Similarity of Neural Network Representations Revisited

Simon Kornblith, Mohammad Norouzi, Honglak Lee & Geoffrey Hinton
Google Brain
{skornblith,mnorouzi,honglak,geoffhinton}@google.com

Abstract

Recent work has sought to understand the behavior of neural networks by comparing representations between layers and between different trained models. We introduce a similarity index that measures the relationship between representational similarity matrices. We show that this similarity index is equivalent to centered kernel alignment (CKA) and analyze its relationship to canonical correlation analysis. Unlike other methods, CKA can reliably identify correspondences between representations of layers in networks trained from different initializations. Moreover, CKA can reveal network pathology that is not evident from test accuracy alone.

1 Introduction

Despite impressive empirical advances in deep neural networks for solving various tasks, the problem of understanding and characterizing the learned representations remains relatively underexplored. This work investigates the problem of measuring similarities between deep neural network representations. We build upon previous studies investigating similarity between the representations of different neural networks (Li et al., 2015; Raghu et al., 2017; Morcos et al., 2018; Wang et al., 2018) and get inspiration from extensive neuroscience literature that uses representational similarity to compare representations across brain areas (Freiwald & Tsao, 2010), individuals (Connolly et al., 2012), species (Kriegeskorte et al., 2008), and behaviors (Elsayed et al., 2016), as well as between brains and neural networks (Yamins et al., 2014; Khaligh-Razavi & Kriegeskorte, 2014). Our key contributions are summarized as follows:

- We motivate and introduce centered kernel alignment (CKA) as a similarity index.
- We analyze the relationship between CKA and canonical correlation analysis (CCA).
- We show that CKA is able to determine the correspondence between the hidden layers of neural networks trained from different random initializations and with different depths, scenarios where existing similarity indexes do not work well.
- We demonstrate the utility of CKA for discovering training pathology and understanding relationships among neural network architectures.

Problem Statement. We begin by defining the problem of measuring representational similarity of deep neural networks. Let $X \in \mathbb{R}^{n\times p_1}$ denote a matrix of activations of p_1 neurons for n examples, and $Y \in \mathbb{R}^{n\times p_2}$ denote a matrix of activations of p_2 neurons for the same examples. We assume that these matrices are centered , and without loss of generality that $p_1 \leq p_2$. We are concerned with the design and analysis of a scalar similarity index $s(X, Y)$ that can be used to compare representations.

2 Comparing Similarity Structures

Our key insight is that instead of comparing multivariate features of an example in the two representations (*e.g.* via regression), one can first measure the similarity between every pair of *examples* in each representation separately, and then compare the similarity structures. We show below that, if we use an inner product to measure similarity, the similarity between representational similarity matrices reduces to another intuitive notion of pairwise feature similarity.

Dot Product-Based Similarity. A simple formula relates dot products between examples to dot products between features:

$$\langle \text{vec}(XX^\text{T}), \text{vec}(YY^\text{T})\rangle = \text{tr}(XX^\text{T}YY^\text{T}) = ||Y^\text{T}X||_\text{F}^2 \tag{1}$$

The elements of XX^T and YY^T are dot products between the representations of the i^th and j^th examples, and indicate the similarity between these examples according to the respective networks. The left-hand side of equation 1 thus measures the similarity between the inter-example similarity structures. The right-hand side shows that the same result is obtained by measuring the similarity between *features* from X and Y, by summing the squared dot products between every pair.

Hilbert-Schmidt Independence Criterion. Because we assume that X and Y are centered, equation 1 is proportional to the squared Frobenius norm of the cross-covariance matrix between the features of X and Y. The Hilbert-Schmidt Independence Criterion (Gretton et al., 2005) generalizes this index to inner products from reproducing kernel Hilbert spaces. Let $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and let $L_{ij} = l(\mathbf{y}_i, \mathbf{y}_j)$ where k and l are kernel functions. The empirical estimator of HSIC is:

$$\text{HSIC}(K, L) = \frac{1}{(n-1)^2}\text{tr}(KHLH) \tag{2}$$

where H is the centering matrix $H_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^\text{T}$. For a linear kernel between centered representations, $\text{HSIC}(XX^\text{T}, YY^\text{T}) = \text{tr}(XX^\text{T}YY^\text{T})/(n-1)^2$.

Centered Kernel Alignment. HSIC is not invariant to isotropic scaling of features. A normalized form, known as centered kernel alignment (Cortes et al., 2012; Cristianini et al., 2002), can be obtained as:

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

For a linear kernel, CKA is equivalent to the RV coefficient (Robert & Escoufier, 1976) and to Tucker's congruence coefficient (Tucker, 1951; Lorenzo-Seva & Ten Berge, 2006).

Below, we report results of CKA with a linear kernel and the RBF kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||_2^2/(2\sigma^2))$. For the RBF kernel, there are several possible strategies for selecting the bandwidth σ, which controls the extent to which similarity of small distances is emphasized over large distances. We set σ as a fraction of the median distance between examples. For CNNs, we find that RBF kernels and linear kernels give similar results, so we use linear CKA unless otherwise specified. However, the framework extends to any valid kernel, including kernels equivalent to neural networks (Lee et al., 2018; Jacot et al., 2018; Garriga-Alonso et al., 2018; Novak et al., 2019).

3 CKA VERSUS CCA

Rewriting linear CKA in terms of the eigenvectors and eigenvalues of XX^T and YY^T, *i.e.* the principal components of X and Y and the amount of variance that they explain, provides some intuition regarding its relationship with canonical correlation analysis. R^2_CCA, the sum of the squared canonical correlations, is given by the normalized sum of the squared dot products between eigenvectors. CKA resembles R^2_CCA, but with the contributions of each pair of eigenvectors weighted by their corresponding eigenvalues. Let the i^th eigenvector of XX^T be indexed as $\mathbf{u}_X^\text{i}$ and corresponding eigenvalue as λ_X^i. Then:

$$R^2_\text{CCA} = ||U_Y^\text{T}U_X||_\text{F}^2/p_1 = \sum_{i=1}^{p_1}\sum_{j=1}^{p_2}\langle \mathbf{u}_X^i, \mathbf{u}_Y^j\rangle^2/p_1 \tag{4}$$

$$\text{CKA}(XX^\text{T}, YY^\text{T}) = \frac{||Y^\text{T}X||_\text{F}^2}{||X^\text{T}X||_\text{F}||Y^\text{T}Y||_\text{F}} = \frac{\sum_{i=1}^{p_1}\sum_{j=1}^{p_2}\lambda_X^i\lambda_Y^j\langle \mathbf{u}_X^i, \mathbf{u}_Y^j\rangle^2}{\sqrt{\sum_{i=1}^{p_1}(\lambda_X^i)^2}\sqrt{\sum_{j=1}^{p_2}(\lambda_Y^j)^2}} \tag{5}$$

CCA weights all eigenvectors equally, and is thus invariant to *any* invertible linear transformation of the features of X and Y. By contrast, CKA places greater weight on eigenvectors that explain

