Similarity and Matching of Neural Network Representations

Anonymous Author(s)
Affiliation
Address
email

Abstract

We employ a toolset — dubbed Dr. Frankenstein — to analyse the similarity of representations in deep neural networks. With this toolset, we aim to match the activations on given layers of two trained neural networks by joining them with a stitching layer. We demonstrate that the inner representations emerging in deep convolutional neural networks with the same architecture but different initialisations can be matched with a surprisingly high degree of accuracy even with a single, affine stitching layer. We choose the stitching layer from several possible classes of linear transformations and investigate their performance and properties. The task of matching representations is closely related to notions of similarity. Using this toolset we also provide a novel viewpoint on the current line of research regarding similarity indices of neural network representations: the perspective of the performance on a task.

1 Introduction

A central topic in the analysis of deep neural networks is the investigation of the learned representations. A good understanding of the features learned on inner layers provides means to analyse and improve neural network architectures. One of the primary questions in this line of inquiry is "when are two representations the same"? In this paper, we investigate the notion of similarity of deep neural network representations.

A natural approach to define a notion of similarity is via computing statistics on the data embeddings. There exist several statistical measures [8, 13, 17] each of which serves as a different notion of similarity. These measures have been used successfully to obtain valuable insights about the inner workings of deep neural networks [12, 14, 15, 22]. However, the approach of computing statistics in the activation space of given layers has the property that this notion of similarity does not depend in any way on the rest of the network which determines the performance of the network on the task.

We suggest to put the notion of *matchability* of neural network representations next to notions of similarity based on activation statistics. By matchability we mean that one can take the activations of one deep neural network, transform it, then use it as an input to an other network at a given layer, and still obtain a working system. The concept of matchability — with the presence of a particular task — brings us closer to more practical aspects: the performance on the task.

In this paper, we study the similarity and matchability of neural representations with a toolset dubbed Dr. Frankenstein.[1] With this toolset, we aim to match the activations on given layers of two trained neural networks by joining them with a linear stitching layer. This technique was first used by Lenc and Vedaldi [9], but it has not yet been proposed to analyse notions of representational similarity. Our main contributions can be summarized as follows:

[1] Code available (anonymously) at: `https://bit.ly/similarity-and-matching`

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

- We demonstrate that the inner representations emerging in deep convolutional neural networks with the same architecture but different initializations are matchable surprisingly well, even with a single, affine stitching layer (Section 5). We analyse the structure of the resulting transformation matrices and the uniqueness of the solutions (Section 8).
- We provide a novel perspective regarding the current line of research on similarity indices for neural representations by explicitly referencing the task the networks solve, an aspect which is yet unexplored. We show that the values of similarity indices are not necessarily indicative of task performance on a stitched network, and perhaps more surprisingly, high performance matchings can have differences in the values of similarity indices such as Centered Kernel Alignment (CKA) [1, 8]. This viewpoint also reflects on phenomena experienced by the 'end-users' of these indices. For instance, in the context of continual learning the authors of [12] observe that CKA remains constant, while the accuracy on previous tasks drops drastically (Section 6).

2 Related work

Matching representations The idea of 1x1 convolution *stitching layers* connecting the lower half of a network with the upper part of another network first appears in [9]. They demonstrate that such hybrid "Franken-CNNs" can approach the accuracy of their unmodified counterparts in many cases. For shallower layers, this phenomenon can appear even when the first network is trained on a different task, suggesting that lower level representations are sufficiently generic.

Mapping layers as introduced by [10] are single layer linear 1x1 convolutional networks mapping between two representations. They do not incorporate information from the second half of the network. The authors train mapping layers with MSE and L1 weight regularization, given their main goal of identifying sparse mappings between the filters of two different networks. As [10] notes, learning mapping layers can get stuck at poor local minima, even without L1 regularization. This forces the authors to restrict their experiments to the shallowest layers of the networks they inspect.

We manage to overcome the issue of poor local optima by initializing from the optimal least squares matching (Subsection 8), and as our experiments demonstrate, this alleviates the issue, resulting in stitched networks that often achieve the performance of their constituents.

Similarity indices Canonical Correlation Analysis (CCA) provides a statistical measure of similarity. CCA itself has a tendency to give too much emphasis to the behavior of low-magnitude high-noise latent directions. Similarity indices to make CCA more robust in this sense were designed, namely Singular Value CCA [17] and Projection Weighted CCA [13].

