by averaging the Hilbert-Schmidt independence criterion (HSIC), over all minibatches $b \in B$ via:

$$\mathrm{CKA}_{\mathrm{minibatch}} = \frac{\frac{1}{|B|}\sum_{b\in B}\mathrm{HSIC}_1(Q_b;Z_b)}{\sqrt{\frac{1}{|B|}\sum_{b\in B}\mathrm{HSIC}_1(Q_b;Q_b)}\sqrt{\frac{1}{|B|}\sum_{b\in B}\mathrm{HSIC}_1(Z_b;Z_b)}} \tag{4}$$

Where $Q_b = (X_b; X_b)$, $Z_b = (Y_b; Y_b)$, and HSIC_1 is the unbiased estimator of HSIC Song et al. (2012), hence the value of the CKA is independent of the batch size.

B.2 Network Architecture

In Appendix C.6 we use a 9 layer neural network; the first 8 of these layers are convolution layers and the last layer is a fully connected layer used for classification. We use ReLU (Nair & Hinton, 2010) throughout the network. The kernel size of every convolution layer is set to $(3; 3)$ except the first two convolution layers, which have $(7; 7)$ kernels. All convolution layers follow a padding of 0 and a stride of 1. Number of kernels in each layer of the network, from the lower layers onward follows: $[16; 16; 32; 32; 32; 64; 64]$. In this network, every convolution layer is followed by batch normalization (Ioffe & Szegedy, 2015). The network we used in Sec. 3 to obtain Figures 2 and 3 is similar to the network we just described, except the kernel size for all layers are set to $(3; 3)$. For the experiments in Appendix C.2, we use a ResNet-34 (He et al., 2016) network, where we scale up the channels of the network to increase the its width (see Fig. 6).

B.3 Training Details

The models in Appendix C.6, both the generalized and memorized network, were trained for 100 epochs using AdamW (Loshchilov & Hutter, 2017) optimizer with a learning rate (LR) of `1e-3` and a weight decay of `5e-4`. The LR is follows cosine LR scheduler (Loshchilov & Hutter, 2016) with an initial LR stated earlier.

The training of the base model (original) model in Sec. 3 seen in Figures 2 and 3 follows the same training procedure as of the models from Sec. C.6, except in this setting we train the model for 200 epochs, with an initial LR of 0.01. All other models in Sec. 3 seen in Figures 2 and 3 (with a target CKA map to optimize) are also trained with similar training hyperparameters to that of the base model, except the followings: (1) these models are only trained for 30 epochs. (2) the objective function includes a hyperparameter (see Eq. 2), which we initially set to 500 for all models and is changed dynamically following the Algo. 1 during the training by 0.8 on each iteration. (3) The cosine LR scheduler includes a warm-up step of 500 optimization steps. (4) the LR is set to `1e-3` (4) The distillation loss in the objective function depends on a temperature parameter, which we set to 0.2.

The training procedure for the experiments in Appendix C.2 is similar to the previous training procedures in this section (Figures 2 and 3). Except that the *Original* models are trained for only 100 epochs and the *Optimized w.r.t Target Maps* models are trained for 15 epochs.

B.4 CKA Map Loss Balance

Algo. 1 shows the pseudo code of the dynamical scaling of the loss balance parameter seen in Eq. 2. Using the validation set accuracy as a surrogate metric for how well the network's representations are preserved, is then modulated to learn maps. If the difference between the original accuracy of the network and the current validation accuracy is above a certain threshold ($_{\text{acc}}$) we scale down to emphasize the alignment of the network output with the outputs of f_θ , otherwise we scale it up to encourage finer agreement between the target and network CKA maps.

C Additional Results

C.1 CKA Sensitivity to Invertible Linear Transformations

For linear CKA, we experiment with a type of transformation that is not considered by our theoretical results but which we deemed interesting to analyze empirically, namely multiplications by invertible matrices. Consider a matrix $M \in \mathbb{R}^{d\times d}$ whose elements are sampled from a Gaussian distribution with mean and standard deviation . We verify the invertibility of M since it is not guaranteed and only keep invertible matrices. Fig. 5 shows the CKA values between X and the transformed $Y = XM$. Since this is an invertible linear transformation we expect it to only modestly change the representations in X and the CKA value to be only slightly lower than 1. However, we observe that even for small values of and , CKA drops to 0, which suggests that the two sets of representations are dissimilar and not linked by a simple, invertible transformation.

While Thm.1 of Kornblith et al. (2019) implies that invariance to invertible linear transformations is generally not a desirable property for ANN representation similarity measures, there are relatively

Algorithm 1: Dynamical balancing of Distillation and CKA map loss in Eq. 2

Data: 100 Original Accuracy > 0; 100 Current Validation Accuracy > 0; Accuracy Threshold 0; 1 Scaling Factor > 0; Initial Lambda > 0

Result:

acc_0 Original Accuracy;

acc_1 Current Validation Accuracy;

Accuracy Threshold;

Scaling Factor;

Initial Lambda;

$_{acc}$ acc_0 acc_1;

if $_{acc}$ > **then**

|

else

| $\lambda = _{\alpha}$

end

Return ;

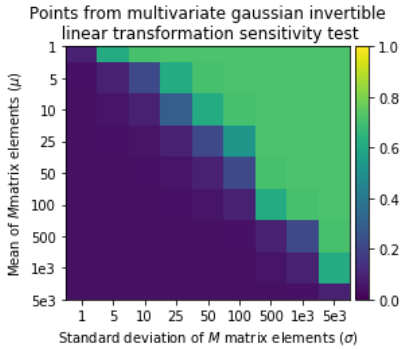

Figure 5: Linear CKA values between the artificial representations X and $Y = XM$ with M being an invertible matrix with elements sampled from $N(; ^2)$ as a function of and . The mean and standard deviation across 10 random instantiations the translation direction and M are shown.

common scenarios in which the hypotheses of the theorem are not necessarily respected, i.e. where the dataset size is larger than the width of the layer. Such is the case in smaller ANNs or even at the last layers of large models which are often fully connected and of far smaller size than the input space or the intermediate layers. Given these situations we see no reason to completely dismiss this invariance as being possibly desirable in certain, albeit not all, contexts.

C.2 Wider Networks

Nguyen et al. (2021) and Nguyen et al. (2022) studied the behaviour of wider and deeper networks using CKA maps, obtaining a block structure, which was subsequently used to obtain insights. We revisit these results and investigate whether the CKA map corresponding to a wider network can be mapped to a thin network. Our results for the test set of CIFAR10 dataset and ResNet-34 (He et al., 2016) are shown in Fig. 6 (for details on the architecture and training procedure see Appendix B.3). We observe that the specific structures associated with wider network can be completely removed and the map can be nearly identical to the thinner model without changing the performance substantially.