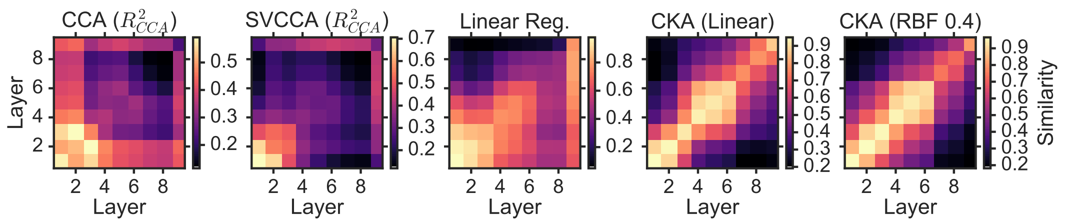

Figure 1: CKA finds correspondences between layers of CNNs trained with different random initializations, whereas other methods do not. Additional SVCCA truncation thresholds are shown in Appendix Figure C.4. For linear regression, we plot R^2 for the fit of the layer on the x-axis with the layer on the y-axis. Results are averaged over 45 pairs of networks. See also Table 1.

more variance, and is thus invariant only to orthogonal transformations, which preserve Euclidean distances between examples in feature space.

We argue that invariance to orthogonal transformation, but not arbitrary linear transformation, is appropriate for measuring similarity in neural networks. Neural networks learn feature spaces where distances are *meaningful*, as demonstrated by the success of perceptual loss and style transfer (Gatys et al., 2016; Johnson et al., 2016; Dumoulin et al., 2017). In the linear case, it can be shown that dynamics of gradient descent are invariant to orthogonal transformation but not invertible linear transformation (LeCun et al., 1991), and in the case of overparameterization or early stopping, the solution reached is scale-dependent. Invariance to invertible linear transformation also presents practical problems for measuring similarity between wide layers; any similarity index that is invariant to invertible linear transformation gives the same result for any representation of width greater than or equal to the dataset size. We discuss invariance properties and their implications further in Appendix A.

Because CKA can be computed efficiently, it is potentially useful as a tool for practitioners. Whereas computing CCA requires matrix decompositions, CKA requires only inner products. There is an unbiased estimator of HSIC (Song et al., 2007) that permits estimation of CKA based on accumulation of scalar minibatch statistics. Thus, CKA can be integrated into model evaluation pipelines without incurring excessive computational cost.

4 RESULTS

4.1 A SANITY CHECK FOR SIMILARITY INDEXES

We propose a simple sanity check for similarity indexes: Given two architecturally identical networks trained from different random initializations, for each layer in the first network, the most similar layer in the second network should be the architecturally corresponding layer. We compare CKA to CCA, singular vector CCA (SVCCA) (Raghu et al., 2017), projection-weighted CCA (PWCCA) (Morcos et al., 2018), and linear regression; see Appendix B for more details. For CCA and SVCCA, we compute similarity as either the sum of the canonical correlations ($\bar{\rho}$) as in Raghu et al. (2017), or as the sum of the squared canonical correlations (R^2_{CCA}).

We first investigate a simple VGG-like convolutional neural network based on All-CNN-C (Springenberg et al., 2014). We provide architecture details in Appendix D. We show results in Figure 1 and Table 1. CKA passes our sanity check: Across $\binom{10}{2} = 45$ pairs of models, a given layer had greater CKA similarity with the corresponding layer in the other model than with any other layer for 99.3% of pairs. Other methods performed substantially worse.

Index	Accuracy
CCA ($\bar{\rho}$)	1.4
CCA (R^2_{CCA})	10.6
SVCCA ($\bar{\rho}$)	9.9
SVCCA (R^2_{CCA})	15.1
PWCCA	11.1
Linear Reg.	45.4
CKA (Linear)	**99.3**
CKA (RBF 0.2)	80.6
CKA (RBF 0.4)	**99.1**
CKA (RBF 0.8)	**99.3**

Table 1: Accuracy of identifying corresponding layers based maximum similarity, for 45 pairs of architecturally identical 10-layer CNNs trained from different initializations.

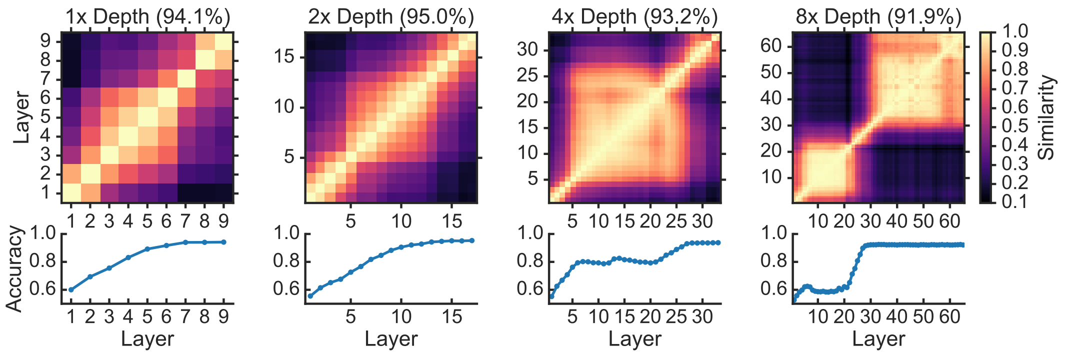

Figure 2: CKA reveals when depth becomes pathological. **Top**: Linear CKA between layers of individual networks of different depths on CIFAR-10. Titles show accuracy of each network. Later layers of the 8x depth network are similar to the last layer. **Bottom**: Accuracy of a logistic regression classifier trained on layers of the same networks is consistent with CKA.

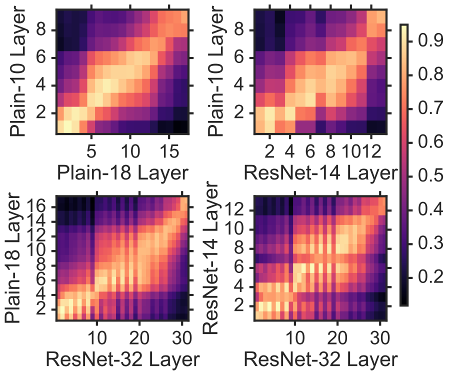

Figure 3: Linear CKA between layers of networks with different architectures.

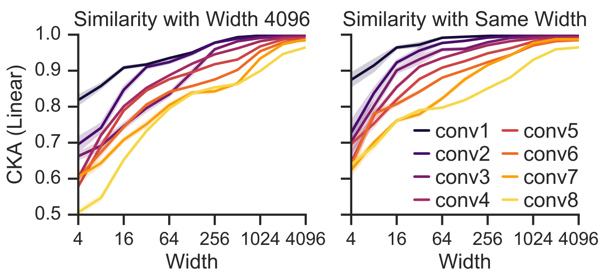

Figure 4: Layers become more similar to each other and to wide models as width increases; similarity of early layers saturates first. **Left**: Similarity with the widest network. **Right**: Similarity with other networks of the same width trained from random initialization.