The current state of the art in measuring similarity is Centered Kernel Alignment (CKA) [1, 2, 8]. The authors in [8] conducted a comparative study of the measures CCA, SVCCA, PWCCA as similarity indices, and they observed that the other similarity indices do not succeed in identifying similar layers of identical networks trained from different initializations, whereas CKA manages to do so, so in this paper we chose to focus on the latter. The authors in [8] also consider nonlinear CKA kernels, but as they demonstrate this does not lead to meaningful improvements, so we focus only on the linear variant, (also commonly known as RV coefficient [18]) in line with other literature employing CKA similarity [12, 14, 15, 22].

3 Introducing Dr. Frankenstein

3.1 Preliminaries and notation

Let $f_\theta : \mathcal{X} \to \mathcal{Y}$ denote the input-output function of a feedforward artificial neural network with layers $f_\theta = f_m \circ \cdots \circ f_1$, where $\mathcal{X}$ is the input space, $\mathcal{Y}$ is the output space, $f_i : \mathcal{A}_{i-1} \to \mathcal{A}_i$ are maps between activation spaces $\mathcal{A}_{i-1}$ and $\mathcal{A}_i$ for $i \in [m]$, with $\mathcal{A}_0 = \mathcal{X}$, θ are the parameters of the network, and $m \in \mathbb{N}$ is the number of layers. We consider models that are trained on a dataset $D = \{(x_i, y_i)\}_{i=1}^N$ in a supervised manner with inputs $x_i \in \mathcal{X}$ and labels $y_i \in \mathcal{Y}$, where $N \in \mathbb{N}$ is the dataset size.

Considering now the inner representations of a given f_θ, the central tool of our investigation involves splitting up the network at a given layer to a *representation map* and a *task map*. The *representation*

map at layer L is a function $R_L : \mathcal{X} \to \mathcal{A}_L$ that maps each data point x_i to its *activation vector* $a_{i,L}$ at layer L:

$$R_L(x_i) = f_L \circ \cdots \circ f_1(x_i) = a_{i,L},$$

where $\mathcal{A}_L$ is the activation space at layer L. The activation vectors for the dataset D at layer L are simply denoted by $A_L = (a_{i,L})_{i=1}^N$. The *task map at layer* L is a function $T_L : \mathcal{A}_L \to \mathcal{Y}$ that maps an activation vector at layer L to the final output of the network:

$$T_L(a_{i,L}) = f_m \circ \cdots \circ f_{L+1}(a_{i,L}).$$

Thus, the input-output map of the network f_θ is simply the composition

$$f_\theta = T_L \circ R_L : \mathcal{X} \to \mathcal{Y}$$

for all $L = 1, \ldots, m$. We will omit the index of the layer L when this does not hurt clarity.

3.2 The Frankenstein learning task

Definition 3.1 (Frankenstein network)**.** *Fix two (pretrained) neural networks f_θ and f_ϕ. We will call a transformation $S : \mathcal{A}_{\theta,L} \to \mathcal{A}_{\phi,M}$ between the two activation spaces at layers L and M a* stitching layer. *Given a stitching layer S,* the Frankenstein *or* stitched network *corresponding to S is the composition $F = F_S : \mathcal{X} \to \mathcal{Y}$,*

$$F(x) = T_{\phi,M} \circ S \circ R_{\theta,L}(x),$$

where $T_{\phi,M}$ is the task map at layer M for the network f_ϕ and $R_{\theta,L}$ is the representation map at layer L for network f_θ.

The stitching layer thus realizes a correspondence between the representations of two neural networks: it transforms the activations of one network at a particular layer to be a suitable input to the corresponding layer in the other network. For better readability, we will often call f_θ as Model 1 and f_ϕ as Model 2 and index the corresponding data accordingly ($f_1 = f_\theta$, $f_2 = f_\phi$, etc.). In the experiments of this paper, Model 1 and Model 2 have the same architecture and $L = M$.

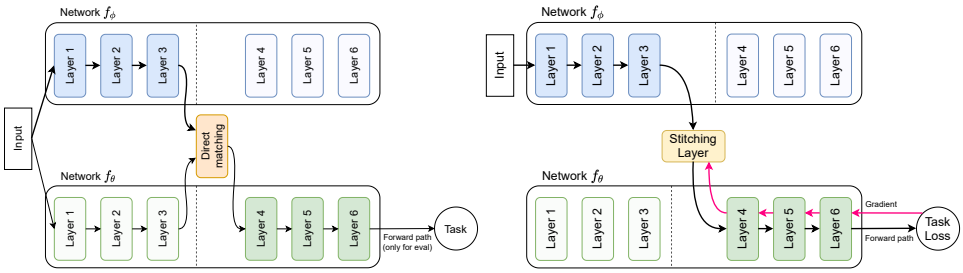
Definition 3.2 (Frankenstein learning task)**.** *Given two pretrained neural networks f_θ, f_ϕ, a task given by a labeled dataset $D = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y} : i = 1, \ldots, N\}$ and a loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, we will call the* Frankenstein learning task at layer L *the task of finding the stitching layer $S : \mathcal{A}_{\theta,L} \to \mathcal{A}_{\phi,L}$ which minimizes the total value of the loss $\mathcal{L}$ on the dataset D.*