In Fig. 7, we use the same networks introduced earlier in Fig. 6 (trained on CIFAR10) and measure their CKA similarity maps over the test set of the Patch Camelyon dataset (Veeling et al., 2018). Patch Camelyon dataset contains histopathologic scans of lymph node sections, which is drastically different from the CIFAR10 dataset both in terms of pixel distribution and the semantics of the data. As we can see in Fig. 7, even under this drastic shift in data distribution the CKA maps of the networks *Optimized w.r.t Target CKA Map* resemble the CKA map of the thin target network, suggesting the generalizability of the CKA map optimization.

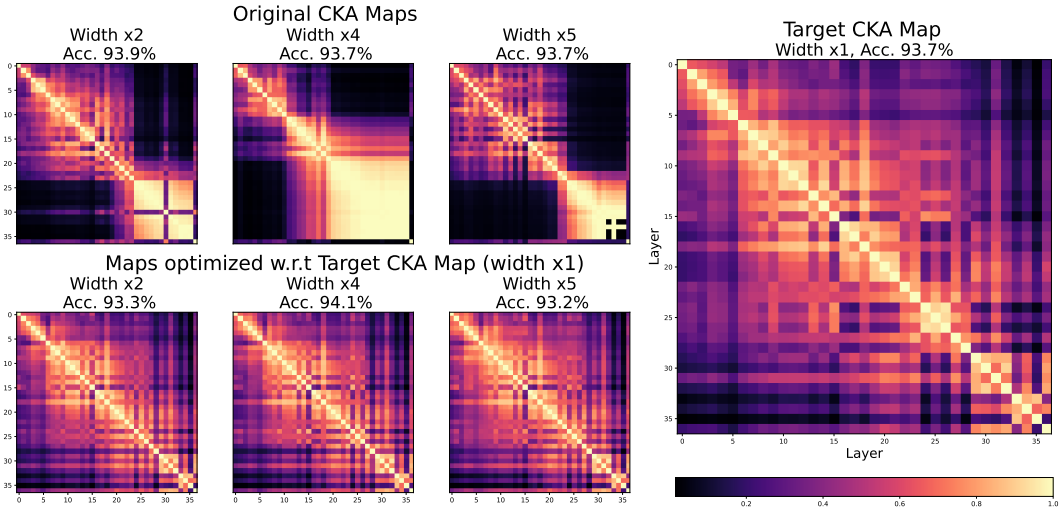

Figure 6: ResNet-34 networks of different widths and their corresponding CKA Maps are modified to produce CKA maps of thin networks. Top row **Original** shows the unaltered test set CKA map of the networks derived from “normal" training. **Optimized** shows the test set CKA map of the networks after their map is optimized to mimic the thin network *target* CKA map.

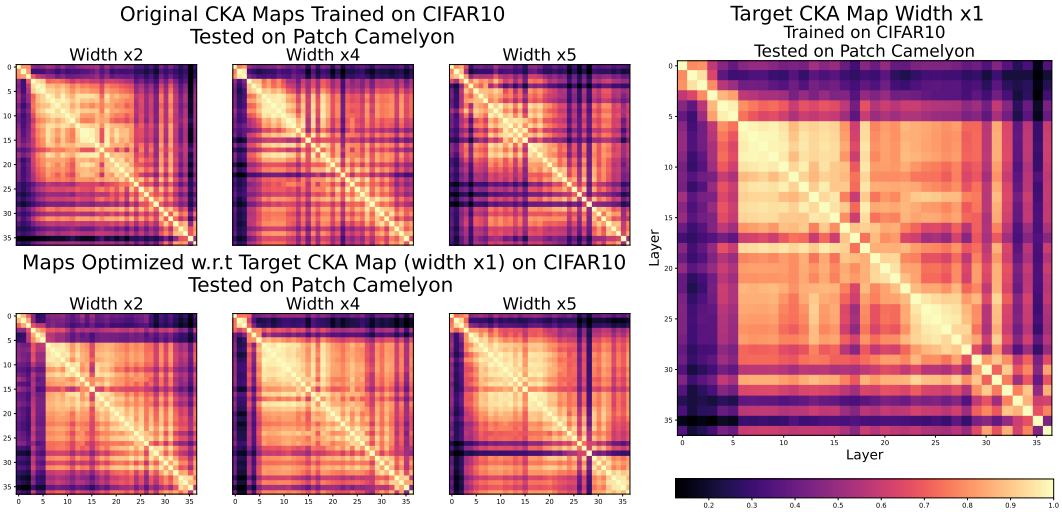

Figure 7: In Fig. 6 we are presented a series of ResNet-34 networks of different widths and their corresponding test set CKA Maps, which are modified to produce CKA maps of thin networks using the CIFAR10 dataset. We used these networks and measured their CKA maps using the test set of the Patch Camelyon dataset. Top row **Original** shows the unaltered CKA map of the networks derived from “normal" training on CIFAR10, tested on the test set of Patch Camelyon dataset. **Optimized** shows the CKA map of the networks after their map is optimized to mimic the thin network *target* CKA map using CIFAR10, tested on the test set Patch Camelyon dataset.

The network architecture presented so far is ResNet-34. We experimented with a VGG style network architecture to broaden our findings to other network architectures (see Sec. B.2 for details). As we can see in Fig. 8 we observe similar results to the ones shown in Fig. 6.

C.3 RBF Kernel

In Fig. 9 we extend our results shown in Fig. 2 to other bandwidth values commonly used for the RBF kernel CKA (Kornblith et al., 2019). When the CKA values are meaningful, we observe that the RBF kernel CKA values can be manipulated via the procedure described in Sec. 3.

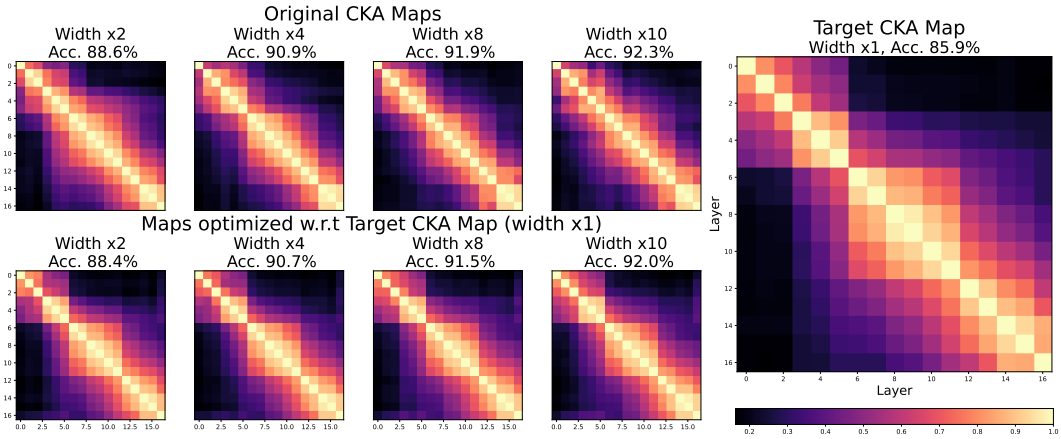

Figure 8: VGG style networks of different widths and their corresponding CKA Maps are modified to produce CKA maps of thin networks. Top row **Original** shows the unaltered test set CKA map of the networks derived from “normal" training. **Optimized** shows the test set CKA map of the networks after their map is optimized to mimic the thin net *target* CKA map.