Both CKA and CCA-based methods give reasonable results for Transformer networks, where all layers are of equal width, as shown in Appendix C.1. However, in most settings, RBF CKA was more accurate at identifying correspondences between Transformer layers than other methods.

4.2 CKA Reveals Network Pathology

In Figure 2, we show CKA between layers of similar CNNs with different depths, where we repeat every layer 2, 4, or 8 times to deepen the network. Doubling depth improves accuracy, but further increasing of depth (4x and 8x) hurts accuracy. At 8x depth, CKA indicates that representations of more than half of the network are very similar to the last layer. We validated that these later layers do not refine the representation by training an ℓ^2-regularized logistic regression classifier on each layer of the network. Shallower architectures show a progressive improvement in classification accuracy, but accuracy of the 8x deeper network plateaus.

4.3 Using CKA to Understand Network Architectures

CKA is equally effective at measuring relationships between different architectures. Figure 3 shows the relationship between different layers of plain CNNs and ResNets. CKA indicates that, as networks are made deeper, the new layers are effectively inserted in between old layers. Other similarity indexes fail to reveal meaningful relationships between different architectures, as we show in Appendix C.5.

In Figure 4, we show CKA between networks with different layer widths. Like Morcos et al. (2018), we find that increasing layer width leads to more similar representations between networks. As width increases, CKA approaches 1 and CKA of earlier layers saturates faster than later layers. Networks are generally more similar to other networks of the same width than they are to the widest network.

4.4 Similar Representations Across Datasets

CKA can also be used to compare networks trained on different datasets. In Figure 5, we show that models trained on CIFAR-10 and CIFAR-100 develop similar representations in their early layers. These representations require training; similarity with untrained networks is much lower. We further explore similarity between layers of untrained networks in Appendix C.6.

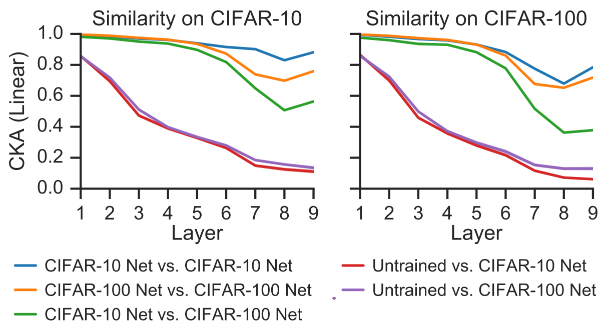

Figure 5: Models trained on different datasets develop similar representations in early layers; these representations differ from untrained models. **Left:** Linear CKA between the same layer of different models on the CIFAR-10 test set. **Right:** CKA on CIFAR-100 test set. CKA is averaged over 10 models of each type.

5 Conclusion

Measuring similarity between the representations learned by neural networks is an ill-defined problem, since it is not entirely clear what aspects of the representation a similarity index should focus on. We have shown that, unlike previously proposed similarity indexes, CKA captures intuitive notions of similarity, *i.e.* that neural networks trained from different initializations should be similar to each other. CKA also has practical utility for understanding network pathology: By inspecting CKA between layers, we can explain what happens when a CNN becomes “too deep” in terms of the learned representation.

Acknowledgements

We thank Gamaleldin Elsayed, Jaehoon Lee, Maithra Raghu, Samuel L. Smith, and Alex Williams for comments on the manuscript, Rishabh Agarwal for ideas, and Aliza Elkin for support.

References

An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural Computation*, 5(6):910–927, 1993.

Andrew C Connolly, J Swaroop Guntupalli, Jason Gors, Michael Hanke, Yaroslav O Halchenko, Yu-Chien Wu, Hervé Abdi, and James V Haxby. The representation of biological classes in the human brain. *Journal of Neuroscience*, 32(8):2608–2618, 2012.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(Mar):795–828, 2012.

Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz S Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems*, pp. 367–373, 2002.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *International Conference on Learning Representations*, 2, 2017.

Gamaleldin F Elsayed, Antonio H Lara, Matthew T Kaufman, Mark M Churchland, and John P Cunningham. Reorganization between preparatory and movement population responses in motor cortex. *Nature communications*, 7:13239, 2016.

Winrich A Freiwald and Doris Y Tsao. Functional compartmentalization and viewpoint generalization within the macaque face-processing system. *Science*, 330(6005):845–851, 2010.

Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow gaussian processes. *arXiv preprint arXiv:1808.05587*, 2018.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2414–2423, 2016.

Gene H Golub and Hongyuan Zha. The canonical correlations of matrix pairs and their numerical computation. In *Linear Algebra for Signal Processing*, pp. 27–49. Springer, 1995.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pp. 63–77. Springer, 2005.

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, 2015.

Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pp. 694–711. Springer, 2016.

Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS computational biology*, 10(11):e1003915, 2014.

Nikolaus Kriegeskorte, Marieke Mur, Douglas A Ruff, Roozbeh Kiani, Jerzy Bodurka, Hossein Esteky, Keiji Tanaka, and Peter A Bandettini. Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, 60(6):1126–1141, 2008.

Malte Kuss and Thore Graepel. The geometry of kernel canonical correlation analysis. Technical report, Max Planck Institute for Biological Cybernetics, 2003.

Yann LeCun, Ido Kanter, and Sara A Solla. Second order properties of error surfaces: Learning time and generalization. In *Advances in Neural Information Processing Systems*, pp. 918–924, 1991.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

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.

Urbano Lorenzo-Seva and Jos MF Ten Berge. Tucker's congruence coefficient as a meaningful index of factor similarity. *Methodology*, 2(2):57–64, 2006.

Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems 31*, pp. 5732–5741, 2018.

Youssef Mroueh, Etienne Marcheret, and Vaibhava Goel. Asymmetrically weighted cca and hierarchical kernel sentence embedding for multimodal retrieval. *arXiv preprint arXiv:1511.06267*, 2015.

Roman Novak, Lechao Xiao, Yasaman Bahri, Jaehoon Lee, Greg Yang, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019.

Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. In *International Conference on Learning Representations*, 2018.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl- Dickstein. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *arXiv e-prints*, art. arXiv:1706.05806, June 2017.

J.O. Ramsay, Jos ten Berge, and G.P.H. Styan. Matrix correlation. *Psychometrika*, 49(3):403–423, 1984.

Paul Robert and Yves Escoufier. A unifying tool for linear multivariate statistical methods: the RV-coefficient. *Applied Statistics*, pp. 257–265, 1976.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

SAS Institute. *Introduction to Regression Procedures*. 2015. URL `https://support.sas.com/documentation/onlinedoc/stat/141/introreg.pdf`.

Le Song, Alex Smola, Arthur Gretton, Karsten M Borgwardt, and Justin Bedo. Supervised feature selection via dependence estimation. In *Proceedings of the 24th international conference on Machine learning*, pp. 823–830. ACM, 2007.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

StataCorp. *Stata Multivariate Statistics Reference Manual*. 2015. URL `https://www.stata.com/manuals14/mv.pdf`.

