A Closer Look at How Fine-tuning Changes BERT

Yichu Zhou
School of Computing
University of Utah
flyaway@cs.utah.edu

Vivek Srikumar
School of Computing
University of Utah
svivek@cs.utah.edu

Abstract

Given the prevalence of pre-trained contextualized representations in today's NLP, there have been many efforts to understand what information they contain, and why they seem to be universally successful. The most common approach to use these representations involves fine-tuning them for an end task. Yet, how fine-tuning changes the underlying embedding space is less studied. In this work, we study the English BERT family and use two probing techniques to analyze how fine-tuning changes the space. We hypothesize that fine-tuning affects classification performance by increasing the distances between examples associated with different labels. We confirm this hypothesis with carefully designed experiments on five different NLP tasks. Via these experiments, we also discover an exception to the prevailing wisdom that "fine-tuning always improves performance". Finally, by comparing the representations before and after fine-tuning, we discover that fine-tuning does not introduce arbitrary changes to representations; instead, it adjusts the representations to downstream tasks while largely preserving the original spatial structure of the data points.

1 Introduction

Pre-trained transformer-based language models (e.g., Devlin et al., 2019) form the basis of state-of-the-art results across NLP. The relative opacity of these models has prompted the development of many probes to investigate linguistic regularities captured in them (e.g., Kovaleva et al., 2019; Conneau et al., 2018; Jawahar et al., 2019).

Broadly speaking, there are two ways to use a pre-trained representation (Peters et al., 2019): as a fixed feature extractor (where the pre-trained weights are frozen), or by fine-tuning it for a task. The probing literature has largely focused on the former (e.g., Kassner and Schütze, 2020; Perone et al., 2018; Yaghoobzadeh et al., 2019; Krasnowska-Kieraś and Wróblewska, 2019; Wallace et al., 2019; Pruksachatkun et al., 2020; Aghajanyan et al., 2021). Some previous work (Merchant et al., 2020; Mosbach et al., 2020b; Hao et al., 2020) does provide insights about fine-tuning: fine-tuning changes higher layers more than lower ones and linguistic information is not lost during fine-tuning. However, relatively less is understood about how the representation changes *during* the process of fine-tuning and why fine-tuning invariably seems to improve task performance.

In this work, we investigate the process of fine-tuning of representations using the English BERT family (Devlin et al., 2019). Specifically, we ask:

1. Does fine-tuning always improve performance?
2. How does fine-tuning alter the representation to adjust for downstream tasks?
3. How does fine-tuning change the geometric structure of different layers?

We apply two probing techniques—classifier-based probing (Kim et al., 2019; Tenney et al., 2019) and DIRECTPROBE (Zhou and Srikumar, 2021)—on variants of BERT representations that are fine-tuned on five tasks: part-of-speech tagging, dependency head prediction, preposition supersense role & function prediction and text classification. Beyond confirming previous findings about fine-tuning, our analysis reveals several new findings, briefly described below.

First, we find that *fine-tuning introduces a divergence between training and test sets*, which is not severe enough to hurt generalization in most cases. However, we do find one exception where fine-tuning hurts the performance; this setting also has the largest divergence between training and test set after fine-tuning (§4.1).

Second, we examine how fine-tuning changes labeled regions of the representation space. For a representation where task labels are not linearly separable, we find that *fine-tuning adjusts it by* *grouping points with the same label into a small number of clusters (ideally one), thus simplifying the underlying representation.* Doing so makes it easier to linearly separate labels with fine-tuned representations than untuned ones (§4.2). For a representation whose task labels are already linearly separable, we find that *fine-tuning pushes the clusters of points representing different labels away from each other, thus introducing large separating regions between labels.* Rather than simply scaling the points, clusters move in different directions and with different extents (measured by Euclidean distance). Overall, these clusters become distant compared to the untuned representation. We conjecture that the enlarged region between groups admits a bigger set of classifiers that can separate them, leading to better generalization (§4.3).

We verify our distance hypothesis by investigating the effect of fine-tuning across tasks. We observe that fine-tuning for related tasks can also provide useful signal for the target task by altering the distances between clusters representing different labels (§4.4).

Finally, *fine-tuning does not change the higher layers arbitrarily.* This confirms previous findings. Additionally, we find that fine-tuning largely preserves the relative positions of the label clusters, while reconfiguring the space to adjust for downstream tasks (§4.5). Informally, we can say that fine-tuning only "slightly" changes higher layers.

These findings help us understand fine-tuning better, and justify why fine-tuned representations can lead to improvements across many NLP tasks[1].

2 Preliminaries: Probing Methods

In this work, we probe representations in the BERT family during and after fine-tuning. First, let us look at the two supervised probes we will employ: a classifier-based probe (e.g., Tenney et al., 2019; Jullien et al., 2022) to assess how well a representation supports classifiers for a task, and DIRECTPROBE (Zhou and Srikumar, 2021) to analyze the geometry of the representation.

2.1 Classifiers as Probes

Trained classifiers are the most commonly used probes in the literature (e.g. Hewitt et al., 2021; Whitney et al., 2021; Belinkov, 2021). To understand how well a representation encodes the labels for a task, a probing classifier is trained over it, with the embeddings themselves kept frozen when the classifier is trained.

