Do Self-Supervised and Supervised Methods Learn Similar Visual Representations?

Tom George Grigg [*†] **Dan Busbridge** [†] **Jason Ramapuram** **Russ Webb**

Apple

Abstract

Despite the success of a number of recent techniques for visual self-supervised deep learning, there has been limited investigation into the representations that are ultimately learned. By leveraging recent advances in the comparison of neural representations, we explore in this direction by comparing a contrastive self-supervised algorithm to supervision for simple image data in a common architecture. We find that the methods learn similar intermediate representations through dissimilar means, and that the representations diverge rapidly in the final few layers. We investigate this divergence, finding that these layers strongly fit to their distinct learning objectives. We also find that the contrastive objective implicitly fits the supervised objective in intermediate layers, but that the reverse is not true. Our work particularly highlights the importance of the learned intermediate representations, and raises critical questions for auxiliary task design.

1 Introduction

In the last two decades, progress in deep learning for visual tasks has primarily been driven by training convolutional neural networks (CNNs) [1, 2, 3, 4, 5, 6, 7] on large labelled datasets via supervised learning (SL). More recently, self-supervised learning (SSL) algorithms have started to close the performance gap [8, 9, 10, 11, 12, 13, 14]. The success of these visual SSL algorithms raises important questions from a representation learning perspective: how are SSL methods building competitive respresentations without access to class labels? Do learned representations differ between SL and SSL? If so, can/should we encourage them to be similar? Do different SSL objectives learn qualitatively different representations? In this work, we begin to shed light in this direction by comparing the representations of CIFAR-10 (C10) induced in a ResNet-50 (R50) architecture by SL against those induced by SimCLR, a prominent contrastive SSL algorithm. We find that:

- Post-residual representations are similar across methods, however residual (block-interior) representations are dissimilar; similar structure is recovered by solving different problems.
- Initial residual layer representations are similar, indicating a shared set of primitives.
- The methods strongly fit to their distinct objectives in the final few layers, where SimCLR learns augmentation invariance and SL fits to the class structure.
- SL does not implicitly learn augmentation invariance, but learning to become invariant to SimCLR's augmentations implicitly fits the class structure and induces linear separability.
- The representational structures rapidly diverge in the final layers, suggesting that SimCLR's performance stems from class-informative *intermediate* representations, rather than implicit structural agreement between learned solutions to the SL and SimCLR objectives.

[*]Work completed during an internship at Apple.

[†]Equal contribution. Correspondence to tomgeorgegrigg@gmail.com and dbusbridge@apple.com.

2nd Workshop on Self-Supervised Learning: Theory and Practice (NeurIPS 2021), Sydney, Australia.

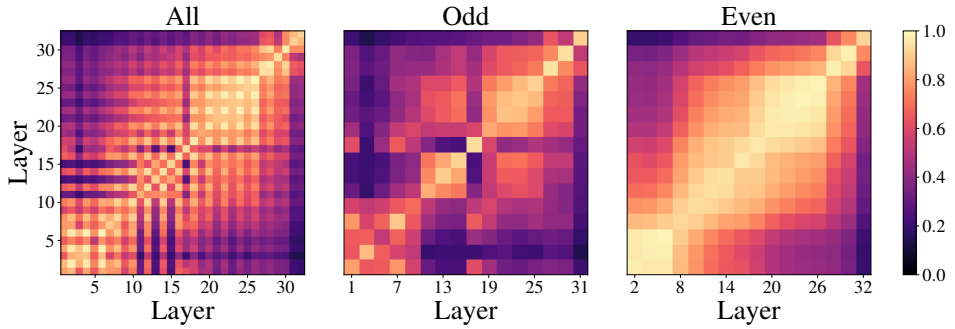

Figure 1: CKA between layers of R50 networks trained via SimCLR. We show all, odd, and even layers in the left, middle, and right plots respectively. In contrast to prior work, we compare across different initializations as a sanity check for solution stability.

2 Background

Multi-view visual SSL A number of recent SSL algorithms for visual data focus on a view invariance auxiliary objective, where the model learns to identify different views of the same input image and distinguish views of different images. Here, we focus on SimCLR [11] as a step towards understanding contrastive self-supervised representation learning. We leave the analysis of alternative visual SSL methods for future work.