Ledyard R Tucker. A method for synthesis of factor analysis studies. Technical report, Educational Testing Service, Princeton, NJ, 1951.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Liwei Wang, Lunjia Hu, Jiayuan Gu, Yue Wu, Zhiqiang Hu, Kun He, and John E. Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. In *Advances in Neural Information Processing Systems*, pp. 9607–9616, 2018.

Daniel LK Yamins, Ha Hong, Charles F Cadieu, Ethan A Solomon, Darren Seibert, and James J DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.

APPENDIX

This appendix contains additional theoretical and experimental results not discussed in the main text of the workshop paper. Section A motivates the use of a similarity index that is invariant to orthogonal transformation and isotropic scaling, but not to invertible linear transformation, as a tool for measuring similarity between neural network representations. Section B introduces the similarity indexes that we compare against, as well as other previously proposed similarity indexes. Section C describes additional experiments, including sanity check results for the Transformer architecture, an analysis of the shared subspace of neural networks that provides further insight into what CKA measures, and similarity results under a variety of settings not explored in the main text. Finally, Section D describes the architectures used for experiments.

A INVARIANCE PROPERTIES OF SIMILARITY INDEXES AND THEIR IMPLICATIONS

In this section, we argue that both intuitive notions of similarity and the dynamics of neural network training call for a similarity index that is invariant to orthogonal transformation and isotropic scaling, but not invertible linear transformation.

A.1 INVARIANCE TO INVERTIBLE LINEAR TRANSFORMATION

A similarity index is invariant to invertible linear transformation if $s(X, Y) = s(XA, YB)$ for any full rank A and B. If activations X are followed by a fully-connected layer $f(X) = \sigma(XW + \beta)$, then transforming the activations by a full rank matrix A as $X' = XA$ and transforming the weights by the matrix inverse A^{-1} as $W' = A^{-1}W$ preserves the output of $f(X)$. This transformation does not appear to change how the network operates, so intuitively, one might prefer a similarity index that is invariant to any invertible linear transformation, as argued by Raghu et al. (2017).

However, a key limitation of invariance to invertible linear transformation is that any invariant similarity index gives the same result for any representation of width greater than or equal to the dataset size, *i.e.* $p_2 \geq n$.

Theorem A.1. *Let X and Y be $n \times p$ matrices. Suppose s is invariant to invertible linear transformation in the first argument,* i.e. *$s(X, Z) = s(XA, Z)$ for arbitrary Z and any A with rank$(A) = p$. If rank$(X) =$ rank$(Y) = n$, then $s(X, Z) = s(Y, Z)$.*

Proof. Let

$$X' = \begin{bmatrix} X \\ K_X \end{bmatrix} \qquad\qquad Y' = \begin{bmatrix} Y \\ K_Y \end{bmatrix}$$

where K_X is a basis for the null space of the rows of X and K_Y is a basis for the null space of the rows of Y. Then let $A = X'^{-1}Y'$.

$$\begin{bmatrix} X \\ K_X \end{bmatrix} A = \begin{bmatrix} Y \\ K_Y \end{bmatrix} \implies XA = Y$$

Because X' and Y' have rank p by construction, A exists and has rank p. Thus, $s(X, Z) = s(XA, Z) = s(Y, Z)$. □

There is thus a practical problem with invariance to invertible linear transformation: Some neural networks, especially convolutional networks, have more neurons in some layers than there are examples the training dataset. It is somewhat unnatural that a similarity index could require more examples than were used to train the network.

A deeper issue is that neural network *training* is not invariant to arbitrary invertible linear transformation of inputs or activations. Even in the linear case, gradient descent converges first along the eigenvectors corresponding to the largest eigenvalues of the input covariance matrix (LeCun et al., 1991), and in cases of overparameterization or early stopping, the solution reached depends on the scaling of the input. Similar results holds for gradient descent training of neural networks in the

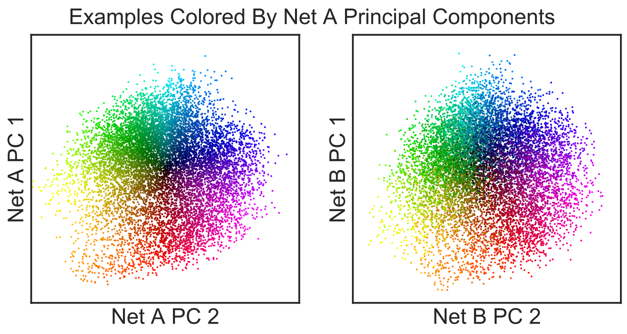

Figure A.1: First principal components of representations of networks trained from different random initializations are similar. Each example from the CIFAR-10 test set is shown as a dot colored according to the value of the first two principal components of an intermediate layer of one network (left) and plotted on the first two principal components of the same layer of an architecturally identical network trained from a different initialization (right).

infinite width limit (Jacot et al., 2018). The importance of scaling in neural networks is further demonstrated by the popularity of batch normalization (Ioffe & Szegedy, 2015).

Invariance to invertible linear transformation implies that the scale of directions in activation space is irrelevant. Empirically, however, scale information is both consistent across networks and useful across tasks. Neural networks trained from different random initializations develop representations with similar large principal components, as shown in Figure A.1. Consequentially, Euclidean distances between examples, which depend primarily upon large principal components, are similar across networks. These distances are *meaningful*, as demonstrated by the success of perceptual loss and style transfer (Gatys et al., 2016; Johnson et al., 2016; Dumoulin et al., 2017). A similarity index that is invariant to invertible linear transformation ignores this aspect of the representation, and assigns the same score to networks that match only in large principal components as to networks that match only in small principal components.

A.2 Invariance to Orthogonal Transformation

Rather than requiring invariance to any invertible linear transformation, one could require a weaker condition, invariance to orthogonal transformation, *i.e.* $s(X, Y) = s(XU, YV)$ for full-rank orthonormal matrices U and V such that $U^\text{T}U = I$ and $V^\text{T}V = I$.

Indexes invariant to orthogonal transformations do not share the limitations of indexes invariant to invertible linear transformation. When $p_2 > n$, indexes invariant to orthogonal transformation remain well-defined. Moreover, orthogonal transformations preserve scalar products and Euclidean distances between examples.

Invariance to orthogonal transformation is desirable for neural networks trained by gradient descent. Invariance to orthogonal transformation implies invariance to permutation, which is needed to accommodate symmetries of neural networks (Chen et al., 1993; Orhan & Pitkow, 2018). In the linear case, orthogonal transformation of the input does not affect the dynamics of gradient descent training (LeCun et al., 1991), and for networks initialized with rotationally symmetric weight distributions, *e.g.* i.i.d. Gaussian weight initialization, training with fixed orthogonal transformations of activations yields the same distribution of training trajectories as untransformed activations, whereas an arbitrary linear transformation would not.

