On the Direct Alignment of Latent Spaces

Zorah Lähner and Michael Moeller
University of Siegen
Hölderlinstr. 3, 57076 Siegen, Germany
zorah.laehner,michael.moeller@uni-siegen.de

Abstract

With the wide adaption of deep learning and pre-trained models rises the question of how to effectively reuse existing latent spaces for new applications. One important question is how the geometry of the latent space changes in-between different training runs of the same architecture and different architectures trained for the same task. Previous works proposed that the latent spaces for similar tasks are approximately isometric. However, in this work we show that method restricted to this assumption perform worse than when just using a linear transformation to align the latent spaces. We propose directly computing a transformation between the latent codes of different architectures which is more efficient than previous approaches and flexible wrt. to the type of transformation used. Our experiments show that aligning the latent space with a linear transformation performs best while not needing more prior knowledge.

1 Introduction

When training a neural network, the goal is for the network to find meaningful, intermediate representations that capture the data distribution so well it also holds information about new instances. These intermediate representations are often called latent spaces and compress the relevant information into a lower dimensional vector form. Normally the network is not restricted in the composition of these vectors but the assumption is that a successful training will build a latent space that is well suited to represent the task. This implies the possibility that different training runs and even different architectures might lead to similar latent space geometries when set up in similar domains but due to the complex energy landscape and the existence of numerous local minima the absolute value of latent vectors cannot be directly compared. This gives rise to the question of if and what properties are preserved in-between the latent spaces of well-trained networks, and whether we can make use of this information to be able to re-use latent spaces of pre-trained networks.

Some recent literature proposed that the transformation between latent spaces for similar tasks is an isometric one [17]. As a result, they proposed to lift the latent codes into a representation that is invariant under rotations and, thus, removes the influence of chosen hyper-parameters and initialisation that lead to different realisations of the isometric space. While this leads to very impressive results for zero-stitching latent spaces together, the change of representation forces a new network to be trained for any down-stream application.

In this work, we show that an invariant representation is not necessary because it just as straightforward to directly compute the transformation between latent spaces to align them. The latent space can then be directly re-used without any need for training a new network on a new representation, and it also gives flexibility in using as much correspondence information as exists and to constrain the transformation in meaningful ways. Our experiments show that while a rigid transformation, composed of a rotation and translation, does give a good alignment of the latent spaces and still meaningful results on down-stream tasks, a linear transformation does a much better job while also being easier to optimise.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

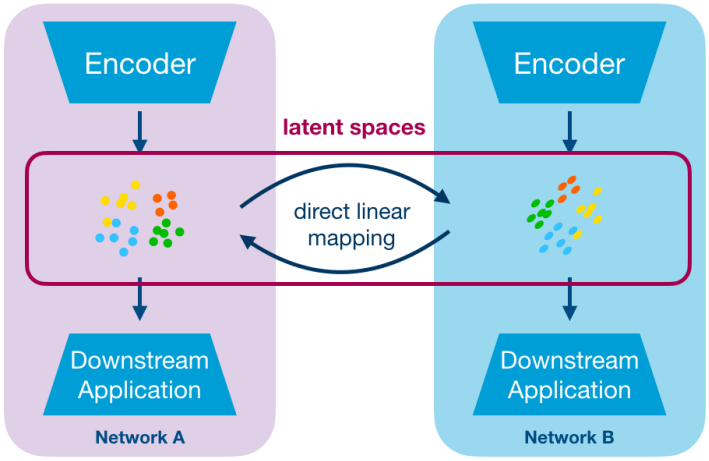


Figure 1: We propose that the latent spaces of semantically similar applications, even for different network architectures, can be aligned by a linear transformation. Linear transformations are flexible but easy to optimise and, thus, provide a useful framework to reuse pre-trained networks without any need for additional training.

Contributions. We propose the theory that semantically related latent spaces even of very different network architectures are related by a linear transformation. Our experiments show that the more general linear transformation outperforms a rigid one (which has been proposed in previous work [17]) while still being efficient to optimise. Additionally, a direct alignment of latent spaces is very flexible w.r.t. the prior knowledge on corresponding elements of the domain, and can be applied to the entire training set, based on only a few selected anchors, or even without any given exactly matching elements.