Frankenstein is an evaluation framework The Frankenstein learning task primarily provides an *evaluation framework* for matching representations, and it does not tie our hands on how the transformation function S is produced. As an evaluation metric, we can either use the mean loss value of the Frankenstein learning task on an evaluation set, or any other appropriate metric (e.g., accuracy in the case of classification tasks).

The class of stitching layers — convolutional representations Our toolset is generic, but here we will only consider convolutional vision architectures. In this context, the activation spaces $\mathcal{A}_i$ have the structure of rank-3 tensors $\mathbb{R}^{w \times h \times c}$, where w and h are the spatial width and height and c is the number of feature maps. We will only consider linear stitching layers of the form $M : \mathbb{R}^c \to \mathbb{R}^c$ (1×1 convolutional layers). When formulating least squares problems on N many activation tensors, we always mean the least squares problem in $\mathbb{R}^c$, which in practice means reshaping the $N \times w \times h \times c$ tensor to a $Nwh \times c$ matrix.

4 Methods to solve the Frankenstein learning task

Here we line up several possibilities to solve the Frankenstein learning task presented in the previous section. We distinguish two types of approaches, whose side-by-side analysis is at the heart of our paper: **(1) Direct matching**: utilize only the outputs of representation maps; **(2) Task loss matching**: utilize the loss of the Frankenstein learning task. Figure 1 illustrates the difference between the two approaches schematically.



(a) Direct matching of representations. (b) Task loss matching of representations.

Figure 1: Illustrating two types of matching approaches. **(a)** Direct matching: first align the representations using only the representations themselves, and use the forward path on the task map of Model 2 — with the transformed activations — only to evaluate the performance of the stitched model. **(b)** Task loss matching: the forward path from the input starts on Model 1, then through the stitching layer it continues on Model 2, then task loss gradients are backpropagated to update the weights of the stitching layer.

4.1 Task loss matching

From the exposition of the Frankenstein learning task a training method naturally follows: utilize the loss function in the Frankenstein learning task itself, and train an appropriately parametrised stitching layer with backpropagation (leaving the other parts of the Frankenstein network fixed).

We use the outputs of Model 2 as a soft label [6] and define the task loss with cross-entropy. Another meaningful option would be to use the original training objective (the one that was used for the training of Model 1 and Model 2, e.g., the one-hot labels in a classification task). Both are reasonable choices as tasks, one corresponding to the stitched network imitating Model 2, the other to solving the task itself. We have experimented with both losses and found very high correlation, with no difference regarding our main observations presented in the paper. We often present performance in terms of relative accuracy compared to Model 2, because it is easier to interpret than cross-entropy.

4.2 Direct matching methods

Due to their close connection to similarity indices of representations, we also consider methods that arrive at the transformation S by using only the representations themselves, namely, utilizing only the outputs of the representation maps of Model 1 and Model 2.

Least squares methods Given $A, B \in \mathbb{R}^{n\times p}$, and a class of transformations $\mathcal{C} \subseteq \mathrm{Hom}(\mathbb{R}^p, \mathbb{R}^p)$, we will consider least squares problems of the following kind: find an optimal $M_o \in \mathcal{C}$, such that

$$\|AM_o - B\|_F = \min_{M\in\mathcal{C}} \|AM - B\|_F. \tag{1}$$

We consider three variants of this problem: 1. arbitrary linear maps: $\mathcal{C} = \mathrm{Hom}(\mathbb{R}^p, \mathbb{R}^p)$, 2. orthogonal transformations: $\mathcal{C} = O(p)$, 3. linear maps in $\mathrm{Hom}(\mathbb{R}^p, \mathbb{R}^p)$ of rank at most k: $\mathcal{C} = \Sigma_k$.

