Understanding Feature Transfer Through Representation Alignment

Ehsan Imani, Wei Hu, and Martha White

Abstract—Training with the true labels of a dataset as opposed to randomized labels leads to faster optimization and better generalization. This difference is attributed to a notion of alignment between inputs and labels in natural datasets. We find that training neural networks with different architectures and optimizers on random or true labels enforces the same relationship between the hidden representations and the training labels, elucidating why neural network representations have been so successful for transfer. We first highlight why aligned features promote transfer and show in a classic synthetic transfer problem that alignment is the determining factor for positive and negative transfer to similar and dissimilar tasks. We then investigate a variety of neural network architectures and find that (a) alignment emerges across a variety of different architectures and optimizers, with more alignment arising from depth (b) alignment increases for layers closer to the output and (c) existing high-performance deep CNNs exhibit high levels of alignment.

Index Terms—Transfer Learning, Representation Learning, Neural Networks.

1 INTRODUCTION

A common strategy for transfer learning is to first learn a neural network on a source task with a large amount of data, then extract features from an intermediate layer of that network and finally train a subsequent model on a related target task using those extracted features. The premise is that neural networks adapt their intermediate representations—hidden representations—to the source task and, due to the commonalities between the two tasks, these learned representations help training on the target task [1]. Availability of large datasets like ImageNet [2] and the News Dataset for Word2Vec [3] provides suitable source tasks that facilitate using neural networks for feature construction for Computer Vision and Natural Language Processing (NLP) tasks [4], [5], [6], [7].

There is as yet much more to understand about when and why transfer is successful. Understanding the properties of the learned hidden representations and their benefits for training on similar tasks has remained a longstanding challenge [8], [9], [10]. One strategy has been to define properties of a good representation, and try to either measure or enforce those properties. Disentanglement and invariance are two such properties [1], where the idea is that disentangling the factors that explain the data and are invariant to most local changes of the input results in representations that generalize and transfer well. Though encoding properties for transfer is beneficial, it remains an important question exactly how to evaluate the representations that do emerge.

One challenge is that even hidden representations of two neural networks trained on identical tasks appear completely different, and studying the representations requires measures that separate recurring properties from irrelevant artifacts [11]. One direction has been to analyze what abstractions the network has learned, agnostic to exactly how it is represented. Shwartz et al. [12] studied neural networks through the lens of information theory and found that, during training, the network preserves the information necessary for predicting the output while throwing away unnecessary information successively in its intermediate layers. Using representational similarity matrices, [13] found that on synthetic datasets where task-relevance of features can be controlled, learned hidden representations suppress task-irrelevant features and enhance task-relevant features. Neyshabur et al. [14] showed that neural networks trained from pre-trained weights stay in the same basin in the loss landscape. In reinforcement learning, Zahavy et al. [15] explained the success of Deep Q-Networks by visualizing how the learned hidden representations break down the input space in a way that respects the temporal structure of the task. Analyses of NLP models have found linguistic information in the hidden representations after training [16], [17], [18], [19].

Other works have focused on individual features in the learned representations. Saliency maps and Layer-Wise Relevance Propagation that show the sensitivity of the prediction to each unit in the model are popular in Computer Vision and demonstrate the appearance of useful features like edge or face detectors in neural network [20], [21], [22]. In NLP, Dalvi et al. [23] studied the relevance of each unit to an external task or the model’s own prediction.

Our approach builds on recent insights on the importance of the relationship between the features and the labels of common datasets. Label randomization tests by Zhang et al. [24] showed that, even though overparameterized models are capable of memorizing large datasets with random labels, they generalize well when the dataset is properly labeled. This result challenged the traditional view that associated the ability of a model to generalize with its restricted capacity. Another observation of Zhang et al. [24] was the substantial
role of labels on the rate of optimization. Inspired by these findings, Arora et al. [25] introduced a notion of alignment between the inputs and labels that is present in common datasets and is broken once the labels are shuffled. Their data-dependent bounds that leverage this property explain the improved generalization and optimization of linear models or overparameterized neural networks when trained on properly labeled data.

In this work, we highlight a simple insight: alignment emerges in the hidden representations of a neural network and this alignment facilitates training on similar target tasks. We first define alignment, and explain why having a representation that is aligned with labels promotes faster learning. We highlight that, even when training on randomized labels, aligned representations consistently emerge under neural network training. We then show, on a synthetic problem composed of similar and dissimilar tasks, that this alignment improves convergence rates on similar tasks (positive transfer) and slows convergence on dissimilar tasks (negative transfer). Moving away from synthetic problems, we further investigate how widespread the phenomena is where alignment emerges in neural networks. We study several common pre-trained CNNs on the large ImageNet dataset, and find they have high levels of alignment, providing some explanation for the success of transferred CNN features in object recognition. Finally, we test different network depths and layer widths, as well as optimizers, and consistently find (a) that aligned representations emerge across all settings and (b) that alignment is higher in later layers.

2 PROBLEM FORMULATION AND NOTATION

We focus on the transfer setting. We assume there is a source task where data is abundant and a target task with a small number of available data points. A neural network is learned on the source task, and the final layer used as learned features. These learned features are then given as the representation for the target task, on which we learn a linear model with gradient descent on these representations. More generally, a nonlinear function—like another, likely simpler, neural network—could be learned, but we restrict our attention to linear models.