2 Related Work

The Geometry of Latent Spaces The question of the geometry of latent spaces has been often discussed in literature [1, 14]. It has been established that neural networks tend to converge to similar, stable minima [5, 18] and, thus, the geometry of the latent space also shows similarities [19, 22, 24]. For example, the semantic clusters in self-supervised learning are stable in-between different trainings and architectures [2], and [9] showed that the filters learned by convolutional neural networks do not significantly differ between similar applications. While the theory is only established for very specific cases, this already gives rise to the question whether once-learned latent spaces can be easily reused in new applications and what prior knowledge is needed for such applications. In [17] Moschella et al. proposed that similar architectures on semantically comparable data learn latent spaces that are rigidly transformed versions of each other. As a result, they build a rigidly-invariant representation that lifts each latent code onto its Eulidean distance to each element of a chosen anchor set in the latent space, called relative representation. This concept can be extended to other measures of similarity than distance [6]. However, none of these measures are invariant under linear transformations [6]. In this work, we propose the theory that linear transformations are able to capture and align the latent spaces in many different settings with the additional advantage that linear transformations are easy to optimise for without the need to (re)train a network.

Reusing Existing Latent Spaces With the existence of a huge variety of already trained networks to download on the internet, the question arises to what extent and with what prior knowledge these existing models can be reused for new applications. Transfer learning is probably the most common method which takes the pre-trained weights for one tasks and continues training for a short time on a different by tasks by just fine-tuning all or a subset of the network weights [4, 25]. However, joint representations can also be learned to allow a free exchange of information between different networks instead of just reusing the existing weights for faster training of a new task. For example, agents with different network architectures and based on different training data can learn a joint

communication protocol in a self-supervised way [16] and robots can adapt to new tasks by aligning latent spaces in an unsupervised way [27].

Instead of improving downstream applications, knowledge about the similarity and correspondence of latent spaces can also be used to gain knowledge to analyse the network structures. The concept of stitching trains a separate network to align the latent spaces of layers in different networks to analyse their similarity in behaviour and learned intermediate representations but is more focused on the theoretical insights than reusing the latent spaces due to the need for careful training these networks [15, 21, 7]. A survey of how similarity of representation can be measured has been posted in [13, 12].

For directly reusing latent space information [17] proposed to lift the latent vector into a higher dimensional space by computing distances to a fixed set of anchors which makes them isometry invariant. Another line of work designed components of networks in a way that makes them reusable in different applications [10]. However, all these approaches require the training of specifically designed networks for re-usability. Most similar to ours is the work of [11] in which the latent spaces are aligned by a linear transformation but the transformation is computed by taking the eigendecomposition of the fully-known latent space whereas we can do with much less information. In this work, we will explore the possibilities of aligning latent spaces of arbitrary networks without any retraining.

3 Method

Let $\mathcal{N}, \mathcal{M}$ be two neural networks operating on the same domain, trained on training data from the same distribution, and generating latent spaces $L_\mathcal{N}$ and $L_\mathcal{M}$, respectively. The goal is to align $L_\mathcal{N}, L_\mathcal{M}$ such that it is possible to reuse the latent embedding of one network for the other. To this end, we propose that using a linear transformation between the latent spaces is sufficient to achieve an alignment tight enough to be useful in practice. This is based on the observation that latent spaces do often show similarity in latent geometry [24] and result in rigidly transformed versions of each other [17]. We will evaluate our direct alignment approach against two proposed solutions from literature: stitching latent spaces together by learning the transformation function [15] (see Section 3.2), even though this was proposed for analysis of learned features, and relative representations [17], which build a new embedding based on distances to anchor points (see Section 3.3).

3.1 Direct Latent Alignment

Based on the observation of [17] and [24] that latent spaces tend to have a similar structure, we propose the alignment of $L_\mathcal{N}$ and $L_\mathcal{M}$ by optimising for the optimal linear or rigid transformation based on a chosen energy. Notice that while the assumption is the same, [17] lifts the latent space into a different representation which requires retraining at least part of the network. Our approach is able to directly operate on the existing latent spaces and can reuse all existing network parts. We optimise the following energy:

$$\mathcal{T} = \arg\min_{T \in S} E(L_\mathcal{N}, L_\mathcal{M}, T), \tag{1}$$

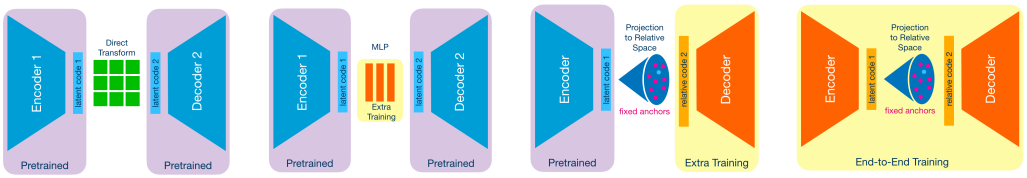
where in our experiments S is either the Euclidean group $E(n)$ or $\mathbb{R}^{n \times n}$ (but other choices are possible), and E describes an energy between the latent spaces. The main choice will be to optimise the alignment of known pairs (x, y) of corresponding latent vectors in some given set $X \subset L_\mathcal{N} \times L_\mathcal{M}$:

$$E(l_\mathcal{N}, l_\mathcal{M}, T) = \sum_{(x,y) \in X} \|x - Ty\|_2^2. \tag{2}$$

By choosing the squared Euclidean distance, it becomes straightforward to compute the optimal rigid or linear transformation between a set of points. The resulting transformation can easily be applied to new samples and does not require any kind of (re-)training of a network.