SimCLR SimCLR learns representations by contrasting different views $\boldsymbol{v}_t(\boldsymbol{x}), \boldsymbol{v}_{t'}(\boldsymbol{x})$ of a single image $\boldsymbol{x}$ to views $\boldsymbol{v}_{t''}(\boldsymbol{x}_-), \boldsymbol{v}_{t'''}(\boldsymbol{x}_-)$ of other images, where we sample from a family of augmentations $t, t', t'', t''' \sim \mathcal{T}$. Views are constructed through application: $\boldsymbol{v}_t(\boldsymbol{x}) = (g_{\boldsymbol{\theta}} \circ f_{\boldsymbol{\theta}} \circ t)(\boldsymbol{x})$, where $f_{\boldsymbol{\theta}}$ is the parametric backbone, typically a CNN, and $g_{\boldsymbol{\theta}}$ is the Noise Contrastive Estimation (NCE) head, typically an MLP. SimCLR's objective is then to minimize InfoNCE [11, 15]:

$$\mathcal{L}^{(i,j)}_{\text{InfoNCE}} = -\log \frac{\exp(\text{sim}(\boldsymbol{v}_i, \boldsymbol{v}_j)/\tau)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(\text{sim}(\boldsymbol{v}_i, \boldsymbol{v}_k)/\tau)}, \tag{1}$$

where $\boldsymbol{v}_i = \boldsymbol{v}_t(\boldsymbol{x})$, $\boldsymbol{v}_j = \boldsymbol{v}_{t'}(\boldsymbol{x})$ are different views of the same image, $\boldsymbol{v}_k = \boldsymbol{v}_{t''}(\boldsymbol{x}^{(k)}_-)$ are different views of different images, τ is the temperature, and $\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\mathsf{T}\boldsymbol{v}/\|\boldsymbol{u}\|_2 \|\boldsymbol{v}\|_2$ is cosine similarity.

Comparing neural representation spaces Comparing neural representations is challenging due to their distributed nature, potential misalignment, and high dimensionality. Prior work has demonstrated the utility of Centered Kernel Alignment (CKA) as a similarity index which elegantly addresses these challenges [16], enabling the analysis of a variety of neural architectures [16, 17, 18].

Let $\boldsymbol{X} \in \mathbb{R}^{m \times p_1}$, $\boldsymbol{Y} \in \mathbb{R}^{m \times p_2}$ be p_1 and p_2 dimensional representation matrices whose rows are aligned[1]. Let $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^\mathsf{T}$, $\boldsymbol{L} = \boldsymbol{Y}\boldsymbol{Y}^\mathsf{T}$ be the corresponding Gram matrices. The CKA value is the normalized Hilbert-Schmidt Independence Criterion (HSIC) [19] of these Gram matrices:

$$\text{CKA}(\boldsymbol{K}, \boldsymbol{L}) = \frac{\text{HSIC}(\boldsymbol{K}, \boldsymbol{L})}{\sqrt{\text{HSIC}(\boldsymbol{K}, \boldsymbol{K})\,\text{HSIC}(\boldsymbol{L}, \boldsymbol{L})}}, \tag{2}$$

We use the linear kernel due to its strong empirical performance and computational efficiency, simplifying the calculation of HSIC to $(m^2 - 1)\,\text{HSIC}(\boldsymbol{K}, \boldsymbol{L}) = \text{Tr}(\boldsymbol{K}\boldsymbol{H}\boldsymbol{L}\boldsymbol{H}) = \text{vec}(\boldsymbol{K}') \cdot \text{vec}(\boldsymbol{L}')$, where $\boldsymbol{K}' = \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}$, $\boldsymbol{L}' = \boldsymbol{H}\boldsymbol{L}\boldsymbol{H}$, and $\boldsymbol{H} = \boldsymbol{I}_m - \frac{1}{m}\boldsymbol{1}\boldsymbol{1}^\mathsf{T}$ is the centering matrix.