Given a similarity index $s(\cdot, \cdot)$ that is invariant to orthogonal transformation, one can construct a similarity index $s'(\cdot, \cdot)$ that is invariant to any invertible linear transformation by first orthonormalizing the columns of X and Y, and then applying $s(\cdot, \cdot)$, *i.e.* given thin QR decompositions $X = Q_A R_A$ and $Y = Q_B R_B$ for $Q_X^T Q_X = Q_Y^T Q_Y = I$, one can construct a similarity index $s'(X, Y) = s(Q_X, Q_Y)$. $s'(\cdot, \cdot)$ is invariant to invertible linear transformation because orthonormal bases with the same span are related to each other by orthonormal transformation. We show this formally below.

		Invariant to		
Similarity Index	**Formula**	**Invertible Linear Transform**	**Orthogonal Transform**	**Isotropic Scaling**
CCA (R^2_{CCA})	$\|\|Q_Y^{\text{T}} Q_X\|\|_{\text{F}}^2/p_1$	✓	✓	✓
CCA ($\bar{\rho}_{\text{CCA}}$)	$\|\|Q_Y^{\text{T}} Q_X\|\|_*/p_1$	✓	✓	✓
SVCCA (R^2_{SVCCA})	$\|\|(U_Y T_Y)^{\text{T}} U_X T_X\|\|_{\text{F}}^2/\text{min}(\|\|T_X\|\|_{\text{F}}^2, \|\|T_Y\|\|_{\text{F}}^2)$	In a Subspace	✓	✓
SVCCA ($\bar{\rho}_{\text{SVCCA}}$)	$\|\|(U_Y T_Y)^{\text{T}} U_X T_X\|\|_*/\text{min}(\|\|T_X\|\|_{\text{F}}^2, \|\|T_Y\|\|_{\text{F}}^2)$	In a Subspace	✓	✓
Linear Reg. (R^2_{LR})	$\|\|Q_Y^{\text{T}} X\|\|_{\text{F}}^2/\|\|X\|\|_{\text{F}}^2$	Y Only	✓	✓
PWCCA	$\sum_{i=1}^{p_1} \alpha_i \rho_i/\|\|\alpha\|\|_1, \alpha_i = \sum_j \|\langle \mathbf{h}_i, \mathbf{x}_j \rangle\|$	✗	✗	✓
Linear CKA	$\|\|Y^{\text{T}} X\|\|_{\text{F}}^2/(\|\|X^{\text{T}} X\|\|_{\text{F}} \|\|Y^{\text{T}} Y\|\|_{\text{F}})$	✗	✓	✓
RBF CKA	$\text{tr}(KHLH)/\sqrt{\text{tr}(KHKH)\text{tr}(LHLH)}$	✗	✓	✓[1]

Table B.1: Summary of similarity methods investigated. Q_X and Q_Y are orthonormal bases for the columns of X and Y. U_X and U_Y are the left-singular vectors of X and Y sorted in descending order according to the corresponding singular vectors. $||\cdot||_*$ denotes the nuclear norm. T_X and T_Y are truncated identity matrices that select left-singular vectors such that the cumulative variance explained reaches some threshold. For RBF CKA, K and L are kernel matrices constructed by evaluating the RBF kernel between the examples, and H is the centering matrix $H_n = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^{\text{T}}$.

Proposition A.1. *Let X be an $n \times p$ matrix of full column rank and let A be an invertible $p \times p$ matrix. Let $X = Q_X R_X$ and $XA = Q_{XA} R_{XA}$, where $Q_X^T Q_X = Q_{XA}^T Q_{XA} = I$ and R_X and R_{XA} are invertible. If $s(\cdot, \cdot)$ is invariant to orthogonal transformation, then $s(Q_X, Y) = s(Q_{XA}, Y)$.*

Proof. If $Q_X \tilde{A} = Q_{XA}$, then:

$$I = Q_{XA}^{\text{T}} Q_{XA} = \tilde{A}^{\text{T}} Q_X^{\text{T}} Q_X \tilde{A} = \tilde{A}^{\text{T}} \tilde{A}$$

X and XA, and thus Q_X and Q_{XA}, have the same column span, so $\tilde{A}$ exists. Specifically, $\tilde{A} = R_X A R_{XA}^{-1}$. Thus, $s(Q_X, Y) = s(Q_X \tilde{A}, Y) = s(Q_{XA}, Y)$. □

A.3 Invariance to Isotropic Scaling

We expect the similarity index to be invariant to isotropic scaling, *i.e.* $s(X, Y) = s(\alpha X, \beta Y)$ for any $\alpha, \beta \in \mathbb{R}^+$. That said, if a similarity index is invariant to both orthogonal transformation and non-isotropic scaling, *i.e.* rescaling of individual features, then it is invariant to any invertible linear transformation. This follows from the existence of the singular value decomposition of the transformation matrix. Generally, we are interested in similarity indexes that are invariant to isotropic scaling but not necessarily invariant to non-isotropic scaling.

B Related Similarity Indexes

In this section, we briefly review linear regression, canonical correlation, and other related methods in the context of measuring similarity between neural network representations. We let Q_X and Q_Y represent any orthonormal bases for the columns of X and Y, *i.e.* $Q_X = X(X^{\text{T}}X)^{-1/2}$, $Q_Y = Y(Y^{\text{T}}Y)^{-1/2}$ or rotations thereof. Table B.1 summarizes the formulae and invariance properties of the indexes used in experiments. For a comprehensive review of linear indexes for measuring multivariate similarity, see Ramsay et al. (1984).

Linear Regression. A simple way to relate neural network representations is via linear regression. One can fit every neuron in Y as a linear combination of neurons from X. A suitable summary statistic is the total fraction of variance explained by the fit:

$$R^2_{\text{LR}} = 1 - \frac{\min_B ||Y - XB||_{\text{F}}^2}{||Y||_{\text{F}}^2} = \frac{||Q_Y^{\text{T}} X||_{\text{F}}^2}{||X||_{\text{F}}^2} \tag{6}$$

[1]Invariance of RBF CKA to isotropic scaling depends on the procedure used to select the RBF kernel bandwidth parameter. In our experiments, we select the bandwidth as a fraction of the median distance, which yields a similarity index that is invariant to isotropic scaling.

We are unaware of any application of linear regression to measuring similarity of neural network representations, although Romero et al. (2014) used a least squares loss between activations of two networks to encourage thin and deep "student" networks to learn functions similar to wide and shallow "teacher" networks.

Canonical Correlation Analysis (CCA). Canonical correlation finds bases for two matrices such that, when the original matrices are projected onto these bases, the correlation is maximized. For $1 \leq i \leq p_1$, the i^{th} canonical correlation coefficient ρ_i is given by:

$$\begin{aligned} \rho_i = & \max_{\mathbf{w}_X^i, \mathbf{w}_Y^i} \text{corr}(X\mathbf{w}_X^i, Y\mathbf{w}_Y^i) \\ & \text{subject to } \ \forall_{j<i} \ X\mathbf{w}_X^i \perp X\mathbf{w}_X^j, Y\mathbf{w}_Y^i \perp Y\mathbf{w}_Y^j \end{aligned} \tag{7}$$

The vectors $\mathbf{w}_X^i \in \mathbb{R}^{p_1}$ and $\mathbf{w}_Y^i \in \mathbb{R}^{p_2}$ that maximize ρ_i are the canonical weights, which transform the original data into canonical variables $X\mathbf{w}_X^i$ and $Y\mathbf{w}_Y^i$. The constraints in equation 7 enforce orthogonality of the canonical variables.