3.2 Latent Space Stitching

While restricting the transformation to a rigid or linear transformation is meaningful as we will see in the experiments, the transformation might as well be an arbitrary function. Using a network to convert the learned features of a network into each other has been proposed in [15] to analyse the



(a) Ours (b) Stitching (c) RR Decoder-Only (d) RR End-to-End

Figure 2: Schematic of different approaches to make latent spaces re-usable in an autoencoder setting: (a) ours optimising for a linear or rigid transformation, (b) stitching [15] training an MLP to align latent spaces and (c-d) relative representations (RR) [17] lifting the latent space and then training a new decoder (or training everything end-to-end). Blue/purple indicates pre-trained models that need no modification, green is a single optimisation problem and yellow/orange requires a new training of a network.

similarity of layers but can also be utilised to reuse networks. This is an extension of our approach to fix a rigid or linear transformation to the latent spaces and more flexible in terms of transformation but more demanding in terms of knowledge about the training data since with a sufficiently large network any relation between latent spaces can be represented. We use three-layer MLPs to stitch two latent spaces together:

$$F_s : \mathbb{R}^{l_\mathcal{N}} \to \mathbb{R}^{l_\mathcal{M}}, \qquad F_s(x) = FC^{l_\mathcal{M}}_{k \cdot l_\mathcal{N}}(ReLU(FC^{k \cdot l_\mathcal{N}}_{k \cdot l_\mathcal{N}}(ReLU(FC^{k \cdot l_\mathcal{N}}_{l_\mathcal{N}}(x))))) \tag{3}$$

where $l_\mathcal{N}, l_\mathcal{M}$ are the dimensions of $L_\mathcal{N}, L_\mathcal{M}$ and k is the factor of our hidden dimension and FC^j_i is a fully-connected layer from dimension i to j.

3.3 Relative Representations

Another approach to make latent spaces comparable by lifting them into a relative representation was proposed by Moschella et al. [17]. It is based on the observation that latent spaces learned by similar networks for similar tasks on similar data tends to be related by a rigid transformation. Based on a set of pre-selected anchor elements $\mathbb{A} \subset \mathcal{X}$ the relative representation of any $x \in \mathcal{X}$ is defined as

$$r(x) = \big(\text{sim}(E(x), E(a_1)), \text{sim}(E(x), E(a_2)), \ldots, \text{sim}(E(x), E(a_{|\mathbb{A}|}))\big) \tag{4}$$

where sim is a similarity function between elements of the latent space. In [17] sim was chosen to be the Euclidean distance, as this makes the lifting invariant to rigid transformations, but other measurements are possible and were explored in [6].

While this approach allows stitching together even latent spaces from very different distributions, it requires the careful selection of suitable anchors and a network dedicated to processing the relative representations which has to newly trained, either in an end-to-end fashion or with additional training data for each new application.

4 Experiments

We show in our experiments that aligning latent spaces with a linear transformation is as powerful as previously proposed approaches. To that end, we analyse the results with decreasing amount of knowledge about the training data during alignment and increasing differences in the network architecture and training data. We compare our method (Ours, Section 3.1) against learning a network approximation of the transformation (Stitching, Section 3.2) and using relative representations (RelReps, Section 3.3). For simplicity we choose basic autoencoder networks on a selection of image datasets.

In the first set of experiments in Section 4.2.1 we analyse the influence of the latent size onto the test reconstruction error, assuming the correspondence between the entire training set is given to tune each method (only possible for ours and stitching). This serves as a lower bound for the expected performance. In subsequcent experiments the setup is made harder by reducing the known number of correspondences between training examples (see Section 4.2.2 and Section 4.2.3) and choosing conceptually different architectures to generate the latent spaces (see Section 4.3.1).

4.1 Implementation Details

All experiments are done with Adam with learning rate 0.001 and batch size 64 on either a NVIDIA Geforce RTX 3080 or on a cluster with NVIDIA V100s. All experiments with reported standard deviation were repeated three times.

Network Architectures. The image autoencoder are simple convolutional neural networks with only convolutional and linear layers as well as ReLU activations. The focus is not on overall network performance but on comparing behavior between methods. The stitching networks are 3-layer MLPs with ReLUs and the hidden dimension k is the only parameter. Relative representations introduces a relative projection function onto a higher dimensional latent code but we choose the network architectures as similar as possible except for this step for all methods. We will release the code including exact network architectures used after publication.