Experimental setup We use a R50 [3] backbone for each model. For SimCLR, we train as specified in Chen et al. [11]. We group representations into residual (odd) and post-residual (even) layers, in line with the analysis of Kornblith et al. [16]. Further details are outlined in Appendix A.

[1] The ith row in $\boldsymbol{X}$ and $\boldsymbol{Y}$ correspond to the ith sample for all $i \in 1, \ldots, m$.

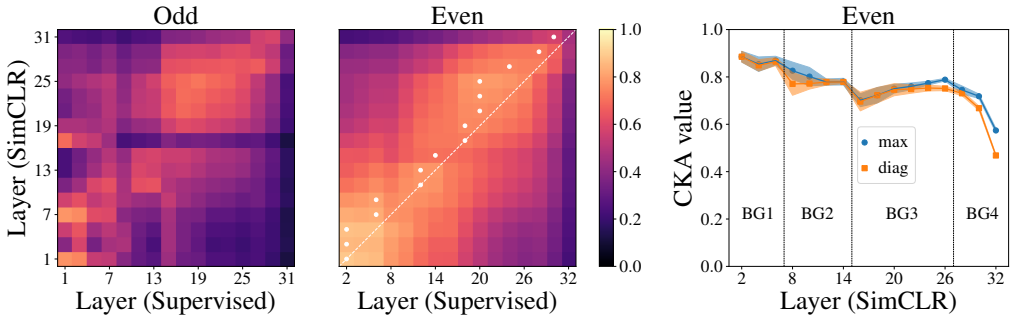

Figure 2: (Left/Middle) CKA between the odd/even layers of networks trained by SimCLR and SL. For the evens, we mark the most similar SL layer for each SimCLR layer with a white dot. (Right) For each even layer in SimCLR, the similarity to its corresponding supervised layer (diag), and to the most similar supervised layer (max). We also denote the block groups (BG) (see Appendix A.1).

3 Results

3.1 Internal representational structure of SimCLR

We begin by using CKA to study the internal representational similarity of SimCLR in Figure 1. This result mirrors the supervised analysis of Kornblith et al. [16], indicating that SL and SimCLR utilize the residual architecture in a similar way, with residual blocks decoupling from each other. For completeness, we replicate the SL result under our experimental setup in Appendix B.

3.2 Comparing early and intermediate SimCLR and supervised representations

Next, we compare the representational structures induced by SimCLR and SL. In Figure 2, we plot the odd and even layer CKA matrices across the learning methods, we observe:

Common primitives Residual representations are similar in the very early layers, perhaps due to both objectives inducing common primitives like Gabor filters [20].

Dissimilar residual (Odd) Beyond these initial layers, similarity between the residual representations substantially reduces, indicating that each method learns residuals that operate on the input in qualitatively different ways – likely a reflection of their distinct learning objectives.

Similar post-residual (Even) Despite the dissimilarity of residuals, there is high similarity across the diagonal in the post-residual layers, indicating that the representations accumulated remain similar across learning methods; similar representations are learned in a dissimilar way.

Stalling behaviour Finally, SimCLR appears to "stall" upon entering a new BG, remaining more similar to previous supervised layers, before "catching up" to the diagonal. This may be induced by SimCLR's strong augmentations, requiring a broader distribution to be compressed after each BG.

3.3 Late layer representational dissimilarity of SimCLR and supervised learning

Figure 2 (right) indicates that the representational structures learned by SimCLR and SL rapidly diverge in the final block group. Here, we analyze this behaviour.

Linear separability of classes We first investigate the effect that this divergence has on performance. We compute the accuracy of linear probes fitted at each layer in the SimCLR and supervised networks (Figure 3 (left)). We find a monotonic increase in linear separability of the classes for both methods. This suggests that despite the divergence in later layers, both representational structures continue to become more linearly separable with respect to the classes. This raises the question: if the structures are diverging, but both are becoming more separable, what exactly is being learned?