For the purpose of this work, we consider two summary statistics of the goodness of fit of CCA:

$$R^2_{\text{CCA}} = \frac{\sum_{i=1}^{p_1} \rho_i^2}{p_1} = \frac{||Q_Y^{\text{T}} Q_X||_{\text{F}}^2}{p_1} \tag{8}$$

$$\bar{\rho}_{\text{CCA}} = \frac{\sum_{i=1}^{p_1} \rho_i}{p_1} = \frac{||Q_Y^{\text{T}} Q_X||_*}{p_1} \tag{9}$$

where $|| \cdot ||_*$ denotes the nuclear norm.

The mean squared CCA correlation coefficient R^2_{CCA} is also known as *Yanai's GCD measure* (Ramsay et al., 1984), and several statistical packages report the sum of the squared canonical correlations $p_1 R^2_{\text{CCA}} = \sum_{i=1}^{p_1} \rho_i^2$ under the name *Pillai's trace* (SAS Institute, 2015; StataCorp, 2015). $\bar{\rho}_{\text{CCA}}$, the mean CCA correlation, was previously used to measure similarity between neural network representations in Raghu et al. (2017).

SVCCA. CCA is sensitive to perturbation when the condition number of X or Y is large (Golub & Zha, 1995). To improve robustness, *singular vector CCA* performs CCA on truncated singular value decompositions of X and Y (Raghu et al., 2017; Mroueh et al., 2015; Kuss & Graepel, 2003). As formulated in Raghu et al. (2017), SVCCA keeps enough principal components of the input matrices to explain a fixed proportion of the variance, and drops remaining components. Thus, it is invariant to invertible linear transformation only insofar as the retained subspace does not change.

Projection-Weighted CCA. Morcos et al. (2018) propose a different strategy to reduce the sensitivity of CCA to perturbation, which they term "projection-weighted canonical correlation":

$$\rho_{\text{PW}} = \frac{\sum_{i=1}^{c} \alpha_i \rho_i}{\sum_{i=1} \alpha_i} \qquad \alpha_i = \sum_j |\langle \mathbf{h}_i, \mathbf{x}_j \rangle| \tag{10}$$

where the $\mathbf{x}_j$ are the columns of X, and the $\mathbf{h}_i$ are the canonical variables formed by projecting X to the canonical coordinate frame. Some algebraic manipulation reveals that PWCCA is closely related to linear regression, since:

$$R^2_{\text{LR}} = \frac{\sum_{i=1}^{c} \alpha_i' \rho_i^2}{\sum_{i=1} \alpha_i'} \qquad \alpha_i' = \sum_j \langle \mathbf{h}_i, \mathbf{x}_j \rangle^2 \tag{11}$$

Neuron Alignment Procedures. Other works have searched for alignment between individual neurons, rather than alignment between subspaces. Li et al. (2015) examine the correlation or empirical mutual information matrix between the neurons in different neural networks, and attempt to find a bipartite match or semi-match that maximizes the sum of the correlations between the neurons, and then to measure the average correlations. Wang et al. (2018) propose to search for subsets of neurons $\tilde{X} \subset X$ and $\tilde{Y} \subset Y$ such that, to within some tolerance, every neuron in $\tilde{X}$ can be represented by a linear combination of neurons from $\tilde{Y}$ and vice versa. They find that the maximum matching subsets are very small for intermediate layers of neural networks.

Mutual Information. Among non-linear measures, one candidate is mutual information, which is invariant not only to invertible linear transformation, but to any invertible transformation. Li et al. (2015) previously used mutual information to measure neuronal alignment. In the context of comparing representations, we believe mutual information is not useful. Given any pair of representations produced by deterministic functions of the same input, the mutual information between either representation and the input must be at least as large as the mutual information between the representations. Moreover, in fully invertible neural networks (Dinh et al., 2017; Jacobsen et al., 2018), the mutual information between any two layers is equal to the entropy of the network's input.

C Additional Experiments

C.1 Layer Correspondence for Transformer Models

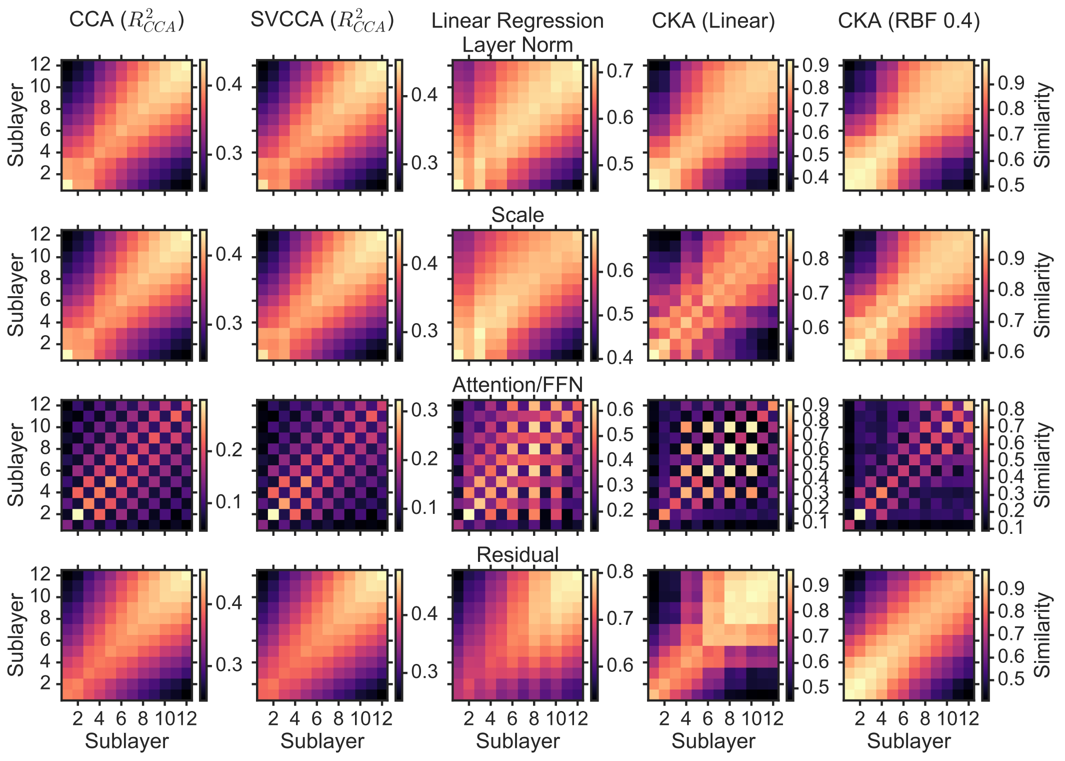

Figure C.1: Similarity between the 12 sublayers of Transformer encoders, for each of the 4 possible places in each sublayer that representations may be taken (see Figure C.2), averaged across 45 pairs of models trained from different random initializations. All similarity indices broadly reflect the structure of the network, although there are clear differences.