In each case, there is a closed form expression relying on singular value decomposition. For $\mathcal{C} = \mathrm{Hom}(\mathbb{R}^p, \mathbb{R}^p)$, the minimal value of (1) is obtained through an orthogonal projection in the space of matrices [16],

$$M_o = M_{LS} := A^\dagger B, \tag{2}$$

where $A^\dagger$ denotes the Moore-Penrose pseudoinverse of A. We will refer to this M_{LS} as the *least squares matching*. For results of this type, see Section 5.

In the case $\mathcal{C} = O(p)$, problem (1) is also known as the *orthogonal Procrustes problem*, see e.g. [3]. If $A^T B = USV^T$ is a singular value decomposition, then the optimal orthogonal map is $M_o = UV^T$.

Finally, the case $\mathcal{C} = \Sigma_k$ is also known as *reduced rank regression* [7]. In terms of the SVD of $AM_{LS} = USV^T$, the optimal rank k map M_o is given by $M_o = M_{LS}V_kV_k^T$, where V_k consists of the first k columns of V. For reduced rank matching results, see Section 7.

Sparse matchings It is known [21] that representations are not fully local, that is, matching cannot be achieved by permutation matrices. High quality sparse matchings between neurons can be interpreted to imply "slightly distributed" representations [10].

To achieve sparsity, a natural approach is to add the regularization term $\|M\|_1 = \sum |m_{ij}|$ to the loss function, where M is the transformation matrix. Then one can consider the L1 regularized versions of 1) task loss matching, 2) direct, least squares matching (Lasso regression, as was used in [10]):

$$\mathcal{L}(A, B) = \|AM - B\|_F + \alpha \cdot \|M\|_1, \tag{3}$$

where A, B denote the activations of the two models, and M denotes the stitching layer. See Section 8 for these experiments.

5 Matching representations in deep convolutional networks

In this section, we now continue our exposition by presenting a series of experiments regarding matching neural representations. Our results in this section can be summarized in the following statement:

Neural representations arising on a given layer of convolutional networks that share the same architecture but differ in intialization can be matched with a single affine stitching layer achieving close to original performance on the stitched network.

5.1 Experiment 1: Matching with least squares and task loss matching

In this experiment, we compare the performance of direct matching and task loss matching. We take two networks of identical architectures, but trained from different initializations. We perform a direct matching as well as a task loss matching from Model 1 to Model 2. We then compare the performance of the stitched network to the performance of Model 2.

We conducted experiments on three different convolutional architectures: a simple, 10-layer convnet called Tiny-10 (used in [8]), on a ResNet-20 [5], and on the Inception V1 network [20]. These three networks are rather standard and represent different types of architectures. Tiny-10 and ResNet are trained and evaluated on CIFAR-10, Inception V1 is on the 40-label CelebA task [11]. Further training details follow common practices and are detailed in Appendix A.2.

Figure 2 summarizes the results: task loss matching achieves near perfect accuracy on ResNet-20, and also succeeds in aligning representations on Tiny-10 without collapsing accuracy. While the direct matching significantly underperforms the task loss matching and collapses performance on several layers, it is still remarkable that direct matching performs fairly well in many cases: e.g., on the layers between Layer1.0 and Layer2.1 on ResNet-20 or almost every case on Tiny-10. For further training details, see Appendix A.2.

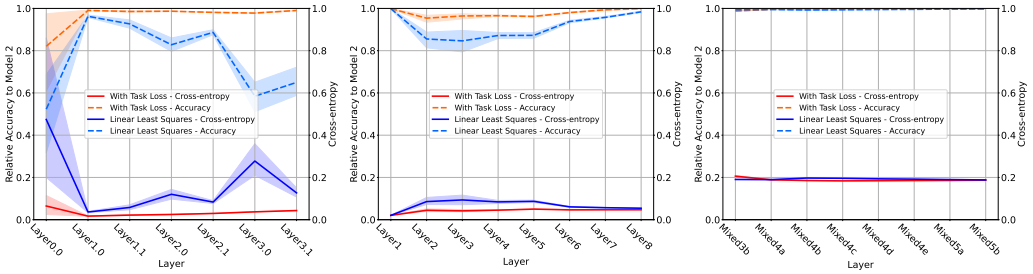
In the case of task loss matching, we found the most stable results when initializing from the least squares matching (2). See Section 8 on further details on how the initialization influences performance. We also evaluated the Procrustes matching described in Section 4.2, and found that on later layers, they achieve the same order of accuracy as the linear least squares matching, however on the earlier layers we observed that there is a more significant difference, see Figure 7.