More formally, we consider a dataset with $n$ samples and input matrix $X_{n \times m}$ and label vector $y_{n \times 1}$. We assume we have a (learned) feature function $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^d$ that maps inputs to features, with representation matrix $\Phi_{n \times d}$ where $n \geq d$. Without loss of generality, we replace the bias unit with a constant feature in the matrix to avoid studying the unit separately. We learn a linear model, with weights denoted by $w_{d \times 1}$, learned using the squared error in regression and logistic loss in classification. In regression, $\|\| \cdot \|\|$ is the $\ell_2$ norm. In classification, $y \in \{-1, +1\}^n$ and the logistic loss is $\sum_{i=1}^n \log(1 + \exp(-\phi^T_i w \cdot y_i)) / \log(2)$.

Alignment will be defined in terms of the singular vectors of $\Phi$ and label vector $y$. The singular value decomposition (SVD) of $\Phi$ is $\Phi = U \Sigma V^T$, where $\Sigma_{n \times d}$ is a rectangular diagonal matrix whose singular vectors $\sigma_{1:d}$ that are in descending order and $U_{n \times n}$ and $V_{d \times d}$ are orthonormal matrices whose columns $u_{1:n}$ and $v_{1:d}$ are the corresponding left and right singular vectors. Implicitly, the singular values $\sigma_{d+1:n}$ are zero, giving this thin SVD where $\Sigma$ is rectangular. For a vector $a$ and orthonormal basis $B$, $a^B$ denotes $B^\top a$ or the representation of $a$ in the basis $B$. We will use $r(\cdot)$ to denote the rank of a matrix.

3 BEYOND EXPRESSIVITY

A representation plays two important roles: expressivity and facilitating optimization. The primary focus is typically expressivity, meaning that the representation facilitates learning a simple (linear) function with much of the nonlinearity in the representation. In classification, this has been synonymous with linear separability, where projecting into a new—typically higher-dimensional—space allows for a linear separating hyperplane to be learned.

The set of expressive representations may differ in terms of optimization behavior when learning a linear function on those features. In particular, some representations may improve the convergence rate, requiring fewer updates to learn the linear function. This improved convergence then also seems to improve generalization ability, shown theoretically for SGD by [26].

Let us clarify this point with an experiment before moving to the transfer learning setting. We sampled 1000 points from the UCI CT Position Dataset [27] and created three sets of 1024-dimensional features. The first representation is extracted from a neural network with one ReLU hidden layer. The second one is obtained with sparse dictionary learning [28]. The third one is the representation learned by a two-layer RBF network. All of these representations are nonlinear and high dimensional and a linear model on each one can perfectly fit this data. We verified this by evaluating the closed form least squares solution on the extracted representations.

![Fig. 1: Performance of a linear model on a ReLU neural network representation (red), RBF network representation (purple), and dictionary learning representation (blue). Dashed and solid curves show train and test mean squared errors. We swept over four different step-size values in $\{0.01, 0.1, 1, 10\}$ and chose the best performing model.](image)

However, as shown in Figure 1, training a zero initialized linear model with gradient descent on these representations shows interesting differences between them. First, optimization is faster when the representation is extracted from a neural network. The second difference is in the error of the models trained on these representation when they are evaluated on a separate test set of 1000 points. These differences—as opposed to expressivity, linear separability, or the ability to fit the data—are the subject of this study.
In the next section we will introduce a measure called representation alignment to evaluate the quality of a representation in terms of the convergence rate and generalization ability that it provides. Neural networks increase alignment in their hidden representation during the training, more so in hidden layers closer to output. We will then discuss how training with gradient descent on a representations with high alignment (such as neural network representations) results in an initial fast phase that reduces most of the loss with small generalization gap.

4 Representation Alignment

Representation alignment is a relationship between a label vector and a representation matrix. Representation alignment facilitates learning the model. In this section, we define and discuss alignment, starting first by providing intuition about the reason for the definition.

To understand alignment, consider first the squared loss, rewritten using the SVD

$$
\| \Phi w - y \|^2 = \| U \Sigma V^T w - U U^T y \|^2 \\
= \| U (\Sigma w^V - y^U) \|^2 \\
= \| \Sigma w^V - y^U \|^2 \\
= \sum_{i=1}^{r(\Phi)} \| \sigma_i w_i^V - y_i^U \|^2 + \sum_{i=r(\Phi)+1}^{n} \| y_i^U \|^2
$$

The vector $y^U$ is the projection of the entire label vector $y$ of $n$ samples onto the basis $U$. In this rewritten form, it is clear that if $\sigma_i$ is small and the rotated $y_i^U = u_i^V y$ is big, then $w_i^V$ has to become very big, which is problematic. The representation $\Phi$ is aligned with the label vector $y$ if $u_i^V y$ is large primarily for the large singular values $\sigma_i$.

Before giving a formal definition, let us visualize these values. In Figure 2a we sampled 10000 points from the first two classes of the MNIST dataset (5000 points from each class) and plotted the singular values of $X$ and the squared dot product between the label vector $y$ and the corresponding left singular values $u_{1:n}$. The dot product is noticeably large for the top few singular vectors, and drops once the singular values become small. This need not always be the case. We created the same plot for shuffled labels in Figure 2b. As the association between the features and the labels in MNIST dataset is lost, the label vector is more or less uniformly aligned with all the singular vectors.