In Figure C.1, we show similarity between the 12 sublayers of the encoders of 45 pairs of Transformer models (Vaswani et al., 2017) trained from different random initializations to perform English to German translation. Each Transformer sublayer contains four operations, shown in Figure C.2, and results vary based which operation the representation is taken after. Table C.1 shows the accuracy with which we can identify corresponding layers between network pairs by maximal similarity. All similarity indexes achieve non-negligible accuracy and thus pass the sanity check, although RBF CKA and R^2_{CCA} typically perform better than other methods.

The Transformer architecture alternates between self-attention and feed-forward network sublayers. The checkerboard pattern in similarity plots for the Attention/FFN layer in Figure C.1 indicates that representations of feed-forward network sublayers are more similar to other feed-forward network sublayers than to self-attention sublayers, and similarly, representations of self-attention sublayers are more similar to other self-attention sublayers than to feed-forward network layers. CKA also reveals a checkerboard pattern for activations after the channel-wise scale operation (before the self-

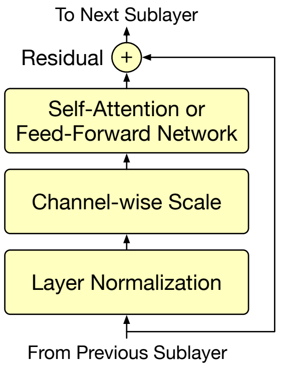

Figure C.2: Architecture of a single sublayer of the Transformer encoder used in experiments. The encoder includes 12 sublayers, alternating between self-attention and feed-forward network sublayers.

Index	Layer Norm	Scale	Attn/FFN	Residual
CCA ($\bar{\rho}$)	85.3	85.3	**94.9**	90.9
CCA (R^2_{CCA})	87.8	87.8	**95.3**	**95.2**
SVCCA ($\bar{\rho}$)	78.3	83.0	89.5	75.9
SVCCA (R^2_{CCA})	85.4	86.9	90.8	84.7
PWCCA	**88.5**	88.9	**96.1**	87.0
Linear Reg.	78.1	83.7	76.0	36.9
CKA (Linear)	78.6	**95.6**	86.0	73.6
CKA (RBF 0.2)	76.5	73.1	70.5	76.2
CKA (RBF 0.4)	**92.3**	**96.5**	89.1	**98.1**
CKA (RBF 0.8)	80.8	**95.8**	**93.6**	90.0

Table C.1: Accuracy of identifying corresponding sublayers based maximum similarity, for 45 pairs of architecturally identical 12-sublayer Transformer encoders. For asymmetric indexes (PWCCA and linear regression) we symmetrize the similarity matrix as $S+S^{\mathrm{T}}$. CKA RBF kernel parameters are specified as the fraction of the median distance used as the standard deviation. Results not significantly different from the best result are bold-faced ($\alpha = 0.05$, jackknife z-test).

attention/feed-forward network operation) that other methods do not. Because CCA is invariant to non-isotropic scaling, CCA similarities before and after channel-wise scaling are identical. Thus, CCA cannot capture this structure, even though it is consistent across different networks.

C.2 Analysis of the Shared Subspace

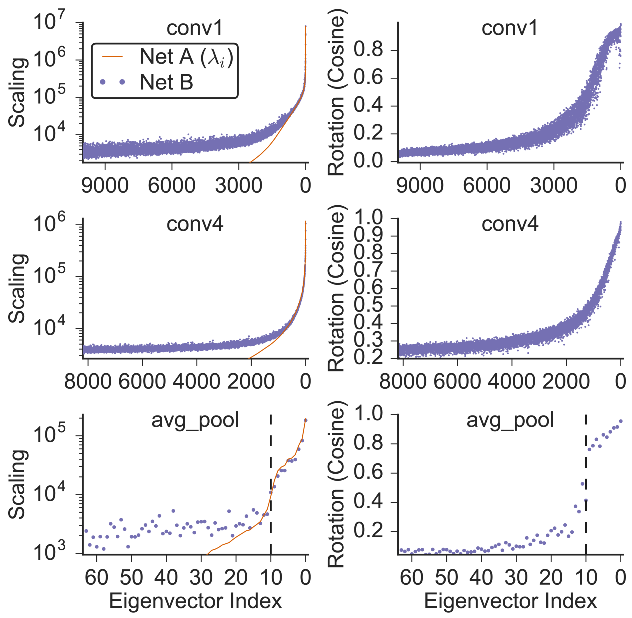

Figure C.3: The shared subspace of two Tiny-10 networks A and B trained from random initialization is spanned primarily by the eigenvectors corresponding to the largest eigenvalues. Each row represents a different network layer. Note that the average pooling layer has only 64 units. **Left**: Scaling of the eigenvectors q_i of the representational similarity matrix XX^{T} from network A by networks A and B. Orange lines show $||XX^{\mathrm{T}}q_i||_2$, *i.e.* the eigenvalues. Purple dots show $||YY^{\mathrm{T}}q_i||_2$, the scaling by the network B. **Right**: Cosine of the rotation by network B, $||q_iYY^{\mathrm{T}}q_i||_2/||YY^{\mathrm{T}}q_i||_2$.

Equation 5 suggests a way of further elucidating what CKA is measuring, based on analyzing the action of one Gram matrix L (YY^{T} in the linear setting) when applied to the eigenvectors Q_K of the other Gram matrix K (XX^{T} in the linear setting). By definition, the columns of KQ_K are scaled by the eigenvalues of K and are not rotated. The degree of scaling and rotation by L thus indicates how similar the action of L is to K, for each eigenvector of K. For visualization purposes, this approach is somewhat less useful than the CKA summary statistic, since it does not collapse the similarity to a single number, but it provides a more complete picture of what CKA measures. Figure C.3 shows that, for large eigenvectors, XX^{T} and YY^{T} have similar actions, but the rank of the subspace where this holds is substantially lower than the dimensionality of the activations. In the penultimate (global average pooling) layer, the dimensionality of the shared subspace is approximately 10, which is the number of classes in the CIFAR-10 dataset.

C.3 Other SVCCA Thresholds

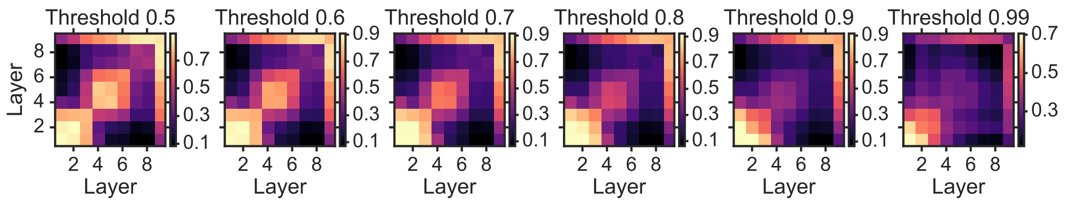

Figure C.4: R^2_{CCA}, the sum of squared CCA singular values, for additional thresholds beyond the 0.99 threshold suggested by Raghu et al. (2017). No threshold reveals the structure of the network.