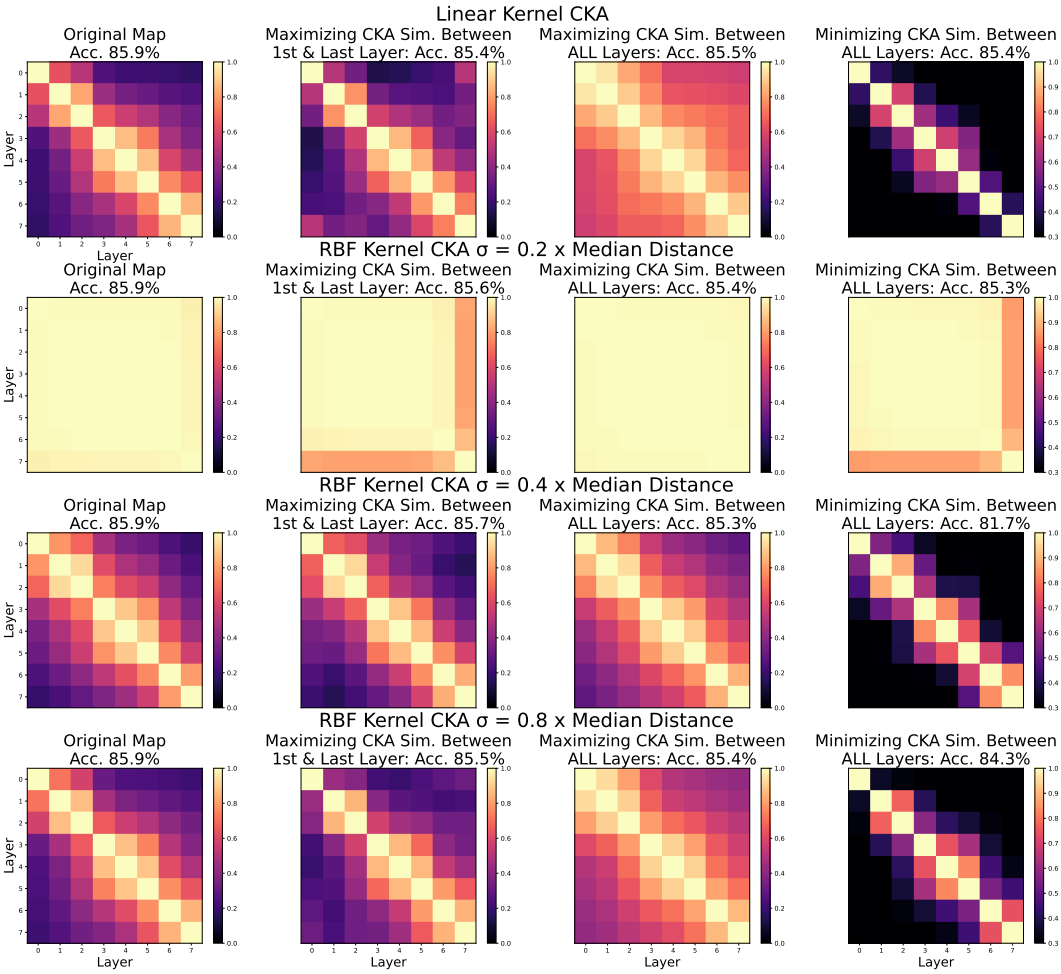

Figure 9: Original Map is the test set CKA map of a network trained on CIFAR10. We manipulate this network to produce CKA maps which: (1) maximizes the CKA similarity between the 1st and last layer, (2) maximizes the CKA similarity between all layers, and (3) minimizes the CKA similarity between all layers. In cases (1) and (2), the network experiences only a slight loss in performance, which counters previous findings by achieving a strong CKA similarity between early and late layers.

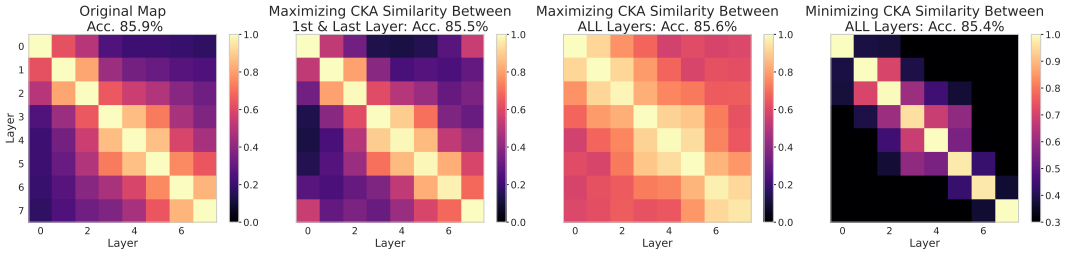

Figure 10: Original Map is the test set CKA map of a network trained on CIFAR10. We manipulate this network following a modified version of Eq. 2 (distillation loss is substituted with cross-entropy loss) to produce CKA maps which: (1) maximizes the CKA similarity between the 1st and last layer, (2) maximizes the CKA similarity between all layers, and (3) minimizes the CKA similarity between all layers. In cases (1) and (2), the network experiences only a slight loss in performance, which counters previous findings by achieving a strong CKA similarity between early and late layers.

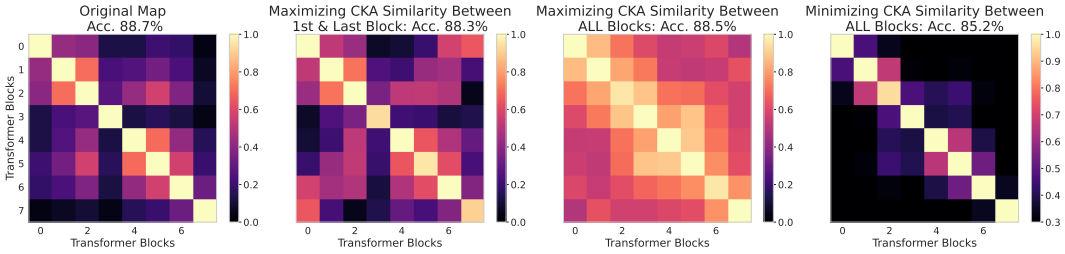

Figure 11: Original Map is the test set CKA map of a ViT (Dosovitskiy et al., 2020) network trained on CIFAR10. We manipulate this network following the Eq. 2 to produce CKA maps which: (1) maximizes the CKA similarity between the 1st and last Transformer block, (2) maximizes the CKA similarity between all Transformer blocks, and (3) minimizes the CKA similarity between all Transformer blocks.