We define alignment relative to a threshold on singular values. We assume $\| x_i \| \leq 1$ for all $i \in \{1, \ldots, n\}$, which can be satisfied by normalization. This constraint ensures singular values are not large simply because feature magnitudes are large, and so facilitates comparing different representations. Given a threshold $\tau \geq 0$, the degree of alignment between the representation $\Phi$ and label vector $y$ is

$$
\text{Alignment}(\Phi, y, \tau) = \sum_{\{i : \sigma_i \geq \tau\}} (u_i^T y)^2
$$

If Alignment($\Phi, y, \tau$) is larger for higher $\tau$, then the projected label vector is concentrated on the largest singular values.

We visualize hypothetical good and bad alignment in Figure 3, if we were to plot alignment versus the threshold. Alignment is higher if the curve stays higher, for longer. The green line depicts a hypothetical curve that has good alignment, and the red bad alignment.

4.1 Emergence of Alignment in Learned Features

We used the raw inputs as the features to give a first example. But, we would like to understand the alignment properties of neural networks. In particular, we can ask if alignment can emerge even in the absence of good alignment between the input matrix and label vector, i.e., when the labels are shuffled. We trained a neural network with three hidden layers as wide as the input layer using Adam [29] for 1000 epochs with batches of size 64 on the shuffled labels and extracted the representation from the final hidden layer.

Figures 2c and 2d show the squared projection of the training label vector on the singular vectors of the hidden representation matrix before and after training. The network
has learned a representation where the label vector is mostly aligned with the top singular vectors. This comparison illustrates emergence of representation alignment in hidden representations even when there is no such relationship between the label and the input features.

One might wonder if this is a property of all representation learning approaches, or particular to neural networks. We have already seen that sparse coding with dictionary learning does not appear to have this property, and in fact when we measure the alignment for Figure 1, we find it is much lower than that of the neural network. But sparse coding is an unsupervised procedure. Perhaps whenever we include a supervised component, alignment improves.

The learned RBF network allows us to assess this hypothesis, since it is supervised but has a different architecture than typical neural networks. In particular, its first layer consists of RBF units, where the training adjusts their centers. We find that alignment for the RBF network is better than sparse coding, but still notably worse than the NN. This one experiment is by no means definitive, and we only include it to show that different methods can have different alignment properties. This work is primarily focused on showing the emergent alignment properties of neural networks, rather than making the much stronger claim that this is a unique property to neural networks.

4.2 An Illustrative Example

We can measure alignment, and see that alignment improves when learning the representation with a neural network. But it may not be obvious when alignment will be low or high. Figure 5 is an example that illustrates how the relationship between the representation and the label vector affects alignment. A set of 1000 points are normalized to unit length so they fall on a circle. 500 points have angles around $\pi/4$ and the angles for the other 500 points are around $-3\pi/4$. This representation matrix has two right singular vectors (principal components) $v_1$ and $v_2$ with singular values $\sigma_1 = 9.58$ and $\sigma_2 = 2.84$. The plot shows two different possible sets of labels, yellow for class one and purple for class two.

The representation is perfectly able to linearly separate the two classes in both scenarios, but the alignment is higher in the leftmost plot. The labeling on the left defines the two classes in the direction of the first principal component, where data is more spread out. As a result, the label vector is mostly in the direction of the first singular vector $w_1$. The labeling on the right defines the classes in the direction of $v_2$ so the classes are closer together. The resulting label vector is mostly in the direction of $v_2$ which has a smaller singular value and so has lower alignment than the previous one.

5 ALIGNMENT AND CONVERGENCE RATES

In this section we discuss how aligned representations help speed learning. We first conceptually explain this phenomenon, by examining the gradients of the loss. We then provide a simple experiment to highlight the connection between alignment and convergence rates.

5.1 Improving Convergence Rates

To understand why alignment helps with convergence rates, let us consider again the decomposition of the squared error, rewritten slightly, and its gradient

$$\|\Phi w - y\|^2 = \sum_{i=1}^{r(\Phi)} \sigma_i^2 \|w_i^V - \frac{y_i}{\sigma_i}\|^2 + \sum_{i=r(\Phi)+1}^n \|y_i^U\|^2,$$

$$\nabla \|\Phi w - y\|^2 = \sum_{i=1}^{r(\Phi)} \sigma_i^2 \nabla \|w_i^V - \frac{y_i}{\sigma_i}\|^2.$$

Each component of the loss will be reduced by movements of the weights in the direction of a right singular vector, seen in the gradient decomposition. Each term in the sum is a one-dimensional optimization process in the direction of a right singular vector and its rate depends on the singular value. We can imagine the weight $w_i^V$ over time as a process, where a larger singular value means larger steps or changes in this process. The processes do not interact with each other since the singular vectors are orthogonal to each other and the movement of the weights in one direction does not depend on its previous movements in other directions. A process with nonzero singular value $\sigma_i$ needs to move the weights to $y_i^U/\sigma_i$ to fully reduce its share of training loss. This required movement is smaller for processes with larger singular values.

From this, we can see that gradient descent will exhibit two phases, as has been previously noted [30]. In an initial fast phase, all the processes are reducing their share of the loss. Since the processes with larger singular values are faster, their loss will be reduced in a few iterations and with a small total movement of the weights. When these processes have essentially stopped, the slower ones with small singular values will keep reducing the loss slowly and growing the weights to large magnitudes. Figure 6 illustrates the fast
network—that need not be overparameterized—is learned to create features for transfer, to a new target task. The aim of this paper is show emergence of alignment in neural network hidden representations and characterize positive and negative transfer in this framework.