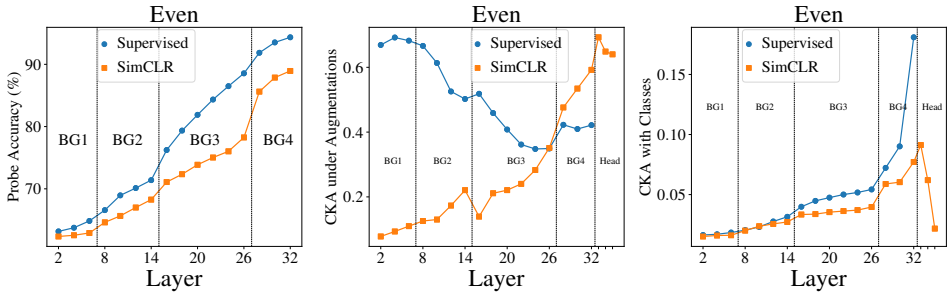

Figure 3: (Left) Linear probe accuracies for learned representations in the SimCLR and SL models. (Middle) CKA between representations of differently augmented datasets at corresponding layers. (Right) CKA of learned representations with the class representations. We plot post-residual (even) layers only, denoting the block groups (BG) and NCE head (Head) where appropriate.

Augmentation invariance In Figure 3 (middle), we inspect what happens in the layers of both networks with respect to SimCLR's augmentation invariance objective. Here, we augment each sample in the C10 test dataset with two augmentations sampled from the augmentation distribution used during training[2], creating pairs of augmented test datasets. We measure the degree of invariance at each layer by plotting the CKA value between the representations of these augmented datasets. We observe that SimCLR's representations become more augmentation invariant with depth, increasingly so in the final few layers of the network. This contrasts with SL where the representations start out similar under (weaker) augmentations, then diverge until the final block group, where we see a small increase in CKA – presumably due to classification. This result tells us (1) SimCLR does learn substantial augmentation invariance and (2) SL does not implicitly learn augmentation invariant representations. This is perhaps surprising from the perspective of classification as a form of augmentation invariance where the augmentation distribution is the class-conditioned data distribution. Full CKA heatmaps are presented in Appendix C.

Mapping to the classes Next we look at the SL objective which, from a representation learning perspective, maps inputs to their assigned vertices on the simplex in the class representation space. In Figure 3 (right), we plot the CKA similarity between the class representations and the learned representations in the layers of the SimCLR and supervised networks. We observe a monotonic increase in CKA with the class structure for both methods throughout the backbone, offering insight into the increasing linear separability. It is however clear that SL accelerates much more rapidly towards the class structure in the final block group due to explicit optimization – likely explaining the divergence of SL and SimCLR in Figure 2. We also observe a decrease in similarity to the class structure after the first layer of the NCE head, perhaps revealing its role as a buffer which allows the backbone to learn richer class-informative features rather than immediately fit to InfoNCE.

4 Conclusion

We have shown the utility of CKA for comparing across *learning methods*, rather than architectures. Using this approach, we have demonstrated that SimCLR representations are similar to those of supervised learning in their intermediate layers. Interestingly, we see divergence in the final few layers where each methods fits to its own objectives. Here, SimCLR learns augmentation invariance, contrasting with supervised learning, which instead is more strongly drawn to align with the labelled class structure. This suggests that it is not similarity of the final representational structures that facilitates SimCLR's strong downstream performance. Rather, it is the similarity of the intermediate representations, i.e. the class-informative features that are learned along the way.

These findings raise important questions for auxiliary task design: Can we build label-free tasks that share more intermediate features with supervised learning? Should we include inductive biases that look like "mapping to the simplex", e.g. orthogonality? Is mapping to the simplex desirable? Or are self-supervised representations more robust in a multi-task/multi-distribution setting? We leave these questions for future work.

[2]ImageNet augmentations for SL, SimCLR augmentations for SimCLR.

Acknowledgements

The authors would like to thank the following people for their help throughout the process of writing this paper, in alphabetical order: Barry-John Theobald, Luca Zappella, and Xavier Suau Cuadros. Additionally, we thank Andrea Klein, Cindy Liu, Guihao Liang, Guillaume Seguin, Li Li, Okan Akalin, and the wider Apple infrastructure team for assistance with developing scalable, fault tolerant code.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL https://doi.org/10.1109/CVPR.2009.5206848.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

[5] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541. URL https://doi.org/10.1162/neco.1989.1.4.541.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL http://arxiv.org/abs/1409.0575.