C.4 CKA Map Optimization via Logistic Loss

In Sec. 3, we manipulated a network's CKA map, while closely maintaining its outputs via the distillation loss seen in Eq. 2. However, a logistic loss also works in this setting, i.e. the substitution of the distillation loss with cross-entropy loss in Eq. 2 yields similar results. In Fig. 10, we repeated the linear CKA experiments seen in the first row of the Fig. 2 using cross-entropy loss instead of distillation loss.

C.5 CKA Optimization of ViT

In Fig. 2, we manipulated the CKA map of a VGG style model trained on CIFAR10 in order to: (1) maximize the CKA similarity between the 1st and last layer, (2) maximize the CKA similarity between all layers, and (3) minimize the CKA similarity between all layers.

We further explored this setting at the model architecture level. Given the recent popularity of the Transformer (Vaswani et al., 2017) architecture in a variety of domains such as NLP Devlin et al. (2018); Farahnak et al. (2021); Raffel et al. (2020); Davari et al. (2020), Computer Vision Dosovitskiy et al. (2020); Zhou et al. (2021); Liu et al. (2021), and Tabular Huang et al. (2020); Arik & Pfister (2021), we implemented a Vision Transformer (ViT) (Dosovitskiy et al., 2020) style model for the CIFAR10 dataset, containing 8 Transformer (Vaswani et al., 2017) blocks (see other architectural details in Tab. 1) in order to: (1) maximize the CKA similarity between the 1st and last Transformer block, (2) maximize the CKA similarity between all Transformer blocks, and (3) minimize the CKA similarity between all Transformer blocks. As we can see in Fig. 11 these manipulations are achieved with minimal loss of performance, which underlines the model-agnostic nature of our approach.

# Transformer Blocks	# Attention Heads	Hidden Size	# Epochs
8	12	256	200

Table 1: Architectural details of out implementation of ViT (Dosovitskiy et al., 2020) for the CIFAR10 dataset. Note that the training process (except the number of epochs, which is listed above) follows the Sect. B.3.

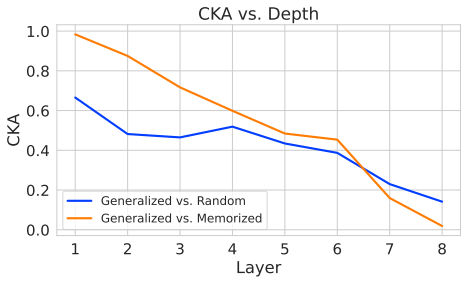

Figure 12: A layer-wise comparison based on the value of the CKA between a generalized, memorized, and randomly populated network. This comparison reveals that early layers of these networks achieve relatively high test set CKA values.

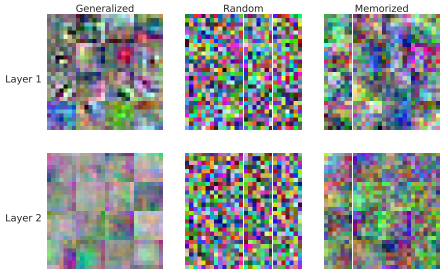

Figure 13: The convolution filters within the first two layers of a generalized, memorized, and a randomly initialized network elucidates that the features are (1) drastically different, and (2) not equally useful despite the CKA results in Fig. 12

C.6 Closer Look at The Early Layers

CKA values are often treated as a surrogate metric to measure the usefulness and similarity of a network's learned features when compared to another network (Ramasesh et al., 2021). In order to analyze this common assumption, we compare the features of: (1) a network trained to generalize on the CIFAR10 image classification task (Krizhevsky et al., 2009), (2) a network trained to "memorize" the CIFAR10 images (i.e. target labels are random), and (3) an untrained randomly initialized network (for network architecture and training details see the Appendix B.3). As show in Fig. 12, early layers of these networks should have very similar representations given the high test set CKA values. Under the previously presented assumption, one should therefore conclude that the learned features at these layers are relatively similar and equally valuable. However this is not the case, we can see in Fig. 13 that the convolution filters are drastically different across the three networks. Moreover, Fig. 13 elucidates that considerably high CKA similarity values for early layers, does not necessarily translate to more useful, or similar, captured features.

In Fig. 14, we can see a layer wise comparison between a generalized, memorized, and randomly populated network using either (Fig. 14-left) the same random seed or (Fig. 14-right) different random seeds. This comparison reveals that, in either case (with same or different random seeds) early layers of these networks achieve relatively high CKA values.

However, as it was shown in Fig. 13, high values of CKA similarity between two networks does not necessarily translate to more useful, or similar, captured features. In order to quantify the usefulness of the features captured by each network in Fig. 12 and 13, we follow the same methodology as used in Self-supervised Learning (Chen et al., 2020; Davari et al., 2022) and in the analysis of intermediate representations (Zeiler & Fergus, 2014b). We evaluate the adequacy of representations by an optimal linear classifier using training data from the original task, in this case the CIFAR10 training data. The test set accuracy obtained by the linear probe is used as a proxy to measure the usefulness of the representations. Fig. 15, shows the linear probe accuracy obtained on the CIFAR10 test set for the generalized, memorized, and randomly populated network seen in Fig. 12 and 13. The results shown in this figure along with the ones shown in Fig. 14 suggests that high values of CKA similarity between two networks does not necessarily translate to similarly useful features.

C.7 Analysis of Modified Representations

Our focus in Sec 3 was to use optimization to achieve a desired target CKA manipulations without any explicit specification of how to perform this manipulation. We perform an analysis to obtain insights

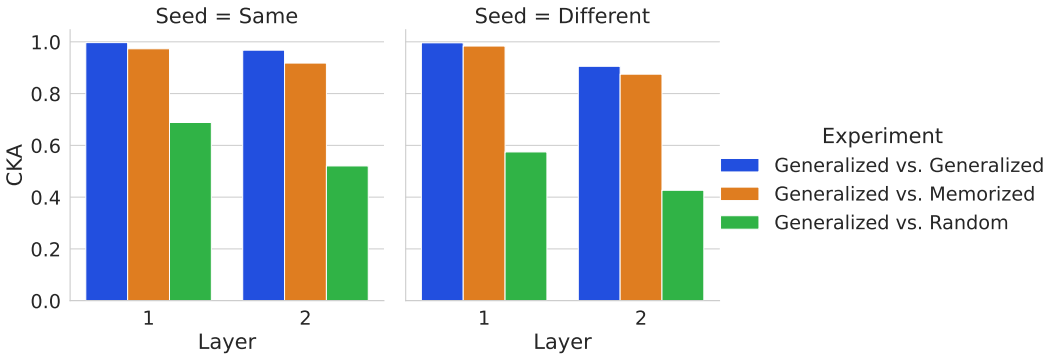

Figure 14: A layer wise comparison between a generalized, memorized, and randomly populated network using either (left figure) the same random seed or (right figure) different random seeds. This comparison reveals that, in either case (with same or different random seeds) early layers of these networks achieve relatively high CKA values.