Relative Representations We use two variations of the relative representation framework, namely *decoder-only* and *end2end*. For decoder-only the encoder part of an autoencoder (as it was also used for our direct alignment and stitching) is used, then a relative representation based on randomly chosen anchors is produced in the latent space of this encoder, and a new decoder trained for the relative representation. In end2end, the anchors are chosen beforehand and the entire autoencoder with a relative representation in between is trained in an end-to-end manner. This allows the training to find a latent space that works well with the chosen anchors. The encoder and decoder architectures in both cases are identical to the architectures of the normal autoencoder except for the input dimension of the decoder which is adjusted to the anchor size.

4.2 Same Architecture

In this section, we conduct experiments using the same architectures and the maximum amount of knowledge about the used training data, and evaluate how well each method can reconstruct the results given the latent codes from a different training run. This will give a baseline for the quality of results and alignment that is achievable with every approach. We use the entire training set to compute the optimal rigid or linear transformation and to train the stitching network. We use an autoencoder setting on MNIST [8] for these experiments and report the reconstruction error on the test set.

4.2.1 Full Training Correspondence

In this experiment, we use the entire training set to compute the optimal rigid or linear transformation, and to train the stitching network.

test reconstruction error $\pm$ std. dev. $\times 10^{-3}$

	$\mathcal{N}$	$\mathcal{N} \rightarrow \mathcal{M}$				
latent	AE baseline	rigid	linear	stitch $k=1$	stitch $k=2$	stitch $k=4$
2	0.646 ± 3.0	0.899 ± 64.7	0.795 ± 20.0	0.803 ± 33.1	0.721 ± 9.5	0.704 ± 13.6
10	0.514 ± 1.9	0.704 ± 78.6	0.549 ± 12.1	0.552 ± 10.1	0.538 ± 3.6	0.531 ± 03.8
30	0.483 ± 0.6	0.535 ± 20.9	0.492 ± 01.5	0.499 ± 01.9	0.495 ± 0.6	0.493 ± 01.2
50	0.476 ± 0.1	0.536 ± 14.0	0.485 ± 03.0	0.491 ± 02.8	0.486 ± 0.2	0.486 ± 00.7
100	0.473 ± 0.2	0.508 ± 20.9	0.476 ± 00.9	0.481 ± 01.1	0.481 ± 0.7	0.484 ± 01.1
150	0.472 ± 0.2	0.487 ± 09.7	$\mathbf{0.474 \pm 0.2}$	$\mathbf{0.479 \pm 0.5}$	0.481 ± 0.7	0.490 ± 03.6

Table 1: Test reconstruction error of ours and stitching on the MNIST dataset using full training data for alignment. Networks architectures are as similar as possible. For stitching k refers to the factor of dimension increase from latent dimension to hidden dimension in the network. We **bold** the best numbers in the transfer setting of each method.

The results can be seen in Table 1. Performance does not significantly increase after latent dimension 50, and it is clearly visible that the linear transformation is a lot closer to the original architecture performance than a rigid transformation only. This indicates that the rigidity assumption (and therefore usage of Euclidean distances) in relative representations might not be the optimal choice. For the stitching network, $k = 2$ tends to be the best option for higher latent dimensions. It might seem surprising that the in-theory strictly more general non-linear network performs worse than the

linear transformations, but this is the performance on the test set. The network shows very high accuracy on the train set but does not perform as well on the test set which indicates that the restriction to linear transformations is meaningful. For relative representation the correspondence information is given through the anchors but a higher amount of anchors increases the network size significantly. Therefore, we skip relative representations for this setting.

4.2.2 Anchor-Based Alignment

In this setting instead of knowing the correspondence of the entire training set, we assume only a handful of so-called anchor elements with correspondence is given (as coined by [17], see Section 3.3). Anchors are easy to produce for networks which have the same domain as an element can be encoded by two networks to produce corresponding latent codes but are not obvious to obtain for different domains. For this experiment we choose the anchors randomly from the training set but we explore some more structured approaches in Section **??**.