[7] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer, 2014. doi: 10.1007/978-3-319-10590-1_53. URL https://doi.org/10.1007/978-3-319-10590-1_53.

[8] Jean-Baptiste Alayrac, Adrià Recasens, Rosalia Schneider, Relja Arandjelovic, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman. Self-supervised multimodal versatile networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0060ef47b12160b9198302ebdb144dcf-Abstract.html.

[9] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In

Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/70feb62b69f16e0238f741fab228fec2-Abstract.html.

[10] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021. URL https://arxiv.org/abs/2104.14294.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20j.html.

[12] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fcbc95ccdd551da181207c0c1400c655-Abstract.html.

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

[14] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *CoRR*, abs/2103.03230, 2021. URL https://arxiv.org/abs/2103.03230.

[15] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. URL http://arxiv.org/abs/1807.03748.

[16] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR, 2019. URL http://proceedings.mlr.press/v97/kornblith19a.html.

[17] 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 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=KJNcAkY8tY4.

[18] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *CoRR*, abs/2108.08810, 2021. URL https://arxiv.org/abs/2108.08810.

[19] Arthur Gretton, Kenji Fukumizu, Choon Teo, Le Song, Bernhard Schölkopf, and Alex Smola. A kernel statistical test of independence. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper/2007/file/d5cfead94f5350c12c322b5b664544c1-Paper.pdf.

[20] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a

local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010. URL http://portal.acm.org/citation.cfm?id=1953039.

[21] Zhouyuan Huo, Bin Gu, and Heng Huang. Large batch optimization for deep learning using new complete layer-wise adaptive rate scaling. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 7883–7890. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16962.

[22] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. URL http://arxiv.org/abs/1706.02677.

[23] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017. URL http://arxiv.org/abs/1708.07120.

[24] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

A Experimental Setup

Experimental setup We choose the same ResNet50 architecture [3] for SimCLR and the supervised model. We follow the training procedure described in Chen et al. [11]: all models use the LARS optimizer [21] with linear warmup [22] and a single cycle cosine annealed learning rate schedule [22, 23]. SimCLR models are trained for 1300 epochs with a batch size of 4096 under SimCLR augmentations [11], whereas our supervised models are trained for 300 epochs using a batch size of 8192 under standard ImageNet augmentations[3]. For SimCLR, we implement the original version of Chen et al. [11], where in particular the NCE head is a 3-layer MLP.

For each learning method, we train from 3 different random initializations, resulting in 6 models. Models are trained on the training set of CIFAR-10 (50,000 samples) [24]. Representations are produced on the test set of CIFAR-10 (10,000 samples) under each model's corresponding test augmentation family. Representations are flattened, producing a single vector for each sample.

A.1 Even and Odd representations

For each bottleneck layer, we extract the following representations:

```
class Bottleneck(nn.Module):
    # Other definitions

    def forward(self, x: Tensor) -> Tensor:
        identity = x

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)

        out = self.conv2(out)
        out = self.bn2(out)
        out = self.relu(out)

        out = self.conv3(out)
        ODD_REPRESENTATIONS[i] = out = self.bn3(out)

        if self.downsample is not None:
            identity = self.downsample(x)

        out += identity.
        EVEN_REPRESENTATIONS[i] = out = self.relu(out)

        return out
```

i.e. two representations per bottleneck.

A ResNet50 is built out of 4 block groups, each subsequent group increasing dimensionality (see Table 1). The total number of bottleneck layers across all groups is $16 = 3 + 4 + 6 + 3$, resulting in 16 odd representations and 16 even representations that we use in our analysis.

Table 1: The filter properties and bottleneck multiplicities in each BG of a ResNet50.