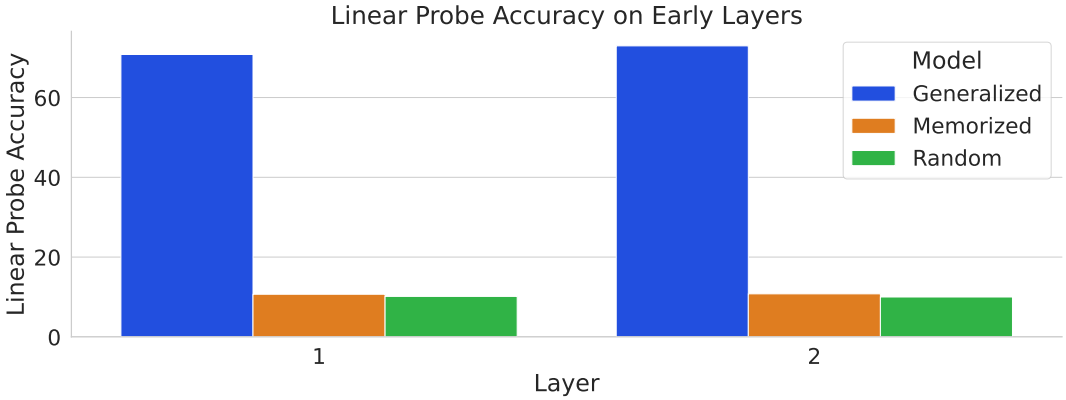

Figure 15: The linear probe accuracy obtained on the CIFAR10 test set for the generalized, memorized, and randomly populated network seen in Fig. 12 and 13. The results shown in this figure along with the ones shown in Fig. 14 suggests that high values of CKA similarity between two networks does not necessarily translate to similarly useful features.

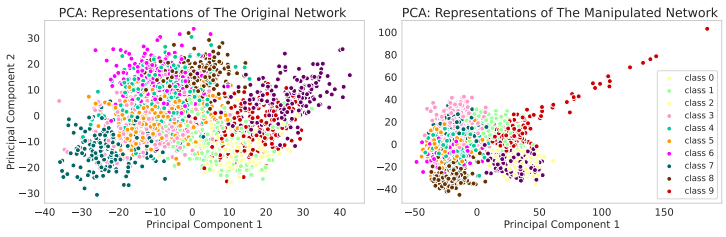

Figure 16: PCA of the networks presented in Fig. 2 before (left) and after (right) being optimized to manipulate the CKA map with Eq. 2. Noticeably to achieve the objective the optimization displaces a subset of a single class.

on how representations are changed by optimizing Eq. 2 in Fig. 16. Here using the modified network from case (2) of Fig. 2 we compute the PCA of the *test set*'s last hidden representation (whose CKA compared to the first layer is increased). We observe that a a single class has a very noticeable set of points that are translated in a particular direction, away from the general set of classes. This mechanism of manipulating the CKA aligns with our theoretical analysis. We emphasize, that in this case it is a completely emergent behavior.

D Proofs

Proof. **Theorem 1**
First we introduce the notation $C_1 = S$ and $C_2 = X \setminus S$ and note that C_1 and C_2 form a partition of X, i.e. $X = C_1 \sqcup C_2$ with $C_1 \cap C_2 = \emptyset$. We note C_j' the set of indices of C_j, meaning that $i \in C_j' \iff x_i \in C_j$. We then rewrite $X_{S,\vec{v},c}$ as being the union of the set of points in C_1 and the points in C_2 translated by c in direction $\vec{v}$:

$$X_{S,\vec{v},c} = \{x : x \in C_1\} \cup \{x + c\vec{v} : x \in C_2\}$$

It is standard practice to center the two sets of representations being compared before using a representation similarity measures. X is centered by hypothesis but $X_{S,\vec{v},c}$ is not. We first note that the mean of $X_{S,\vec{v},c}$ across all representations $\overline{X}_{S,\vec{v},c}$ is the vector:

$$\begin{aligned}
\overline{X}_{S,\vec{v},c} &= \frac{1}{n} \sum_{x \in X_{S,\vec{v},c}} x \\
&= \frac{1}{n} \left(\sum_{i \in C_1'} x_i + \sum_{i \in C_2'} x_i + c\vec{v} \right) && \text{by definition of } X_{S,\vec{v},c} \\
&= \frac{1}{n} \left(\sum_{i \in C_1' \cup C_2'} x_i + \sum_{i \in C_2'} c\vec{v} \right) \\
&= \frac{1}{n} \sum_{x \in X} x + \frac{1}{n} \sum_{i \in C_2'} c\vec{v} && \text{because } C_1, C_2 \text{ form a partition of } X \\
&= \frac{|C_2| c\vec{v}}{n} && \text{because } X \text{ is centered by hypothesis}
\end{aligned}$$

From now on we note Y as being the centered set of representations $X_{S,\vec{v},c}$ where we subtracted the mean $\overline{X}_{S,\vec{v},c}$ (here we used the fact that $|C_1| + |C_2| = n$):

$$Y = \{x - \frac{|C_2| c\vec{v}}{n} : x \in C_1\} \cup \{x + \frac{|C_1| c\vec{v}}{n} : x \in C_2\}$$

Now that we have workable expressions for X and $X_{S,\vec{v},c}$ we focus on the computation of linear CKA which relies on the computation of three HSIC values: between X and itself, between Y and itself and between X and Y:

$$\text{CKA}_{lin}(X, Y) = \frac{\text{HSIC}_{lin}(X, Y)}{\sqrt{\text{HSIC}_{lin}(X, X)\text{HSIC}_{lin}(Y, Y)}} \tag{5}$$

We also remind the reader that linear HSIC takes the form:

$$\text{HSIC}_{lin}(X, Y) = \frac{1}{(n-1)^2} tr(XX^\top YY^\top) = \frac{1}{(n-1)^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle x_i, x_j \rangle \langle y_j, y_i \rangle \tag{6}$$

We can split the terms of the two sums into three distinct categories and compute the values of the inner products independently in terms of x_i, x_j and c for the three HSIC terms:

1. $i \in C_1'$ and $j \in C_1'$ (i.e. $x_i \in C_1$ and $x_j \in C_1$):

$$
\begin{aligned}
\langle x_i, x_j\rangle^2 &= \langle x_i, x_j\rangle^2 \\
\langle x_i, x_j\rangle\langle y_i, y_j\rangle &= \langle x_i, x_j\rangle\langle x_i, \tfrac{cjC_2'j}{n}\sharp V, x_j\rangle \quad \tfrac{cjC_2'j}{n}\sharp V\rangle \\
&= \langle x_i, x_j\rangle\left[\langle x_i, x_j\rangle \quad \tfrac{cjC_2'j}{n}\langle x_i, \sharp V\rangle \quad \tfrac{cjC_2'j}{n}\langle \sharp V, x_j\rangle + \left(\tfrac{cjC_2'j}{n}\right)^2\langle \sharp V, \sharp V\rangle\right] \\
&= \langle x_i, x_j\rangle\left[\langle x_i, x_j\rangle \quad \tfrac{cjC_2'j}{n}\langle x_i, \sharp V\rangle \quad \tfrac{cjC_2'j}{n}\langle \sharp V, x_j\rangle + \tfrac{c^2jC_2'j^2}{n^2}\right] \\
&= \langle x_i, x_j\rangle^2 \quad \tfrac{cjC_2'j}{n}\langle x_i, \sharp V\rangle\langle x_i, x_j\rangle \quad \tfrac{cjC_2'j}{n}\langle \sharp V, x_j\rangle\langle x_i, x_j\rangle + \tfrac{c^2jC_2'j^2}{n^2}\langle x_i, x_j\rangle \\
&= O(c) + \tfrac{c^2jC_2'j^2}{n^2}\langle x_i, x_j\rangle \\
\langle y_i, y_j\rangle^2 &= \langle x_i \quad \tfrac{cjC_2'j}{n}\sharp V, x_j\rangle \quad \tfrac{cjC_2'j}{n}\sharp V\rangle^2 \\
&= \left[\langle x_i, x_j\rangle \quad \tfrac{cjC_2'j}{n}\langle x_i, \sharp V\rangle \quad \tfrac{cjC_2'j}{n}\langle \sharp V, x_j\rangle + \tfrac{c^2jC_2'j^2}{n^2}\right]^2 \\
&= O(c^3) + \tfrac{c^4jC_2'j^4}{n^4}
\end{aligned}
$$

2. $i \in C_1'$ and $j \in C_2'$ (i.e. $x_i \in C_1$ and $x_j \in C_2$):

$$
\begin{aligned}
\langle x_i, x_j\rangle^2 &= \langle x_i, x_j\rangle^2 \\
\langle x_i, x_j\rangle\langle y_i, y_j\rangle &= \langle x_i, x_j\rangle\langle x_i \quad \tfrac{cjC_2'j}{n}\sharp V, x_j\rangle + \tfrac{cjC_1'j}{n}\sharp V\rangle \\
&= \langle x_i, x_j\rangle\left[\langle x_i, x_j\rangle + \tfrac{cjC_1'j}{n}\langle x_i, \sharp V\rangle \quad \tfrac{cjC_2'j}{n}\langle \sharp V, x_j\rangle \quad \tfrac{c^2jC_1'jjC_2'j}{n^2}\langle \sharp V, \sharp V\rangle\right] \\
&= \langle x_i, x_j\rangle\left[\langle x_i, x_j\rangle + \tfrac{cjC_1'j}{n}\langle x_i, \sharp V\rangle \quad \tfrac{cjC_2'j}{n}\langle \sharp V, x_j\rangle \quad \tfrac{c^2jC_1'jjC_2'j}{n^2}\right] \\
&= \langle x_i, x_j\rangle^2 + \tfrac{cjC_1'j}{n}\langle x_i, x_j\rangle\langle x_i, \sharp V\rangle \quad \tfrac{cjC_2'j}{n}\langle x_i, x_j\rangle\langle \sharp V, x_j\rangle \quad \tfrac{c^2jC_1'jjC_2'j}{n^2}\langle x_i, x_j\rangle \\
&= O(c) \quad \tfrac{c^2jC_1'jjC_2'j}{n^2}\langle x_i, x_j\rangle \\
\langle y_i, y_j\rangle^2 &= \langle x_i \quad \tfrac{cjC_2'j}{n}\sharp V, x_j\rangle + \tfrac{cjC_1'j}{n}\sharp V\rangle^2 \\
&= \left[\langle x_i, x_j\rangle + \tfrac{cjC_1'j}{n}\langle x_i, \sharp V\rangle \quad \tfrac{cjC_2'j}{n}\langle \sharp V, x_j\rangle \quad \tfrac{c^2jC_1'jjC_2'j^2}{n^2}\right]^2 \\
&= O(c^3) + \tfrac{c^4jC_1'j^2jC_2'j^2}{n^4}
\end{aligned}
$$

3. $i \in C_2'$ and $j \in C_2'$ (i.e. $x_i \in C_2$ and $x_j \in C_2$):

$$
\begin{aligned}
\langle x_i, x_j \rangle^2 &= \langle x_i, x_j \rangle^2 \\
\langle x_i, x_j \rangle \langle y_i, y_j \rangle &= \langle x_i, x_j \rangle \langle x_i, \bar{v} \rangle \langle x_j, \bar{v} \rangle + \frac{cjC_1'j}{n} \langle \bar{v}, x_j \rangle + \frac{cjC_1'j}{n} \langle \bar{v}, \bar{v} \rangle \\
&= \langle x_i, x_j \rangle \left[\langle x_i, x_j \rangle + \frac{2cjC_1'j}{n} \langle x_i, \bar{v} \rangle + \left(+ \frac{cjC_1'j}{n} \right)^2 \langle \bar{v}, \bar{v} \rangle \right] \\
&= \langle x_i, x_j \rangle \left[\langle x_i, x_j \rangle + \frac{2cjC_1'j}{n} \langle x_i, \bar{v} \rangle + \frac{c^2 jC_1'j^2}{n^2} \right] \\
&= \langle x_i, x_j \rangle^2 + \frac{2cjC_1'j}{n} \langle x_i, x_j \rangle \langle x_i, \bar{v} \rangle + \frac{c^2 jC_1'j^2}{n^2} \langle x_i, x_j \rangle \\
&= O(c) + \frac{c^2 jC_1'j^2}{n^2} \langle x_i, x_j \rangle \\
\langle y_i, y_j \rangle^2 &= \langle x_i + \frac{cjC_1'j}{n} \bar{v}, x_j + \frac{cjC_1'j}{n} \bar{v} \rangle^2 \\
&= \left[\langle x_i, x_j \rangle + \frac{2cjC_1'j}{n} \langle x_i, \bar{v} \rangle + \frac{c^2 jC_1'j^2}{n^2} \right]^2 \\
&= O(c^3) + \frac{c^4 jC_1'j^4}{n^4}
\end{aligned}
$$