5.2 Experiment 2: Networks of different width

This experiment focuses on the effect of network width on stitching performance. Matching happens with cross-entropy to Model 2 output as the loss function. Transformation is initialized to the optimal least squares matching. Performance is measured on the original classification task, relative to Model 2 performance. Figure 3 summarizes the results on ResNet-20 with different widths. The base level of relative accuracy is already very high, but a clear trend of performance increasing with width is still visible.

6 Similarity indices and matching representations

We now turn our attention to the relation of similarity and matchability of representations. Finding an appropriate similarity index for neural representations is an active area of research [4, 8, 13, 17]. In



(a) Results with ResNet-20. (b) Results with Tiny10. (c) Results with Inception v1.

Figure 2: Performance of Frankenstein networks stitched together at specific layers (horizontal axis) measured in relative accuracy and cross-entropy (left and right vertical axes, respectively). This experiment compares two methods: Task Loss Matching and Linear Least Squares (a direct Matching method). Results are averages of 10 runs, the bands are representing standard deviations.

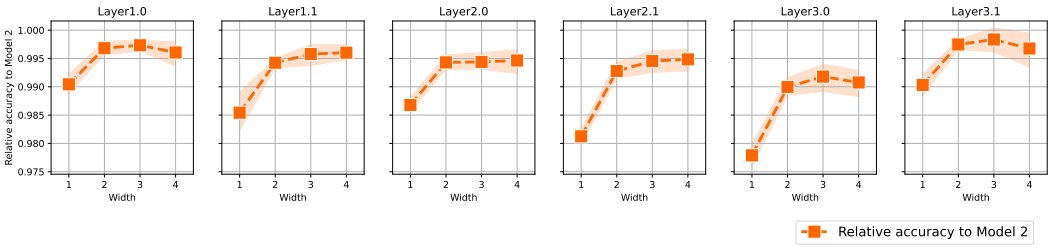


Figure 3: Relative accuracy values with ResNet-20 nets of different width. x-axis shows width multiplier compared to baseline network. Plots correspond to different layers with names denoted at the top. Averages of 10 runs, bands show standard deviations.

the pursuit of a suitable notion, one seeks to capture the right invariance and equivariance properties to provide a tool with theoretical guarantees and practical applicability. Our brief summarizing statement in this section acts as a warning for the end-users of statistical similarity indices:

By their nature, similarity indices depend on the task only indirectly, and their values are not necessarily indicative of task performance on a stitched network.

6.1 Similarity indices

Given two sets of activations $A = (a_i)_{i=1}^N, B = (b_j)_{j=1}^N$, in the activation spaces $a_i \in \mathcal{A}_1 = \mathbb{R}^s, b_j \in \mathcal{A}_2 = \mathbb{R}^r$, a *similarity index* $s(A, B) \in [0, 1]$, measures the similarity of the two data representations, i.e. it is a statistic on the observation space $s : \mathcal{A}_1 \times \mathcal{A}_2 \to [0, 1]$. We now briefly review the similarity indices under consideration. First, the least squares matching M_{LS} defined earlier can be used to define a statistic that can be used as a similarity index [8]:

Definition 6.1. The coefficient of determination *of the best fit is given in terms of the optimal least squares matching* $M_{LS} = A^\dagger B$ *as:*

$$R^2_{LR}(A, B) = 1 - \frac{\|AM_{LS} - B\|_F^2}{\|B\|_F^2}.$$

Let k, l be positive definite kernels on a space $\mathcal{X}$ and let $K_{ij} = k(x_i, x_j)$ and $L_{ij} = l(x_i, x_j)$ be the Gram matrices of a finite sample $x_i \in \mathcal{X}$, $i = 1, \ldots, m$ drawn from a distribution on $\mathcal{X}$. The (empirical) *alignment of the Gram matrices* K, L was defined in [2] as

$$\mathrm{KA}(K, L) = \frac{\langle K, L \rangle_F}{\|K\|_F \cdot \|L\|_F}.$$

The importance of centering the kernels in practical applications was highlighted in [1]. Given a Gram matrix K, denote the *centered Gram matrix* by $K^c := CKC$, where $C = I_m - \frac{\mathbf{1}\cdot\mathbf{1}^T}{m}$ is the centering matrix of size m. The *Centered Kernel Alignment of the Gram matrices* K, L is defined [1] as