We first do a baseline experiment and hyperparameter tuning for relative representations on MNIST, see Table 2. The performance is not on-par with the results of Table 1 but this expected due to using less prior knowledge. Interestingly, increasing the number of anchors does not significantly increase (and in some cases even decreases) the performance which might be due to more anchors introducing more noise into the relative representation or badly chosen anchors.

test reconstruction error $\pm$ std. dev. $\times 10^{-3}$

		AE baseline	decoder-only		end2end	
latent	anchor	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N} \to \mathcal{M}$	$\mathcal{N}$	$\mathcal{N} \to \mathcal{M}$
2	10	0.646 ± 3.0	0.698 ± 0.3	0.845 ± 04.7	0.751 ± 8.3	0.821 ± 11.0
2	50	0.646 ± 3.0	0.698 ± 0.1	0.848 ± 05.8	0.665 ± 2.2	0.700 ± 17.2
10	50	0.514 ± 1.9	0.522 ± 0.6	0.616 ± 00.8	0.528 ± 0.3	0.548 ± 00.6
10	100	0.514 ± 1.9	0.521 ± 0.2	0.618 ± 00.8	0.529 ± 2.1	0.577 ± 14.8
30	100	0.483 ± 0.6	0.484 ± 0.2	0.575 ± 02.1	0.494 ± 0.7	0.505 ± 00.9
30	300	0.483 ± 0.6	0.485 ± 0.2	0.548 ± 00.6	0.494 ± 0.1	0.504 ± 02.0
50	100	0.476 ± 0.1	0.479 ± 0.4	0.839 ± 06.1	0.488 ± 0.8	0.498 ± 02.0
50	200	0.476 ± 0.1	0.479 ± 0.1	0.813 ± 10.1	0.488 ± 0.5	0.499 ± 02.2
100	200	0.473 ± 0.2	0.477 ± 0.3	0.498 ± 00.2	0.485 ± 0.1	$\mathbf{0.491 \pm 01.0}$
100	300	0.473 ± 0.2	0.478 ± 1.4	$\mathbf{0.496 \pm 01.7}$	0.489 ± 0.6	0.495 ± 00.2

Table 2: Test reconstruction error of relative representations with different latent sizes and number of anchors on MNIST. Anchors are chosen at random. AE baseline refers to a normal training without relative representations for reference. decoder-only takes the latent space of AE baseline and trains a decoder on a relative representation of this latent space. end2end trains the relative latent space directly in the autoencoder training.

The second part of the experiments applies our direct alignment and stitching with anchors instead of the full training set. The results are reported in Table 3. Our direct alignment with a linear transformation achieves the best results. The performance of stitching decreases significantly from the full training correspondence case due to the network not being able to generalise to the test set from the small amount of samples. The direct alignment only has a small decrease in performance which is likely due to the stronger assumption on the transformation which captures the change well and, therefore, generalises very quickly with some exceptions where the random anchors are chosen badly. This could, for example, happen if the linear system is underdetermined for which we use the minimum norm solution but it is not clear whether this is meaningful. In future experiments, which are all based on few anchor points, we will not consider stitching anymore as it is not well suited for these settings.

Interesting Note. In all methods that are based on an assumption of rigidity between latent spaces (relative representations decoder-only and rigid direct alignment) and using the base autoencoder, there is a drop in transfer performance visible for latent dimension 50. This is due to all pairings with one specific of the auto-encoders not transferring well and all rigid transfers based on this latent space having very high errors. This indicates while often the rigid assumption is reasonable, there might be some configuration which it cannot capture but linear transformations can.

test reconstruction error $\pm$ std. dev. $\times 10^{-3}$

		$\mathcal{N}$	$\mathcal{N} \rightarrow \mathcal{M}$				
l.	a.	AE baseline	rigid	linear	stitch $k=1$	stitch $k=2$	stitch $k=4$
2	10	0.646 ± 3.0	0.897 ± 61.4	0.810 ± 16.9	0.888 ± 5.8	0.900 ± 11.5	0.902 ± 19.6
2	50	0.646 ± 3.0	0.889 ± 60.2	0.810 ± 29.8	0.909 ± 26.1	0.892 ± 13.9	0.889 ± 14.6
10	50	0.514 ± 1.9	0.705 ± 79.5	0.560 ± 19.2	0.921 ± 48.1	0.909 ± 52.4	0.863 ± 55.7
10	100	0.514 ± 1.9	0.708 ± 80.9	0.554 ± 12.7	0.898 ± 47.2	0.860 ± 42.1	0.795 ± 59.4
30	100	0.483 ± 0.6	0.539 ± 19.6	0.483 ± 02.1	0.813 ± 17.5	0.759 ± 21.9	0.708 ± 15.1
30	300	0.483 ± 0.6	0.536 ± 20.8	0.494 ± 01.9	0.704 ± 15.9	0.665 ± 24.7	0.611 ± 09.7
50	100	0.476 ± 0.1	0.551 ± 17.6	0.500 ± 08.9	0.768 ± 13.5	0.714 ± 11.7	0.673 ± 03.7
50	200	0.476 ± 0.1	0.542 ± 11.3	0.490 ± 04.9	0.703 ± 05.0	0.662 ± 10.8	0.639 ± 20.0
100	200	0.473 ± 0.2	0.522 ± 19.7	0.480 ± 01.5	0.649 ± 12.1	0.625 ± 10.5	0.635 ± 15.0
100	300	0.473 ± 0.2	0.517 ± 24.4	$\mathbf{0.478 \pm 1.2}$	0.615 ± 12.2	0.594 ± 08.9	$\mathbf{0.589 \pm 6.8}$

Table 3: Test reconstruction error of ours and stitching on the MNIST dataset using n corresponding anchor points for alignment. Networks architectures are as similar as possible. For stitching k refers to the factor of dimension increase from latent dimension to hidden dimension in the network.