When we take $\lim_{c \to \infty} CKA_{lin}(X, Y) = \lim_{c \to \infty} \frac{\text{HSIC}_{lin}(X,Y)}{\sqrt{\text{HSIC}_{lin}(X,X)\text{HSIC}_{lin}(Y,Y)}}$, it is easy to see that the terms with the highest powers of c will dominate the expression. At the numerator that is c^2 and at the denominator that is c^4 inside the square root. To convince oneself of this it suffices to divide by c^2 at the numerator and at the denominator, all terms except the higher power ones will then tend to 0 as c tends to infinity, so at the limit we have:

$$
\begin{aligned}
&\lim_{c\to\infty} CKA_{lin}(X;Y) \\
&= \lim_{c\to\infty} \frac{\mathrm{HSIC}_{lin}(X;Y)}{\sqrt{\mathrm{HSIC}_{lin}(X;X)\mathrm{HSIC}_{lin}(Y;Y)}} \\
&= \lim_{c\to\infty} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} \langle x_i;x_j\rangle\langle y_j;y_i\rangle}{\sqrt{\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2\right)\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\langle y_i;y_j\rangle^2\right)}} \\
&= \lim_{c\to\infty} \frac{\frac{c^2|C_2^0|^2}{n^2}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i;x_j\rangle \quad \frac{2c^2|C_1^0||C_2^0|}{n^2}\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle + \frac{c^2|C_1^0|^2}{n^2}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2}\sqrt{\sum_{i\in C_1^0}\sum_{j\in C_1^0}\frac{c^4|C_2^0|^4}{n^4} + \sum_{i\in C_1^0}\sum_{j\in C_2^0}\frac{2c^4|C_1^0|^2|C_2^0|^2}{n^4} + \sum_{i\in C_2^0}\sum_{j\in C_2^0}\frac{c^4|C_1^0|^4}{n^4}}} \\
&= \lim_{c\to\infty} \frac{\frac{c^2|C_2^0|^2}{n^2}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i;x_j\rangle \quad \frac{2c^2|C_1^0||C_2^0|}{n^2}\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle + \frac{c^2|C_1^0|^2}{n^2}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2}\sqrt{\frac{c^4|C_2^0|^4|C_1^0|^2}{n^4} + \frac{2c^4|C_1^0|^3|C_2^0|^3}{n^4} + \frac{c^4|C_1^0|^4|C_2^0|^2}{n^4}}} \\
&= \lim_{c\to\infty} \frac{\frac{c^2|C_2^0|^2}{n^2}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i;x_j\rangle \quad \frac{2c^2|C_1^0||C_2^0|}{n^2}\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle + \frac{c^2|C_1^0|^2}{n^2}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2}\sqrt{\frac{c^4|C_2^0|^2C_1^0|^2}{n^4}(jC_2'j^2 + 2jC_1'jjC_2'j + jC_1'j^2)}} \\
&= \lim_{c\to\infty} \frac{\frac{c^2|C_2^0|^2}{n^2}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i;x_j\rangle \quad \frac{2c^2|C_1^0||C_2^0|}{n^2}\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle + \frac{c^2|C_1^0|^2}{n^2}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle}{\frac{c^2|C_2^0||C_1^0|}{n^2}\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2}\sqrt{jC_2'j^2 + 2jC_1'jjC_2'j + jC_1'j^2}} \\
&= \lim_{c\to\infty} \frac{\frac{|C_2^0|}{|C_1^0|}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i;x_j\rangle \quad 2\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle + \frac{|C_1^0|}{|C_2^0|}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2}\sqrt{jC_2'j^2 + 2jC_1'jjC_2'j + jC_1'j^2}} \\
&= \frac{\frac{|C_2^0|}{|C_1^0|}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i;x_j\rangle \quad 2\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle + \frac{|C_1^0|}{|C_2^0|}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2}\sqrt{jC_2'j^2 + 2jC_1'jjC_2'j + jC_1'j^2}} \\
&= \frac{\frac{|C_2^0|}{|C_1^0|}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i;x_j\rangle \quad 2\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle + \frac{|C_1^0|}{|C_2^0|}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2}\sqrt{(jC_2'j + jC_1'j)^2}} \\
&= \frac{\frac{|C_2^0|}{|C_1^0|}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i;x_j\rangle \quad 2\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle + \frac{|C_1^0|}{|C_2^0|}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2}\,p\overline{n^2}} \\
&= \frac{\frac{|C_2^0|}{|C_1^0|}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i;x_j\rangle \quad 2\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle + \frac{|C_1^0|}{|C_2^0|}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i;x_j\rangle}{n\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\langle x_i;x_j\rangle^2}}
\end{aligned}
$$

If we look directly at the numerator, by linearity of the inner product we have:

$$
\begin{aligned}
&\frac{|C_2'|}{|C_1'|}\sum_{i\in C_1^0}\sum_{j\in C_1^0}\langle x_i, x_j\rangle \quad 2\sum_{i\in C_1^0}\sum_{j\in C_2^0}\langle x_i, x_j\rangle + \frac{|C_1'|}{|C_2'|}\sum_{i\in C_2^0}\sum_{j\in C_2^0}\langle x_i, x_j\rangle \\
&= |C_1'||C_2'|\left\langle \frac{1}{|C_1'|}\sum_{i\in C_1^0} x_i, \frac{1}{|C_1'|}\sum_{j\in C_1^0} x_j\right\rangle \quad 2|C_1'||C_2'|\left\langle \frac{1}{|C_1'|}\sum_{i\in C_1^0} x_i, \frac{1}{|C_2'|}\sum_{i\in C_2^0} x_j\right\rangle \\
&\qquad + |C_1'||C_2'|\left\langle \frac{1}{|C_2'|}\sum_{i\in C_2^0} x_i, \frac{1}{|C_2'|}\sum_{j\in C_2^0} x_j\right\rangle \\
&= |C_2'||C_1'|\left[\langle \overline{x_1}, \overline{x_1}\rangle \quad 2\langle \overline{x_1}, \overline{x_2}\rangle + \langle \overline{x_2}, \overline{x_2}\rangle\right] \\
&= |C_2'||C_1'|\,\|\overline{x_1} \quad \overline{x_2}\|^2
\end{aligned}
$$

Where $\overline{x_j} = \mathbb{E}_{x\in C_j}[x] = \frac{1}{|C_j^0|}\sum_{i\in C_j^0} x_i$ is the mean of the points in C_j. At the denominator we can multiply by $\frac{n}{n}$ to obtain:

$$
\begin{aligned}
n\frac{n}{n}\sqrt{\sum_{x_i\in X}\sum_{x_j\in X}\langle x_i, x_j\rangle^2} &= n^2\sqrt{\sum_{x_i\in X}\sum_{x_j\in X}\frac{1}{n^2}\langle x_i, x_j\rangle^2} \\
&= n^2\sqrt{\sum_{i=1}^{n} \ ^2_i}
\end{aligned}
$$