5.2 A Small Experiment Connecting the Alignment Curve and Convergence Rates

To see the impact of alignment on convergence rates, we fix the representation and create two synthetic label vectors with different alignment properties for that representation. The representation is obtained by sampling 1000 points from the UCI CT Position dataset. The first label vector, \( y_1 \), is set to the 10th singular vector (approximately magnitude 70). The second label vector, \( y_2 \), is a weighted sum of the 5th and the 50th singular vectors (approximately magnitude 115 and 30). We visualize their alignment curves in Figure 7.

We trained a linear model on each of these two label vectors and plotted the learning curves. These two models in fact show different comparative rates of convergence in early and late training. Initially, the model trained on \( y_2 \) converges faster as it is reducing the loss corresponding to the 5th singular vector (high magnitude). When this loss is mostly reduced, the model keeps slowly reducing the part that corresponds to the small singular value. The model for \( y_1 \) consistently decreases error to zero, since its perfectly aligned with the singular value of magnitude \( 2 \). Learning curves for these two label vectors. The convergence rates are as predicted by the alignment curve.

Fig. 7: (Left) Alignment curves for the two label vectors. After a threshold of 30 the alignment curve drops slightly for \( y_2 \), representing that a small amount of \( y_2 \) is aligned with the smaller singular value, and then drops fully at threshold 115, as most of \( y_2 \) is aligned with the large magnitude singular value. The alignment curve for \( y_1 \) drops at 70, since all of \( y_1 \) is aligned with the singular value of magnitude 70. (Right) Learning curves for these two label vectors. The convergence rates are as predicted by the alignment curve.

Fig. 6: The fast and slow processes in a well aligned representation (left) and a poorly aligned representation (right), both with rank 2. The singular values are \( \sigma_1 = 6 \) for the fast process and \( \sigma_2 = 1 \) for the slow process. The y-axis is the corresponding share of loss and the x-axis is the projection of weights in the corresponding direction. The orange circle shows the state of the process at zero initialization and the green circle shows the state of the process at the end of the initial fast phase, i.e., when the fast process has mostly reduced its loss. With the well aligned representation in (a), the fast process gets a larger portion of overall training loss, and reduces that loss (from orange to green). With low alignment in (b), each process gets half of the overall loss, and so the fast process reduces less of the loss (less change between orange to green). With high alignment, there is little remaining error to reduce in the slow phase. With low alignment, the optimization needs to continue through the slow phase and grow the weights.

and slow process for two representations with high and low alignment.

There is also theory to support improved convergence from this alignment. [25] showed that for an input gram matrix, similar to \( XX^T \), the convergence rate is exactly dictated by alignment. Their analysis was for two layer ReLU networks, so does not match the transfer setting we consider with linear models. However, their proof extends in a straightforward way to linear models, to show that the convergence rate is dictated by

\[
\sum_{i=1}^{r} \eta (1 - \eta \sigma_i^2) \sigma_i^2 k (u_i^T y_i)^2
\]

where \( \eta \) is the stepsize and \( k \) is the number of iterations (steps of descent) so far (see appendix A for the proof). Moreover, with high alignment gradient descent requires a small magnitude of weights to reduce a large portion of the loss. Early stopping after this amount of loss is reduced guarantees small weights, and thus better generalization, without a need for explicitly regularizing the norm of the weights [30].

The role of alignment in optimization and generalization has been previously noted [25], [30], [31], in overparameterized networks for the standard training-testing setting. We leverage these insights for the transfer setting, where a neural
Fig. 8: Training increases alignment with the source task and related tasks and decreases alignment with unrelated tasks. The plots show alignment between the label vectors of the source task, related task, and unrelated task and the original representation (original), the initial hidden representation (init), and the hidden representation after training on the source task (transfer). The results are averaged over 10 runs. Error bars show standard errors. In (a), training on the source task increases the representation alignment for the source task itself. Alignment with the related task’s labels is also increased (b), and the transferred representation is better aligned than the original features. For the unrelated tasks, however, training on the source task has reduced the alignment, and the transferred representation is worse than the original features (c).

Fig. 9: Train (dashed lines) and test (solid lines) errors of linear models after a certain number of steps (a,b) and the magnitude of weights after a certain reduction of training loss (c,d). The results are averaged over 10 runs after a sweep on three different step-size values. On the related task (a) for the model with the transferred representation that has better alignment the initial fast phase of the training, where the weights and the generalization gap is small, can substantially reduce the training loss (see the initial sharp drop of the green curve). This fast phase is almost nonexistent in b since the alignment of the transferred representation is decreased, and from the beginning the model needs to grow its weights and the generalization gap to larger magnitudes to reduce the training loss.

We demonstrate the correlation between alignment, transfer, and task similarity using peaks functions, a framework of related and unrelated tasks for neural networks [32]. A peaks function is of the form \( P_{X,Y,Z} = \begin{cases} 1 & \text{if } X > 0.5 \\ Y & \text{else} \\ Z \end{cases} \) where \( X, Y, Z \) are variables. There are six possible variables \( A, B, C, D, E, F \) defined on \([0, 1)\) and the label of a peaks function depends on three out of these six variables. A total of 120 peaks functions can be defined using different permutations of three out of six variables: \( P_{A,B,C}, P_{A,B,D}, \ldots, P_{F,E,D} \).