4.2.3 Geometry-Based Alignment

While constructing anchors is certainly possible in many settings, it does require some knowledge about the training setup or prior changes to incorporate them (in case of relative representations). In the ideal case, we could align the latent spaces without or just minimal knowledge. We propose to do this just by using the class information of the training data. First, we use the latent class means as anchors instead of exact correspondences. Second, we perform iterative closest points (ICP) [3] using the class information by restricting the nearest neighbor step to points of the same class to guide the optimization. These experiments are not possible with relative representations and stitching as their require too many anchors or training examples. Thus, we only evaluate our approach.

test reconstruction error $\pm$ std. dev. $\times 10^{-3}$

	$\mathcal{N}$	$\mathcal{N} \rightarrow \mathcal{M}$			
method	baseline	means		means+ICP	
		rigid	linear	rigid	linear
2	0.646 ± 3.0	0.887 ± 70.5	0.801 ± 019.8	0.937 ± 74.4	$\mathbf{0.958 \pm 68.7}$
10	0.514 ± 1.9	0.707 ± 79.9	0.702 ± 111.4	0.956 ± 29.4	1.062 ± 15.5
30	0.483 ± 0.6	0.730 ± 25.9	$\mathbf{0.659 \pm 003.1}$	0.957 ± 04.2	0.976 ± 27.4
50	0.476 ± 0.1	0.660 ± 4.0	0.747 ± 025.4	0.950 ± 07.5	0.987 ± 12.0

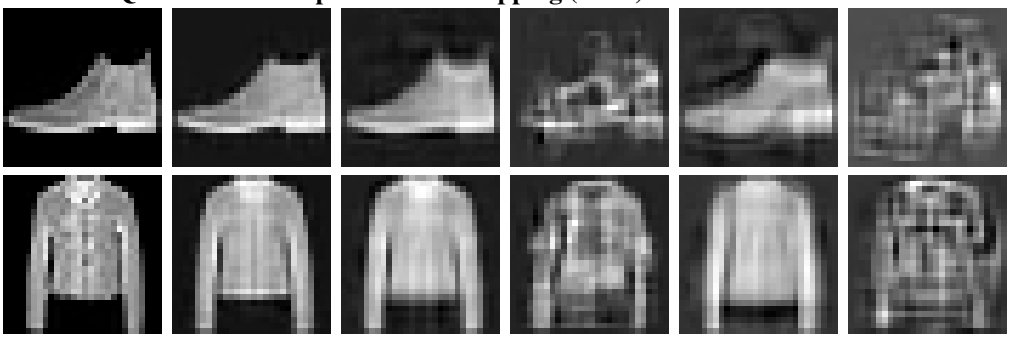
Table 4: Test reconstruction error after alignment of latent spaces with no correspondences and just class-wise labels. Means computes the class latent mean as anchors and means+ICP does iterative closest points with only class-wise nearest neighbors.

The results are reported in Table 4. For low dimensions using the means as anchors leads to a similar performance than using random correspondences. This is expected as the means are guaranteed to cover the majority of the latent space. For higher dimensions the performance lacks behind the random anchors, likely due to the linear system being underdetermined. Since MNIST only contains 10 classes, this means only 10 anchors can be generated in this way. This is a weakness of this approach as the number of classes cannot be manipulated, however, we believe any uniform sampling scheme from the latent space would work as a replacement. Unfortunately, applying ICP on the training latent codes, even when using class information as initialization or for the nearest neighbor step, leads to degration of the accuracy. We believe this might be due to ICP relying on local geometric details for a tight alignment which the latent space does not provide. Thus, a completely prior knowledge-free method to align the latent spaces was not achieved.

4.3 Different Network Architecture

While it might be less surprising that one architecture generates latent spaces of similar structure, we test if this trend transfers to latent spaces of different architectures but trained on the same (Section 4.3.1).