C.4 CKA Results for Other CNN Architectures

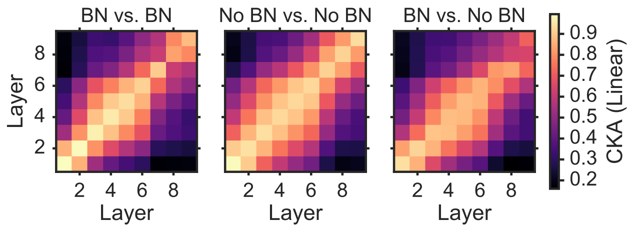

Figure C.5: Linear CKA between networks with and without batch normalization trained from different random initializations. The largest difference between networks with and without batch normalization appears to be at the last convolutional layer. Optimal hyperparameters were separately selected for the batch normalized network (93.9% average accuracy) and the network without batch normalization (91.5% average accuracy).

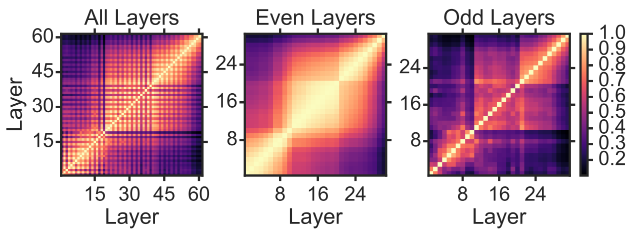

Figure C.6: Linear CKA between layers of a ResNet-62 model shows no pathology. The grid pattern for ResNets in the left panel arises from the architecture. Right panels show similarity separately for even layer (post-residual) and odd layer (block interior) activations.

C.5 Similarity Between Different Architectures with CCA-Based Methods

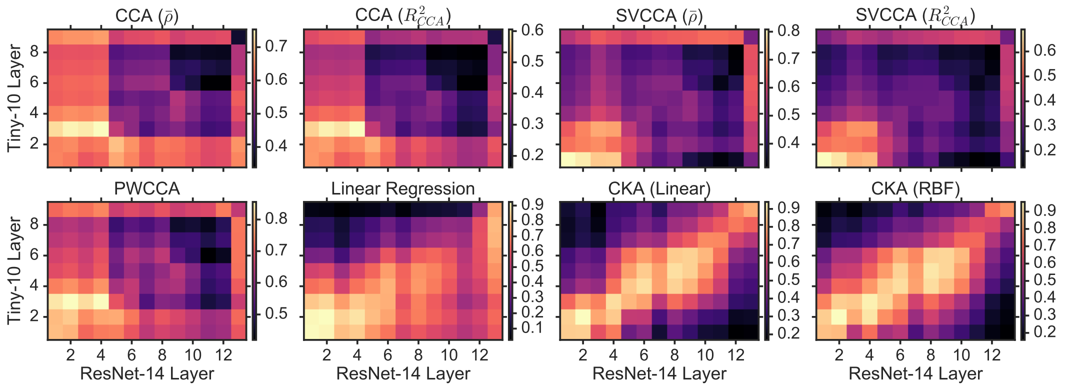

Figure C.7: Similarity between layers of different architectures (Tiny-10 and ResNet-14) for all methods investigated. Only CKA reveals meaningful correspondence. SVCCA results resemble Figure 7 of Raghu et al. (2017). All plots show similarity on the CIFAR-10 training set.

C.6 CKA Results for Networks at Initialization

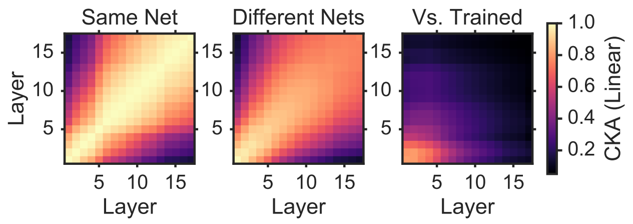

Figure C.8: Similarity of the Plain-18 network at initialization. **Left**: Similarity between layers of the same network. **Middle**: Similarity between untrained networks with different initializations. **Right**: Similarity between untrained and trained networks.

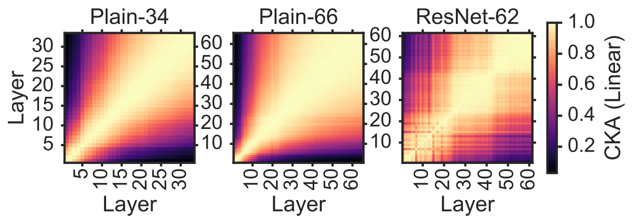

Figure C.9: Similarity between layers at initialization for deeper architectures.

D Architecture Details

All non-ResNet architectures are based on All-CNN-C (Springenberg et al., 2014), but none are architecturally identical. The Plain-10 model is very similar, but we place the final linear layer after the average pooling layer and use batch normalization because these are common choices in modern architectures. We use these models because they train in minutes on modern hardware.

Tiny-10
3×3 conv. 16-BN-ReLu $\times 2$
3×3 conv. 32 stride 2-BN-ReLu
3×3 conv. 32-BN-ReLu $\times 2$
3×3 conv. 64 stride 2-BN-ReLu
3×3 conv. 64 valid padding-BN-ReLu
1×1 conv. 64-BN-ReLu
Average pooling
Logits

Table D.1: The Tiny-10 architecture, used in Figure 1 and Table 1. The average Tiny-10 model achieved 89.4% accuracy.

Plain-$(8n + 2)$
3×3 conv. 96-BN-ReLu $\times(3n - 1)$
3×3 conv. 96 stride 2-BN-ReLu
3×3 conv. 192-BN-ReLu $\times(3n - 1)$
3×3 conv. 192 stride 2-BN-ReLu
3×3 conv. 192 BN-ReLu $\times(n - 1)$
3×3 conv. 192 valid padding-BN-ReLu
1×1 conv. 192-BN-ReLu $\times n$
Average pooling

Table D.2: The Plain-$(8n + 2)$ architecture, used in Figures 2 and 3. Mean accuracies: Plain-10, 93.9%; Plain-18: 94.8%; Plain-34: 93.7%; Plain-66: 91.3%

Width-n
3×3 conv. n-BN-ReLu $\times 2$
3×3 conv. n stride 2-BN-ReLu
3×3 conv. n-BN-ReLu $\times 2$
3×3 conv. n stride 2-BN-ReLu
3×3 conv. n valid padding-BN-ReLu
1×1 conv. n-BN-ReLu
Average pooling
Logits

Table D.3: The architecture used for width experiments in Figure 4.