Each variable is encoded into 10 features instead of being fed directly to a network as a scalar. The features are obtained by evaluating an RBF function centered at points 0.0, 0.1, \ldots, 0.9, with height 0.5 and standard deviation 0.1. Therefore, there are 60 features representing the 6 variables, and a scalar label that depends on 30 of these features. Inputs are normalized to a length of one and the labels are normalized to have a mean of zero.

The neural network used in the experiments has 60 input units, one hidden layer with 60 neurons and ReLU activation, and one linear output unit. The model is trained by reducing mean squared error using Adam with batch-size 64 and step-size 0.001.

First we study whether training a neural network on the source task results in a representation that is well aligned with the source tasks and similar tasks. We pick a source task \( S \) by randomly choosing 3 variables from \( A-F \) and consider two target tasks \( T_1 \) and \( T_2 \). \( T_1 \) has the same variables as \( S \) but in a different order and \( T_2 \) uses the three other variables. For example a possible setup is: \( S = P_{A,B,C}, T_1 = P_{B,A,C}, T_2 = P_{E,D,F} \). We create 10000 inputs by sampling the variables \( A-F \) randomly from \([0, 1)\) and compute the outputs for the source and target tasks to create a dataset of size 10000 for these tasks. Then we train a neural network only on \( S \) until convergence and compare the alignment between the labels of the source and target tasks with the hidden representation after training and the hidden representation at initialization.

The hypothesis is that training on \( S \) increases the alignment between the hidden representation and the labels of
a task if the task is similar to \( S \) and reduces it if the task is dissimilar. We also include the alignment between the original inputs and the labels. The extracted representations are normalized to length one, to appropriately measure alignment. In Figure 8, we can see that the hypothesis is verified: training on the source task increases the alignment for both the source task and the related task, and decreases alignment with the unrelated task.

Next we want to check if representations with better alignment also have better generalization when training data is scarce. This time we choose 100 out of the 10000 points for the related and unrelated task, and compared the performance of a linear model on the original features, hidden representation at initialization, and hidden representation after training on the \( S \). The test error is evaluated on a new dataset of size 1000 for evaluating the model’s generalization. Figure 9 (a,b) show training and test errors after a certain number of iterations for the related and unrelated task. Figure 9 (c,d) show the norm of weights after a certain reduction of training loss.

The relative performance of the representations mirrors their alignment with the labels. Further, the role of alignment is noticeable in the fast and slow phases of the learning curves. When the target task is similar, the initial fast phase of the transferred representation reduces a large ratio of the overall loss with a small generalization gap and without requiring large weights. Comparison with the representation at initialization shows that this benefit comes from training on the source task. In the unrelated task, where training on the source task reduces alignment, optimization with the transferred features enters the slow phase early in training and performs worse than both the original features and the initial hidden representation. This reduction in alignment provides a clear explanation for this negative transfer.

7 Experiments Measuring Alignment in Pre-trained CNNs

A natural setting to test alignment is in areas where big neural networks are pre-trained for easy re-use. One such setting is computer vision, where end-to-end training with deep Convolutional Neural Networks (CNNs) has replaced training linear models on hand-crafted features. Training a deep CNN from scratch, however, requires a large amount of labeled data which is not available for many tasks. Instead, a common strategy is to use transfer [5], [33]. A CNN is trained on a large corpus of data, presumably related to a wide array of target tasks, and made publicly available. The pre-trained model is then used to create features for the target task—often using the last hidden layer of the pre-trained CNN—and a simpler model trained on these features with the more limited dataset for the target task.

We take a CNN first trained on the large ImageNet dataset [2], and use features given by the last layer to train a linear model on Cifar10 and Cifar100 [34]. The benefit of feature transfer from ImageNet to Cifar10 and Cifar100 is already established in the literature [4], and we ask whether an increase in representation alignment is behind this gain. The CNN models that are studied are VGG16 [35] and the two residual networks ResNet50 and ResNet101 [36].

For each one of target tasks 1000 data points are sampled from two random classes and given \( \pm 1 \) labels. We then plot the alignment for the input features and the hidden representations before and after training on ImageNet. Recall alignment from Equation (1), \( \sum_{i \in \{1,\ldots,n\}} (u_i y_i)^2 \) for a given threshold \( \tau \). Figure 10 shows the results for the three networks on Cifar10 and Cifar100.

Training on the source task (ImageNet) significantly increases alignment with the related tasks (Cifar10 and Cifar100). The alignment stays high and drops off suddenly, indicating that the label vector is largely concentrated on larger singular values. An interesting observation is that the hidden representation at random initialization is extremely poorly aligned. Further, the drop in alignment at initialization

![Fig. 10: Representation alignment for input features (original), hidden representation at initialization (init), and hidden representation after training on ImageNet (transfer). Training on ImageNet increases the alignment for all three models and on both Cifar10 and Cifar100.](image)

![Fig. 11: ResNet features show higher alignment than hand-crafted image features and RBF features. The curves are averaged over 5 runs and the shades show standard errors.](image)
Fig. 12: Train (dashed) and test (solid) learning curves for squared error and logistic loss minimization on different representations. CNN representations show the best performance.

is more extreme in deeper networks. We see this again in the experiments in the next section, and discuss it there.