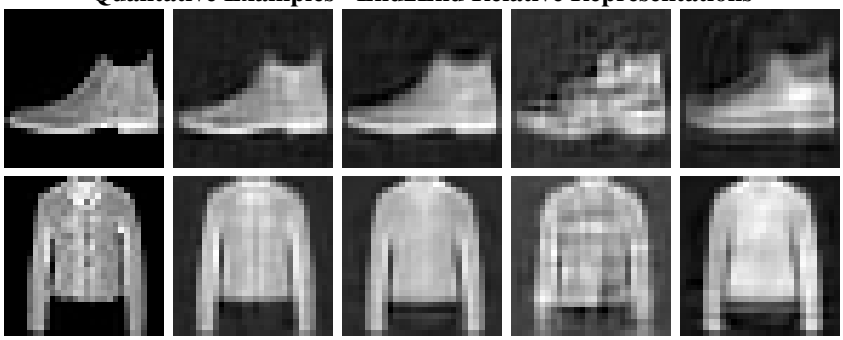
Qualitative Examples - Direct Mapping (Ours) and Relative Decoder



input conv MLP rigid linear decoder

Figure 3: Qualitative examples comparing the change in performance when transferring latent codes between the conv and MLP networks from Section 4.3.1.

Qualitative Examples - End2End Relative Representations



input conv MLP MLP $\rightarrow$ conv conv $\rightarrow$ MLP

Figure 4: Qualitative examples comparing the change in performance when transferring latent codes between the conv and MLP end2end trained relative representation networks from Section 4.3.1.

4.3.1 Anchor-Based Alignment

In this experiment, we train two autoencoders, a convolutional network (similar to the last sections) and an MLP on the FashionMNIST dataset [26]. Then, we use our direct mapping and relative representations to transfer the latent codes between these networks. The results are reported in Table 5 and Table 6. Qualitative results of the transfer are shown in Figure 3 and Figure 4.

test reconstruction error $\pm$ std. dev. $\times 10^{-3}$

		$\mathcal{N}$	$\mathcal{M}$	$\mathcal{N} \rightarrow \mathcal{M}$		
l.	a.	conv	MLP	rigid	linear	decoder
50	100	0.066 ± 3.4	0.094 ± 0.8	1.805 ± 372.4	0.241 ± 8.9	4.00 ± 858.4
100	200	0.045 ± 0.8	0.058 ± 0.6	0.355 ± 010.8	0.162 ± 10.8	$\mathbf{0.712 \pm 11.3}$
		$\mathcal{N}$	$\mathcal{M}$	$\mathcal{M} \rightarrow \mathcal{N}$		
l.	a.	conv	MLP	rigid	linear	decoder
50	100	0.066 ± 3.4	0.094 ± 0.8	0.558 ± 017.0	0.217 ± 9.7	1.363 ± 102.6
100	200	0.045 ± 0.8	0.058 ± 0.6	0.713 ± 064.6	$\mathbf{0.141 \pm 6.5}$	0.745 ± 019.1

Table 5: Test reconstruction error of ours and relative representations decoder on the FashionMNIST dataset when using both a CNN and an MLP as the two base networks. The $\mathcal{N}$ and $\mathcal{M}$ rows are identical in top and bottom and just repeated for reference.

The first interesting observation in the results is that the base performance of the non-relative autoencoder without transfer is better than the relative one and the qualitative results show less sharpness in the CNN. This could indicate that lifting to the relative representation is not loss-free (possibly due to chosen anchors being outliers). Note that we did not use Fourier features in the MLP, thus, the

test reconstruction error $\pm$ std. dev. $\times 10^{-3}$					
		$\mathcal{N}$	$\mathcal{M}$	$\mathcal{N} \to \mathcal{M}$	$\mathcal{M} \to \mathcal{N}$
l.	a.	conv	MLP	conv→MLP	MLP→conv
50	100	0.096 ± 1.7	0.138 ± 1.2	0.298 ± 10.0	0.288 ± 10.2
100	200	0.079 ± 0.9	0.107 ± 2.6	$\mathbf{0.249 \pm 13.2}$	0.304 ± 13.5

Table 6: Test reconstruction error of end2end trained relative representations on the FashionMNIST dataset when using a CNN and an MLP in the base networks. While the original end2end training performs much better than for the non-relative case (Table 5), the performance degrades significantly more when transferring to the latent space of a different architecture.

reconstruction is smoothed [20, 23] but we wanted to keep the influence clean and transfer between networks with varying performance is more interesting.

Overall, the transfer in-between the CNN and MLP architecture works best when using a linear transformation to align the latent spaces while the direct rigid alignment leads to very inaccurate solutions. Relative representations work better than the rigid alignment, even though it is also based on a rigid assumption, and might be able to learn some robust against deviation from rotational changes in the extra training process. The decoder-only transfer fails completely, as is visible in the qualitative examples. It is unclear to us why the results are that bad. It might be a bug in the code but we could not find it.