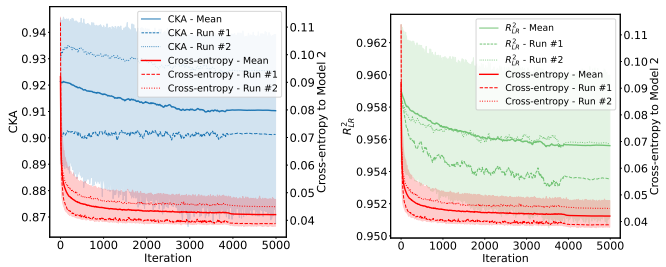


Figure 4: CKA, L2 and cross-entropy values over the training iterations of Task Loss matching started from the optimal Least Squares matching. Tiny-10 network, stitching at Layer 3. Bold lines are averages of 10 runs, with bands representing standard deviations; dotted and dashed lines correspond to two specific runs to show the individual characteristics of the training of the stitching layer. Further experiment details are in Appendix A.5.

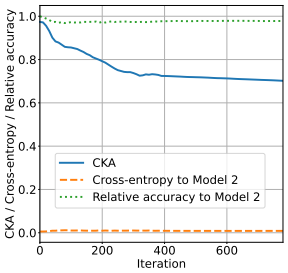


Figure 5: CKA, relative accuracy and cross-entropy values over the course of training for a Frankenstein task with a loss that explicitly penalizes CKA value while maintaining the model performance.

$\mathrm{CKA}(K, L) = \mathrm{KA}(K^c, L^c)$. If the kernels are linear on their respective feature spaces, and if X, Y are samples of the distribution in these feature spaces, then $\mathrm{CKA}(K_X, L_Y) = \mathrm{KA}(K_{XC}, L_{YC})$, using that $K^c_X = K_{XC}$. In particular, in the case of linear kernels CKA is also referred to as the *RV coefficient* [18]. Furthermore, if X, Y are the *centered* matrices representing the samples of the distribution in the feature space, CKA can be given by the following definition:

Definition 6.2. *The* empirical (linear) Centered Kernel Alignment *or* RV coefficient *of the samples* X, Y *centered in feature space is*

$$\mathrm{RV}(X, Y) := \mathrm{CKA}(K_X, K_Y) = \frac{\|X^T Y\|_F^2}{\|X^T X\|_F \cdot \|Y^T Y\|_F}.$$

In the context of similarity indices, the probability distribution describes the data distribution in $\mathcal{X}$, the feature spaces are the activation spaces and the feature maps are the representation maps $R : \mathcal{X} \to \mathcal{A}$. As a measure of similarity of representations, CKA was introduced and used successfully in [8].

The authors in [8] also considered other similarity indices such as CCA, SVCCA, PWCCA [13, 17] in their experiments, but they found that the other similarity indices do not succeed in identifying similar layers of identical networks trained from different initializations, whereas CKA manages to do so. For this reason, we chose to focus our attention on these two indices.

6.2 Experiment 3: Similarity indices and task performance

In this set of experiments, we monitor the CKA and R^2_{LR} indices throughout the training process of the stitching layer trained with task loss matching. We measure these values between the Model 2 activations and the transformed activations of Model 1. Figure 4 shows the results. We observe that CKA has a relatively large variance from the first step of the training process and is not predictive with respect to the performance on the task of the stitched network.

This experiment also highlights that R^2_{LR} can correlate inversely with the performance: as the training proceeds with decreasing cross-entropy, the R^2_{LR} values also decrease. Indeed, since we initialized the task loss training from the optimal least squares matching, the distances naturally must increase under any training process. However, this highlights the fact that the least squares matching is ignorant of the decision boundaries in a neural network.

6.3 Experiment 4: Low CKA value with high accuracy

We now show that it is easy to align representations in a way that results in high task performance while the CKA value at the alignment is small. The setup is the following: we take one trained model and stitch it with *itself* utilizing a linear stitching layer initialised with the identity mapping. With this initialisation we essentially reproduce the original model, thus, this trivially results in a Frankenstein network with the same performance as the original model. We train this stitching layer with an additional loss term that penalizes the CKA value at the stitching, meanwhile we maintain

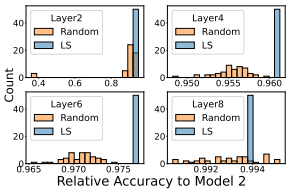


Figure 6: Relative accuracies by initialization type.