We then extract 1000 images from two random classes of Cifar10 and Cifar100 and compare the alignment of representation extracted from ResNet101 pre-trained on ImageNet, SIFT [37] features with dictionary size 2048, and HOG features [38] with 9 orientations, 64 pixels per cell, and 4 cells per block. We also include Radial Basis Function (RBF) representation of the data with bandwidth 1.0. The input to Radial Basis Functions is a flattened image normalized to length one, and the centers are set to all the 1000 normalized flattened training data points to ensure expressivity. We see in Figure 11 that CNN features show considerably higher alignment than all three other representations on both datasets.

Finally we train models with both squared error and logistic loss on these representations. Figure 12 shows the learning curves. With both loss functions, higher alignment results in faster optimization and better performance on test data. An interesting observation is the high training error of RBF despite its expressivity. We computed the closed form least squares solution for this representation and found that it achieves zero squared error and full classification accuracy, which suggests that the high training error of gradient descent is due to extremely slow convergence. Note that alignment is not a simple consequence of linear separability, and, as in the case of this RBF representation, a linearly separable representation can have low alignment.

8 Experiments Measuring Alignment in a Variety of Networks

We provide an empirical study of representation alignment with various architectures and optimizers for regression in the UCI CT Position Dataset [27] and for classifying the first two classes of Cifar10 and MNIST. We then investigate the degree of alignment in internal layers of the learned networks.

Training increases alignment in different architectures and optimizers. We test five different settings. In the first, we fix the hidden layer width to 128, the optimizer to Adam and train networks of different depth. In the second, we set the depth to 4, the optimizer to Adam and train networks of different hidden layer width. In the third, we set the depth to 4, the hidden layer width to 128 and train networks with different optimizers. The fourth and fifth settings consider other activations (tanh, PReLU, and LearkyReLU) and batch-sizes (32, 128, and 256). In all of these settings, training increases alignment in the final hidden representation compared to both the input features and the hidden representation at initialization. The increase in alignment occurs on both train and test data. The plots for train and test data are shown in Figures 15 and 16.

Additionally, we can see that for both Cifar10 and CT Position, deeper networks resulted in higher levels of alignment in the final layer. Similarly for these datasets, Adam, as compared with SGD and RMSprop, more significantly improves alignment, especially in Cifar10.

Alignment is higher in layers closer to the output. Conventional wisdom in deep learning is that networks learn better representations in the layers closer to output [33]. Alain et al. [39] trained linear classifiers on the learned
hidden representations and showed that linear separability increases along the depth. Another example is in Computer Vision where the first hidden layers of a CNN usually find generic features like edge detectors, and the last hidden layer provides features that are more specialized and allow the final layer to achieve high performance [34], [40].

We test the hypothesis that training a multi-layer network results in hidden representations that are successively better aligned with the labels. For MNIST and CT Position, we pick 10000 random points and train a neural network with three hidden layers of width 128 using Adam with batch-size 64 until convergence. Figure 13 shows the alignment for all the hidden representations along with the original features before and after training, averaged over 5 runs.

On both datasets, layers closer to the output are monotonically better aligned after training. Comparison with the plots at initialization shows that this pattern is the result of training on the task. In fact, the ordering is completely reversed before training. This mirrors what we observed for the CNN experiments. A possible explanation is that with
Fig. 16: Training increases alignment in hidden representation on test data. The orange line is the input features, the dashed lines are hidden representations before training, and the solid lines are hidden representations after training. The plots are averaged over 5 runs and the error bars show standard errors. The label vector for the regression task was moved to the interval $[-1, 1]$ for easier comparison with classification.

more depth, successively more information in the inputs is discarded. Previous work has observed that at initialization, deeper layers have little mutual information with the output [12]. Such layers are closer to random vectors, in terms of relationship to the label vector, so unlikely to be well aligned.

**Higher layers can specialize to the source task.** We ran the experiment with peaks functions this time with three hidden layers instead of one to let the later layers specialize to the source task. Figure 14 shows layer-by-layer alignment curves of the learned representations on the source task (left) and the related task (right). The monotonous pattern in the main paper persists in the left plot as the representations closer to output adapt to the source task, and there is no such monotonous increase in alignment with another task as seen in the right plot. This observation is consistent with the experiments by [40] on transfer learning with fine-tuning that show layers that are closer to output specialize to the source task, often at the cost of performance on the target task. As the depth of the transferred layer increases, the set of possible target tasks that would be deemed similar to the source task becomes smaller, since the learned representation only keeps the information necessary for the source task.
9 Related Work

Early work by Hansen et al. [41] studied the closed-form solution of linear regression with \( \ell_2 \) regularization or truncated SVD under the Discrete Picard Condition (DPC). DPC states that the magnitude of projections of the label vector on the left singular vectors decays to zero faster than the sequence of singular values. A follow-up paper [42] studied the closed-form solution of \( \ell_2 \)-regularized regression under DPC using L-curves which show the magnitudes of residual and weights at different levels of regularization. The name of these plots comes from their L-shaped appearance which means that a certain amount of residual can be reduced with small weights and reducing the rest of the loss requires large weights. The authors attributed this pattern to the projection of the label vector onto the directions of large and small singular values.