5 Conclusion

We explored the possibility of directly aligning semantically similar latent spaces with a rigid or linear transformation which opens the option to switch latent codes between already trained networks. The advantage of this approach that it is not necessary to change anything about the network or retrain something, any pre-trained model can be reused and only minimal knowledge about the transformation is needed in the form on sparse anchor points to optimise the matrix. Our experiments include cases where the network architecture is completely different, a CNN and an MLP, but the full extend of this property and its theoretical background needs to be explored in future work. Specifically, we showed that a linear transformation works considerably better than a rigid one which indicates that the assumption used for example in [17] that the latent spaces are rigidly related often does not hold well, and linear transformations are a kind of transformation that can not yet be captured by relative transformations [6].

Acknowledgements. Zorah Lähner is funded by a KI-Starter grant of the Ministry of Culture and Science of the State of North Rhine-Westphalia.

References

[1] G. Arvanitidis, L. K. Hansen, and S. Hauberg. Latent space oddity: On the curvature of deep generative models. In *International Conference on Learning Representations (ICLR)*, 2018.

[2] I. Ben-Shaul, R. Shwartz-Ziv, T. Galanti, S. Dekel, and Y. LeCun. Reverse engineering self-supervised learning. *arXiv:2305.15614*, 2023.

[3] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1992.

[4] S. Bozinovski and A. Fulgosi. The influence of pattern similarity and transfer learning upon the training of a base perceptron b2. *Proceedings of Symposium Informatica*, 1976.

[5] J. Bruck and J. Goodman. A generalized convergence theorem for neural networks. *IEEE Transactions on Information Theory*, 34(5), 1988.

[6] I. Cannistraci, M. Fumero, L. Moschella, V. Maiorca, and E. Rodolà. Infusing invariances in neural representations. *ICML Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML)*, 2023.

[7] A. Csiszárik, P. Kőrösi-Szabó, Ákos K. Matszangosz, G. Papp, and D. Varga. Similarity and matching of neural network representations. In *Advances in Neural Information Processing Systems (NeuRIPS)*, 2021.

[8] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[9] P. Gavrikov and J. Keuper. Cnn filter db: An empirical investigation of trained convolutional filters. *CVPR*, 2022.

[10] M. Gygli, J. Uijlings, and V. Ferrari. Towards reusable network components by learning compatible representations. *AAAI*, 2021.

[11] S. Jain, A. Radhakrishnan, and C. Uhler. A mechanism for producing aligned latent spaces with autoencoders. *arXiv:2106.15456*, 2019.

[12] M. Klabunde, T. Schumacher, M. Strohmaier, and F. Lemmerich. Similarity of neural networks: A survery of functional and representational measures. *arXiv:2305.06329*, 2023.

[13] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning (ICML)*, 2019.

[14] L. Kühnel, T. Fletcher, S. Joshi, and S. Sommer. Latent space geometric statistics. In *Pattern Recognition. ICPR International Workshops and Challenges*, 2021.

[15] K. Lenc and A. Vedaldi. Understanding image representations by measuring their equivariance and equivalence. *International Journal of Computer Vision (IJCV)*, 2018.

[16] M. Mahaut, F. Franzon, R. Dessì, and M. Baroni. Referential communication in heterogeneous communities of pre-trained visual deep networks. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023.

[17] L. Moschella, V. Maiorca, M. Fumero, A. Norelli, F. Locatello, and E. Rodolà. Relative representations enable zero-shot latent space communication. In *International Conference on Learning Representations (ICLR)*, 2023.

[18] R. Mulayoff, T. Michaeli, and D. Soudry. The implicit bias of minima stability: A view from function space. *NeurIPS*, 2021.

[19] B. Neyshabur, R. Tomioka, and N. Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *ICLR*, 2015.

[20] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *NeurIPS*, 2007.

[21] M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan. Revisiting model stitching to compare neural representations. In *Advances in Neural Information Processing Systems (NeuRIPS)*, 2021.

[22] A. L. Smith, D. M. Asta, and C. A. Calder. The geometry of continuous latent space models for network data. *Stat Sci.*, 34(3), 2019.

[23] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[24] A. Tsitsulin, M. M. ad Davide Mottin, P. K. A. M. Bronstein, I. V. Oseledets, and E. Müller. The shape of data: Intrinsic distance for data distributions. *International Conference on Learning Representations (ICLR)*, 2020.

[25] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 3(9), 2016.

[26] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[27] T. Yoneda, G. Yang, M. R. Walter, and B. C. Stadie. Invariance through latent alignment. *Robotics: Science and Systems*, 2022.