We note that the term with the square root function is the Frobenius norm of the Gram matrix of the data (matrix of inner products) $XX^\top$ multiplied by $\frac{1}{n}$ which, in turn, is equal to the square root of the sum of it's squared eigenvalues, where $_i$ is the i-th eigenvalue of the matrix $\frac{1}{n}XX^\top$. However, through singular value decomposition, the Gram matrix (multiplied by $\frac{1}{n}$) has the same eigenvalues as the (biased) covariance matrix of the data, i.e. $\frac{1}{n}X^\top X$. Using the notation $X_{i,j}$ to go over the rows (representations) of X with i and over the columns (dimensions or neurons) of X with j we can write the variance in the data as the sum of the variances from all dimensions:

$$
\begin{aligned}
Var(X) &= \sum_{j=1}^{p} Var(X_{:,j}) \\
&= \sum_{j=1}^{p}\sum_{i=1}^{n}\frac{1}{n}(X_{i,j} \quad \overline{X_{:,j}})^2 \\
&= \frac{1}{n}\sum_{j=1}^{p}\sum_{i=1}^{n} X_{i,j}^2 \qquad\qquad \text{because the data is centered} \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{p} X_{i,j}^2 \\
&= \mathbb{E}_i[\|X_{i,:}\|^2] \\
&= \mathbb{E}_{x\in X}[\|x\|^2] \\
&= \sum_{l=1}^{p} \ _i
\end{aligned}
$$

We can then write the denominator as:

$$
\begin{aligned}
n^2\sqrt{\sum_{i=1}^{n} \ {}_i^2} &= n^2\frac{Var(X)}{Var(X)}\sqrt{\sum_{i=1}^{n} \ {}_i^2} \\
&= n^2 Var(X)\frac{\sqrt{\sum_{i=1}^{n} \ {}_i^2}}{Var(X)} \\
&= n^2 \mathsf{E}_{x\in X}[kxk^2]\frac{\sqrt{\sum_{i=1}^{n} \ {}_i^2}}{\sum_{i=1}^{n} \ {}_i} \\
&= n^2 \mathsf{E}_{x\in X}[kxk^2] PR(X)^{-1/2}
\end{aligned}
$$

And we can rewrite the whole expression as:

$$
\lim_{c\to\infty} CKA_{lin}(X;Y) = \frac{jC_1' jjC_2' jk\overline{x_1} \quad \overline{x_2}k^2\sqrt{PR(X)}}{n^2\mathsf{E}_{x\in X}[kxk^2]}
$$

Where $PR(X)$ is the participation ratio, an effective dimensionality estimator often used in the literature and is defined as:

$$
PR(X) = \frac{\left(\sum_{i=1}^{p} \ {}_i\right)^2}{\sum_{i=1}^{p} \ {}_i^2}
$$

With ${}_i$ being the i-th eigenvalue of the covariance matrix of the data X. We make the replacements $\frac{|C_1|}{n} = \frac{|S|}{n} =$ and $\frac{|C_2|}{n} = \frac{n-|C_1|}{n} = 1$. Also, because the data is centered we have $jC_1j\overline{x_1} + jC_2j\overline{x_2} = 0$ and we can isolate $\overline{x_2} = \frac{-|C_1|\overline{x_1}}{|C_2|}$ so we have:

$$
\begin{aligned}
k\overline{x_1} \quad \overline{x_2}k^2 &= k\overline{x_1} + \frac{jC_1j\overline{x_1}}{jC_2j}k^2 \\
&= (1 + \frac{jC_1j}{jC_2j})^2 k\overline{x_1}k^2 \\
&= (1 + \frac{njC_1j}{njC_2j})^2 k\overline{x_1}k^2 \\
&= (1 + \frac{}{1})^2 k\overline{x_1}k^2
\end{aligned}
$$

We then define to contain all terms of :

$$
\begin{aligned}
(p) &= (1 \quad)(1 + \frac{}{1})^2 \\
&= \frac{}{1}
\end{aligned}
$$

The following bounds hold: ( ) 2 $(0;1]$ for 2 $(0;0.5]$ reached when $= \frac{1}{2}$. Finally, we get the final expression by changing $\overline{x_1} = \mathsf{E}_{x\in C_1}[x] = \mathsf{E}_{x\in S}[x]$:

$$
\lim_{c\to\infty} CKA_{lin}(X;Y) = \ ( )\frac{k\mathsf{E}_{x\in S}[x]k^2}{\mathsf{E}_{x\in X}[kxk^2]}\sqrt{PR(X)}
$$

□

Proof. **Corollary 2**
To prove Corollary 2 it suffices to note that the fact that $= \frac{|S|}{|X|}$ is in $(0; \frac{1}{2}]$ is not used anywhere in the proof of Thm. 1 other than to derive the bounds for $( )$. We can then conclude that Thm. 1 still holds if S is taken such that $= \frac{|S|}{|X|}$ 2 $(0.5; 1)$ only with different bounds for $( )$.

We note however that the expression in Thm. 1 can be written in terms of S and or in terms of $S' = X n S$ and $' = \frac{|S^0|}{|X|}$ interchangeably. We first note that $' = 1$ and for simplicity purposes we will use the notation $\overline{S} = \mathsf{E}_{x \in S}[x]$ and $\overline{S'} = \mathsf{E}_{x \in S^0}[x]$. We recall that the expression in Thm. 1 is:

$$\lim_{c \to \infty} CKA_{lin}(X; X_{S, \sharp v, c}) = ( ) \frac{k\overline{S}k^2}{\mathsf{E}_{x \in X}[kxk^2]} \sqrt{PR(X)}$$

The term $\frac{p}{\frac{PR(X)}{\mathsf{E}_{x\ 2\ X}[\|x\|^2]}}$ does not depend on the choice of using S or $X n S$ so we can focus on the rest:

$$(')k\overline{S'}k^2 = '(1 \quad ')(1 + \frac{'}{1 \quad '})^2 k\overline{S'}k^2$$

$'(1 \quad ')$ is easily found to be equal to $(1 \quad)$ and for the rest we have:

$$\begin{aligned}
(1 + \frac{'}{1 \quad '})^2 k\overline{S'}k^2 &= k(1 + \frac{'}{1 \quad '})\overline{S'}k^2 \\
&= k\overline{S'} \quad \overline{S}k^2 \\
&= k\overline{S} \quad \overline{S'}k^2 \\
&= k\overline{S} + \frac{}{1 \quad }\overline{S}k^2 \\
&= (1 + \frac{}{1 \quad })^2 k\overline{S}k^2
\end{aligned}$$

Where we used the fact that the data is centered so $jSj\overline{S} + jS'j\overline{S'} = 0$ so we can isolate $S = \frac{-|S^0|\overline{S^0}}{|S|}$. We conclude that we have:

$$\lim_{c \to \infty} CKA_{lin}(X; X_{S, \sharp v, c}) = (') \frac{k\overline{S'}k^2}{\mathsf{E}_{x \in X}[kxk^2]} \sqrt{PR(X)}$$

□

Proof. **Corollary 4**
We already have $x\ 2\ S\ )\ hw; xi \quad k$. Pick any $\sharp v\ 2\ \mathsf{R}^p$ that is orthogonal to w and we have:

$$\begin{aligned}
hw; x + c\sharp v i &= hw; xi + chw; \sharp v i && \text{by the linearity of the inner product} \\
&= hw; xi && \text{since } \sharp v \text{ is orthogonal to } w \\
&> k && 8x\ 2\ X n S
\end{aligned}$$

□