Jacot et al. [31] provided the Neural Tangent Kernel (NTK) framework that approximates the behaviour of over-parametrized neural networks and can be studied theoretically. They also observed faster convergence of neural networks in directions that correspond to the first eigenvectors of the kernel’s Gram matrix. As label randomization tests by Zhang et al. [24] resurrected interest in the relationship between the task and representation, [25] studied the difference between datasets with true and random labels for gradient descent optimization and generalization. They provided an analysis for overparameterized neural networks that requires projecting the label vector on the eigendirections of the Gram matrix and found that true labels are mostly aligned with directions of large singular values. Oymak et al. [30] refined the analysis of Arora et al. [25] by separating the large and small eigenvalues with a threshold and provided better generalization guarantees by early stopping before the slow processes grow the weights to large magnitudes.

Recent work has explored adaptation of the network to a task [30, 43, 44, 45, 46]. They showed that the matrix of a kernel associated with the network evolves such that the labels are mostly aligned with the first eigenvectors of the corresponding matrix. Through a similar functional view, Lampinen et al. [47] studied adaptation of deep linear networks to the task without the kernel approximation and showed the benefit of learning multiple related tasks at the same time.

It is important to distinguish this form of adaptation, which is sometimes called “neural tangent feature alignment” or “neural feature alignment,” and alignment of hidden representations. Neural tangent feature alignment views the whole network as a black-box function and studies adaptation of the model in the space of functions. Our work investigates the intermediate layers one by one and shows that each one increases alignment in its representation. For transfer learning, neural tangent feature alignment implies that if a network is trained on a source task, the whole network can be further trained efficiently on a related target task. Hidden representation alignment means that the features learned in each hidden layer are suitable for training a new model on the related target task.

Finally, Maennel et al. [48] showed how the weights in each layer of neural network adapt their spectral properties to the input features, which explains the benefit of feature transfer on tasks with similar input. This analysis disregards adaptation to labels and cannot explain why transfer to unrelated tasks on the same input can hurt (like the case of negative transfer to dissimilar peaks functions). The authors explain negative transfer through inactivity of ReLU units.

10 Conclusion

In this work, we developed and investigated alignment as a potential reason for why neural network representations transfer well. This alignment emerges from the training process. In contrast, we showed that random neural networks have poor alignment, even if they are sufficiently large to enable accurate approximation. This alignment property significantly improves the learning process for new tasks, given those features, because the loss is dominated by a small set of directions that reduce error quickly during the optimization process. We measured the properties of a variety of neural network representations, showing that it can explain positive transfer for similar tasks and negative transfer for dissimilar tasks; that widely used CNNs have high levels of alignment; and finally that this property appears to emerge across a variety of architectures, with higher alignment in the later layers. An interesting question for future work is the role of alignment in the setting where lower layers are also fine-tuned on the target task. Understanding how the dynamics of gradient descent leads to the emergence of alignment is another direction to pursue.

Acknowledgments

This research was funded by the Canada CIFAR AI Chair program, the Alberta Machine Intelligence Institute (Amii) and NSERC. We would also like to thank Alireza Fallah, Behnam Neyshabur, and members of the Reinforcement Learning and Artificial Intelligence group at the University of Alberta for their helpful comments.

References

[1] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, 2013.
[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” International Journal of Computer Vision, 2015.
[3] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Advances in Neural Information Processing Systems, 2013.
[4] S. Kornblith, J. Shlens, and Q. V. Le, “Do better imagenet models transfer better?” in Conference on Computer Vision and Pattern Recognition, 2019.
[5] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks,” in Conference on Computer Vision and Pattern Recognition, 2014.
[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pretraining of deep bidirectional transformers for language understanding,” arXiv preprint arXiv:1810.04805, 2018.
[7] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in Conference on Empirical Methods in Natural Language Processing, 2014.
[8] D. S. Touretzky and D. A. Pomereau, “What’s hidden in the hidden layers?” BYTE, vol. 14, no. 8, 1989.
**APPENDIX A**

**Convergence Rate**

For completeness, we include the convergence rate result here. For representation matrix $\Phi$ and label vector $y$, the optimal linear regression solution is $w^* = (\Phi^T \Phi)^{-1} \Phi^T y$ where for this result we assume $\Phi$ is full rank. Let $\sigma_{\text{max}}$ be the maximum singular value for $\Phi$.

**Proposition 1.** Given representation matrix $\Phi$ and label vector $y$, the batch gradient descent updates $w_{t+1} = w_t - \eta \Phi^T (\Phi w_t - y)$ with stepsize $0 < \eta < \sigma_{\text{max}}^{-2}$ and $w_0 = 0$ results in

$$
\|w_t - w^*\| \leq \sqrt{\sum_{j=1}^{r(\Phi)} (1 - \eta \sigma_j^2)^{2t-1} (u_j^T y)^2}
$$

$$
\|\Phi w_t - \Phi w^*\| \leq \sqrt{\sum_{j=1}^{r(\Phi)} (1 - \eta \sigma_j^2)^{2t} (u_j^T y)^2}
$$

**Proof.** For $\Phi = U \Sigma V^T$ the thin SVD of $\Phi$ with $\Sigma \in \mathbb{R}^{n \times d}$, let $\Lambda = \Sigma^T \Sigma$, $\Lambda = \Phi^T \Phi = V A^T V^T$ and $b = \Phi^T y = V \Sigma^T U^T y$. The gradient descent update corresponds to the following iterative linear system update

$$
w_{t+1} = w_t - \eta \Phi^T (\Phi w_t - y) = w_t - \eta (A w_t - b)
$$

with $b = \eta (I - \eta A) y$.