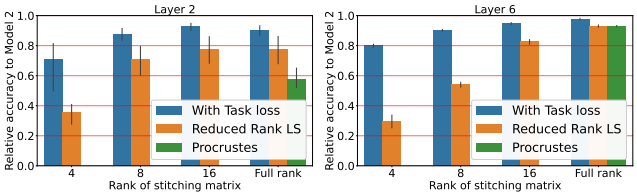


Figure 7: The performance of the low-rank analogues of least squares and task loss matchings in terms of relative accuracy.

the task performance with the usual cross-entropy term. We use the CKA value itself as the penalty as it is differentiable by definition and thus, suitable for gradient-based optimization.

Figure 5 shows the outcome of such a training process: the relevant values over the course of this training: we find that the CKA value drops by a large amount (achieving lower than 0.7), whereas the accuracy remains essentially unchanged.

Discussion Assuming that a similarity index s is invariant under a given class $\mathcal{C}$ of transformations, it is not surprising that the high value of a similarity index $s(A, B)$ does not indicate the relative performance of the activations A and B: indeed, given a set of activations A, $s(AF, A) = 1$ for any $F \in \mathcal{C}$, while the task loss of the transformed activations AF typically varies greatly. On the other hand, it is somewhat surprising that high performance matchings can have a significant drop in the values of meaningful similarity indices such as CKA.

7 Low rank representations — a gap between SVD and task loss matching

In this experiment, we compare the effectiveness of the low-rank analogues of the above presented least squares matching and task loss matching. More precisely, we apply SVD directly in the activation space to obtain the low-rank least squares matching, and use a *bottleneck* stitching layer to realise a low-rank matching with the task loss.

Figure 7 depicts the results for two selected layers of Tiny-10 with prescribed ranks 4, 8, and 16. (We observed similar results for the other layers, and also for ResNet-20.) It is clear that task loss matching outperforms least squares matching in the activation space when the rank of the transformation is fixed. This suggests that the objective for the low-rank approximation of explaining the most of the variance in the activation space (i.e., the objective of SVD) is not satisfactory to attain the highest possible performance with a prescribed rank. Ultimately, this suggests that similarity indices that use SVD directly in the activation space — either as a preprocessing step or as another key element in the computation — might fail to capture certain aspects of the representation. Details and figures for the other layers are in Appendix A.

8 Structure of stitching layers: uniqueness, mode connectivity, sparsity

The previous experiments show that the stitching layer can be trained to achieve the same performance as the original Model 2. What is the structure of the obtained stitching matrix? In this chapter, we explore the question of sensitivity with respect to initialization, uniqueness properties of the stitching layer and the relationship of direct and task loss matching with sparsity.

Experiment 5: Dependence of accuracy on initializations Are stitching layers obtained from different initializations the same? A crude measure of this is the performance of the stitching layer on the stitched network. In this experiment, we fix two identical network architectures trained from different initializations. We train the stitching layer on task loss from 50 random and 50 optimal least squares initializations, and compare their performance. We found that in most cases the stitched networks achieve the same order of accuracy, although not always; the training process might get stuck at poor local minima. However, the optimal least squares matching initialization performs consistently well. Figure 6 shows a histogram of the results. For further details, see Appendix A.

Experiment 6: Linear mode connectivity of transformation matrices We also investigated the uniqueness properties of the stitching layer by examining the linear mode connectivity of the found

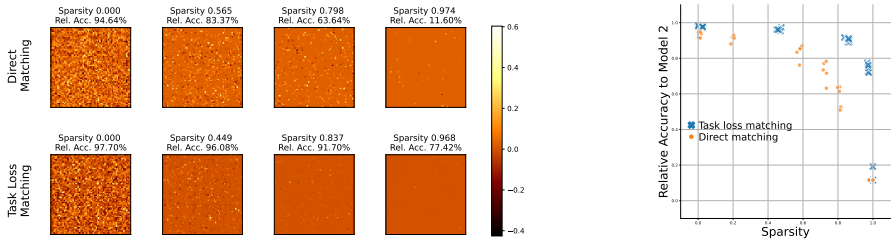


Figure 8: Comparison of stitching matrices and the relative accuracy of L1-regularized direct and task loss matching. To achieve sparsity, we set every entry of the stitching matrix to zero when the absolute value of the entry is below a certain threshold, 10^{-4}. **Left:** Examples of stitching matrices with different sparsity. **Right:** The degradation of relative accuracy with respect to increasing sparsity is significantly less steep with task loss training. More results and details are in Appendix A.8.

transformations. In particular, on a given layer of two trained copies of Tiny-10, we trained stitching transformations from different random initializations. We only kept the transformations achieving a relative accuracy of at least 80 % – there is no reason to expect mode connectivity properties where there is a large difference in relative accuracy. We then evaluated the relative accuracy of the stitched network with the transformation matrix $M_\lambda = \lambda M_1 + (1-\lambda) M_2$ for pairs of transformation matrices M_1 and M_2, and different values of $\lambda \in [0, 1]$. The results showed that mode connectivity holds on most layers, i.e. the convex combination of the stitching layers perform similarly. However, on the first three layers, we noticed an unforeseen lack of mode connectivity, even though the relative accuracies of the corresponding transformations were high. For further details, see Appendix A.7.