Group Name	Number of Bottlenecks	Filters in each Bottleneck
BG1	3	$(1 \times 1, 64)$, $(3 \times 3, 64)$, $(1 \times 1, 256)$
BG2	4	$(1 \times 1, 128)$, $(3 \times 3, 128)$, $(1 \times 1, 512)$
BG3	6	$(1 \times 1, 256)$, $(3 \times 3, 256)$, $(1 \times 1, 1024)$
BG4	3	$(1 \times 1, 512)$, $(3 \times 3, 512)$, $(1 \times 1, 2048)$

[3] `RandomResizedCrop(224)`, `RandomHorizontalFlip` and channel-wise standardisation.

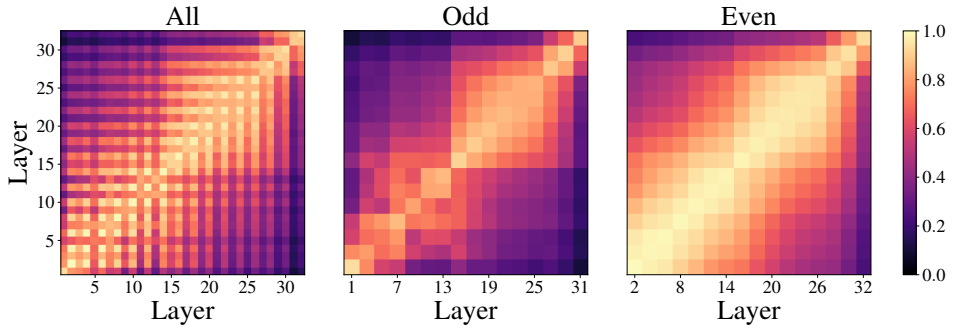

Figure 4: CKA between all layers of ResNet-50 networks trained via supervision. We plot all layers in the left column, and even/odd layers on the middle/right.

B Internal representational structure for supervised learning

Here, we replicate the results of Kornblith et al. [16] in our experimental setup. In particular, in Figure 4 we use CKA to compare the learned representations of ResNet-50 architectures trained via supervised learning, as specified in Appendix A. We note that in contrast to their work, we compare across different initializations in order to check for solution stability.

Corroborating their findings, we observe high similarity across neighbouring post-residual (even) layers in the network, and greater dissimilarity between residual (odd) layers, which largely appear similar only to themselves. The similarity of even layers is explained by the residual connections propagating representations through the network. The dissimilarity of odd layers suggests that each sequential block performs a distinct modification to this propagated residual representation. The similarity within block groups (i.e. at the same dimensionality) is higher than across block groups for all layers. We further note that the results for SimCLR (Figure 1) mirrors that of supervised learning, except there appears to be even greater disagreement across block groups.

C Full Augmentation Invariance CKA Heatmaps

In Figure 5 we provide the all-layers CKA comparisons of the representations of differently augmented test datasets in the same model. The diagonals of Figure 5 correspond to Figure 3 (middle). The augmentation invariance of the supervised model gradually fades starting from the bottom left corner, suggesting that it is initially due to the residual connections and weak augmentation strategy. The SimCLR plot is striking: substantial (but not total) invariance is learned in the NCE head, and this backpropagates into the final few layers of the backbone. However, the representations show limited robustness to augmentation right up until these last few layers of the network.

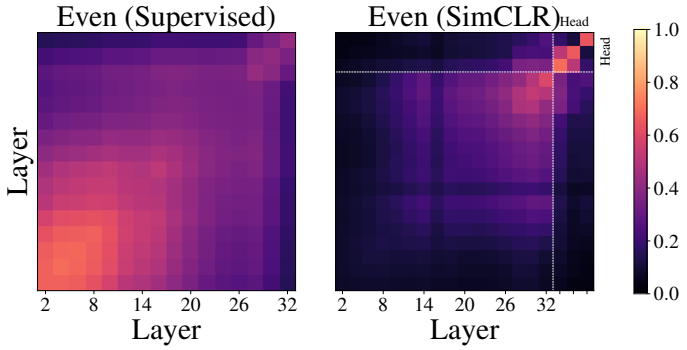

Figure 5: We apply the method-specific training augmentations to the CIFAR-10 test dataset and plot the CKA of the representations as they propagate through the supervised and SimCLR models.