Starting from $w_0$, we get that

$$w_1 = \eta b = \eta (I - \eta A) y b
$$

$$w_2 = (I - \eta A) (\eta b) + \eta b = \eta [(I - \eta A)^2 + (I - \eta A)] b
$$

$$\ldots
$$

$$w_{t+1} = \eta \sum_{i=0}^{t} (I - \eta A)^i b
$$

Further,

$$
(I - \eta A)^t = V (I - \eta \Lambda)^t V^T
$$

$$
\sum_{i=0}^{t} (I - \eta A)^i = V \sum_{i=0}^{t} (I - \eta \Lambda)^i V^T
$$

If we let $\tilde{\Lambda}_t = \sum_{i=0}^{t} (I - \eta \Lambda)^i$ and $\lambda_j$ the $j$ entry on the diagonal in $\Lambda$, then because $\lambda_j > 0$ for all $j$, each component of this diagonal matrix corresponds to

$$
\tilde{\Lambda}_{t,j} = \sum_{i=0}^{t} (I - \eta \lambda_j)^i = \frac{1 - (1 - \eta \lambda_j)^{t+1}}{1 - (1 - \eta \lambda_j)} = \frac{1 - (1 - \eta \lambda_j)^{t+1}}{\eta \lambda_j}
$$

Additionally,

$$
(I - \eta A)^t b = V (I - \eta \Lambda)^t V^T V \Sigma^T U^T y
$$

$$
= V (I - \eta \Lambda)^t \Sigma^T U^T y
$$

For $\hat{\Lambda}_{t,j} = (I - \eta \lambda_j)^{-1} = \frac{1}{\eta \lambda_j}$, where the inverse exists due to conditions on $\eta$, we have $w^* = \eta V \hat{\Lambda}_s \Sigma^T U^T y$. Finally we get

$$
\|w_t - w^*\| = \eta \|V \hat{\Lambda}_t \Sigma^T U^T y - V \hat{\Lambda}_s \Sigma^T U^T y\|
$$

$$
= \eta \|V (\hat{\Lambda}_t - \hat{\Lambda}_s) \Sigma^T U^T y\|
$$

$$
= \eta \|(\hat{\Lambda}_t - \hat{\Lambda}_s) \Sigma^T U^T y\|
$$

Notice that $\tilde{\Lambda}_{s,j} - \tilde{\Lambda}_{t,j} = \frac{(1 - \eta \lambda_j)^{t+1}}{\eta \lambda_j}$ and each component of $\eta (\Lambda_s - \hat{\Lambda}_t) \Sigma^T$ equals $\frac{(1 - \eta \lambda_j)^{t+1}}{\sigma_j}$. We therefore get

$$
\|w_t - w^*\|^2 = \|\eta (\Lambda_s - \hat{\Lambda}_s) \Sigma^T U^T y\|^2
$$

$$
= \sum_{j=1}^{r(\Phi)} \frac{(1 - \eta \lambda_j)^{t+1}}{\sigma_j} u_j^T y
$$

$$
= \sum_{j=1}^{r(\Phi)} (1 - \eta \sigma_j^2)^{2(t+1)} \frac{(u_j^T y)^2}{\sigma_j}
$$

which shows the first part of the proposition. The alignment relationship is clear in this result. The term $\frac{(u_j^T y)^2}{\sigma_j}$ is big if $(u_j^T y)^2$ is big and $\sigma_j$ is small. This is further exacerbated by the fact that $(1 - \eta \sigma_j^2)^{2(t+1)}$ shrinks slowly if $\sigma_j$ is small. On the other hand, consider the case that $(u_j^T y)^2$ is not notably smaller than $\sigma_j$, even if $\sigma_j$ is small. Then this term contributes only a small amount to the loss in the beginning, i.e., $\frac{(u_j^T y)^2}{\sigma_j}$, and this small amount is then reduced with each iteration, slowly if $\sigma_j$ is small and quickly if $\sigma_j$ is big. The ideal rate is obtained when high magnitude $(u_j^T y)^2$ is concentrated on large $\sigma_j$. Now we show the rate of convergence of the predictions:

$$
\|\Phi w_t - \Phi w^*\|^2 = \|\Phi (w_t - w^*)\|^2
$$

$$
= \|\eta U \Sigma V^T V (\Lambda_s - \hat{\Lambda}_s) \Sigma^T U^T y\|^2
$$

$$
= \|\eta U \Sigma (\Lambda_s - \hat{\Lambda}_s) \Sigma^T U^T y\|^2
$$

$$
= \|\eta \Sigma (\Lambda_s - \hat{\Lambda}_s) \Sigma^T U^T y\|^2
$$

$$
= \sum_{j=1}^{r(\Phi)} \frac{(1 - \eta \lambda_j)^{t+1} u_j^T y}{\sigma_j}
$$

$$
= \sum_{j=1}^{r(\Phi)} (1 - \eta \sigma_j^2)^{2(t+1)} \frac{(u_j^T y)^2}{\sigma_j}
$$

The term $(1 - \eta \sigma_j^2)^{2(t+1)}$ appears again in this result and shows the improved convergence rate when $(u_j^T y)^2$ is concentrated on directions of large singular values. \qed