Experiment 7: Sparsity — direct vs. task loss matching We are interested in how sparsity of the stitching layer influences performance of the stitched network. In this experiment, we achieve sparsity by adding an L1-regularizing term to the training as detailed in Section 4.2, similarly to the experiments in [10]. We perform both an L1-regularized task loss matching as well as a direct loss matching with loss term (3). Figure 8 shows the dependence of relative accuracy on sparsity compared for task loss and direct matching as well as a sample of the stitching matrices. These experiments show that even though it is possible to achieve a certain degree of sparse alignment of the neurons, the relationship between two representations is typically not of a combinatorial nature, and much can be gained by considering more complex relationships between the two layers. Furthermore, the difference in the relative accuracy of direct and task loss matching is notable for higher values of sparsity. For further details on this experiment, we refer to Appendix A.8.

9 Limitations

The performance of the stitched solution is inherently just a lower bound due to possible local minima. The current study is restricted to well-established convolutional architectures. Our observations on network width is analysed only in isolation.

10 Conclusions

We studied similarity of neural network representations in conjuction with the notion of matchability. We demonstrated the matchability of representations in convolutional networks through several experiments and analysed the properties of the stitching transformations with respect to the task performance. Regarding similarity notions, we provided a novel perspective which incorporates the performance on a task; a perspective which is yet unexplored in the field. We pointed out weaknesses of popular similarity notions that end-users might encounter.

Broader impact

Our research is of foundational nature and advances machine learning at a fairly general level. While the authors do not foresee that the work herein will have immediate adverse ethical or societal consequences, such analysis of representation learning could help a malicious user avoid detection that is based on representational similarity; on the other hand, it could move forward the field towards more transparent and interpretable machine learning models.

References

[1] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13(1):795–828, 2012.

[2] Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz Kandola. On kernel-target alignment. In *Advances in Neural Information Processing Systems 14*. Citeseer, 2001.

[3] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, fourth edition, 2013. ISBN 978-1-4214-0794-4; 1-4214-0794-9; 978-1-4214-0859-0.

[4] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita, editors, *Algorithmic Learning Theory*, pages 63–77, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31696-1.

[5] 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)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[7] Alan Izenman. Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, 5:248–264, 06 1975. doi: 10.1016/0047-259X(75)90042-1.

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

[9] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. *International Journal of Computer Vision*, 127, 05 2019. doi: 10.1007/s11263-018-1098-y.

[10] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? In *International Conference on Learning Representation (ICLR '16)*, 2016.

[11] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[12] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=Fmg_fQYUejf`.

[13] Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5732–5741, 2018. URL `https://proceedings.neurips.cc/paper/2018/hash/a7a3d70c6d17a73140918996d03c014f-Abstract.html`.

[14] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *CoRR*, abs/2008.11687, 2020. URL `https://arxiv.org/abs/2008.11687`.

[15] 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`.

[16] R. Penrose. On best approximation solutions of linear matrix equations. *Proc. Cambridge Philos. Soc.*, 52:17–19, 1956. ISSN 0008-1981.

[17] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *NIPS*, 2017.

[18] Paul Robert and Yves Escoufier. A unifying tool for linear multivariate statistical methods: The RV-coefficient. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 25(3): 257–265, 1976.

[19] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

[21] Liwei Wang, Lunjia Hu, Jiayuan Gu, Yue Wu, Zhiqiang Hu, Kun He, and John Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9607–9616, Red Hook, NY, USA, 2018. Curran Associates Inc.

[22] John Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Similarity analysis of contextual word representation models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4638–4655, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.422. URL `https://www.aclweb.org/anthology/2020.acl-main.422`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Yes, we dedicate a separate section for that: Section 9. (Although due to page limit it is now short, hopefully we can extend it.)
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Yes, briefly in the Broader Impact section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We only reference theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The source code is public. The instructions to reproduce the results are in the readme file.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The Appendix specifies the training details according to the standards of the field. The source code is public.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars for all our experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Appendix A.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Code licenses are part of the repository.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]