For all our experiments, we use two-layer neural networks as our probe classifiers. We use grid-search to choose the best hyperparameters. Each best classifier is trained five times with different initializations. We report the average accuracy and its standard deviation for each classifier.

The hidden layer sizes are selected from $\{32, 64, 128, 256\} \times \{32, 64, 128, 256\}$, and the regularizer weight from the range 10^{-7} to 10^{0}. All models use ReLUs as the activation function for the hidden layer and are optimized by Adam (Kingma and Ba, 2015). We set the maximum number of learning iterations to 1000. We use `scikit-learn` v0.22 (Pedregosa et al., 2011) for these experiments.

Classifier probes aim to measure how well a contextualized representation captures a linguistic property. The classification performance can help us assess the effect of fine-tuning.

2.2 DIRECTPROBE: Probing the Geometric Structure

Classifier probes treat the representation as a black box and only focus on the final task performance; they do not reveal how fine-tuning changes the underlying geometry of the space. To this end, we use DIRECTPROBE (Zhou and Srikumar, 2021)[2], a recently proposed technique which analyzes embeddings from a geometric perspective. We briefly summarize the technique and refer the reader to the original work for details.

For a given labeling task, DIRECTPROBE returns a set of clusters such that each cluster only contains the points with the same label, and there are no overlaps between the convex hulls of these clusters. Any decision boundary must cross the regions between the clusters that have different labels (see in Figure 1). Since fine-tuning a contextualized representation creates different representations for different tasks, it is reasonable to probe the representation based on a given task. These clusters allow us to measure three properties of interest.

Number of Clusters: The number of clusters indicates the linearity of the representation for a task. If the number of clusters equals the number of labels, then examples with the same label are grouped into

[1]The code and data to replicate our analysis is available at `https://github.com/utahnlp/BERT-fine-tuning-analysis`

[2]We use the DIRECTPROBE implementation from `https://github.com/utahnlp/DirectProbe` with default settings.

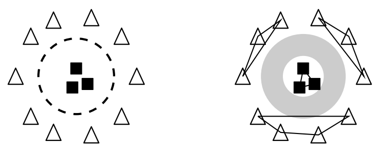

Figure 1: Using the clustering to approximate the set of all decision boundaries. The left subfigure is a simple binary classification problem with a dashed circular decision boundary. The right subfigure is the result of DIRECTPROBE where the gray area is the region that a separator must cross. The connected points represent the clusters that DIRECTPROBE produces.

	Layers	#heads	Dim	#Param
BERT$_{\text{tiny}}$	2	2	128	4.4M
BERT$_{\text{mini}}$	4	4	256	11.3M
BERT$_{\text{small}}$	4	8	512	29.1M
BERT$_{\text{medium}}$	8	8	512	41.7M
BERT$_{\text{base}}$	12	12	768	110.1M

Table 1: Statistics of five different BERT models.

one cluster; a simple linear multi-class classifier will suffice. If, however, there are more clusters than labels, then at least two clusters of examples with the same label can not be grouped together (as in Figure 1, right). This scenario calls for a non-linear classifier.

Distances between Clusters: Distances[3] between clusters can reveal the internal structure of a representation. By tracking these distances during fine-tuning, we can study how the representation changes. To compute these distances, we use the fact that each cluster represents a convex object. This allows us to use max-margin separators to compute distances. We train a linear SVM (Chang and Lin, 2011) to find the maximum margin separator and compute its margin. The distance between the two clusters is twice the margin.

Spatial Similarity: Distances between clusters can also reveal the spatial similarity of two representations. Intuitively, if two representations have similar relative distances between clusters, the representations themselves are similar to each other for the task at hand.

We use these distances to construct a *distance vector* $\mathbf{v}$ for a representation, where each element $\mathbf{v}_i$ is the distance between the clusters of a pair of labels. With n labels in a task, the size of $\mathbf{v}$ is $\frac{n(n-1)}{2}$. This construction works only when the number of clusters equals the number of labels (i.e., the dataset is linearly separable under the representation). Surprisingly, we find this to be the case for most representations we studied. As a measure of the similarity of two representations for a labeling task, we compute the Pearson correlation coefficient between their distance vectors. Note that this coefficient can also be used to measure the similarity between two labeled datasets with respect to the same representation. We exploit this observation to analyze the divergence between training and test sets for fine-tuned representations (§4.1).

[3]We use Euclidean distance throughout this work.

3 Experimental Setup

In this section, we describe the representations and tasks we will encounter in our experiments.

3.1 Representations

We investigate several models from the BERT family (Devlin et al., 2019; Turc et al., 2019). These models all share the same basic architecture but with different capacities, i.e., different layers and hidden sizes. Table 1 summarizes the models we investigate in this work[4]. All of these models are for English text and uncased.

For tokens that are broken into subwords by the tokenizer, we average the subword embeddings for the token representation. We use the models provided by HuggingFace v4.2.1 (Wolf et al., 2020), and Pytorch v1.6.0 (Paszke et al., 2019) for our experiments.

3.2 Tasks

We instantiate our analysis of the BERT models on a diverse set of five NLP tasks, which covers syntactic and semantic predictions. Here, we briefly describe the tasks, and refer the reader to the original sources of the data for further details.[5]

Part-of-speech tagging (POS) predicts the part-of-speech tag for each word in a sentence. The task helps us understand if a representation captures coarse grained syntactic categorization. We use the English portion of the parallel universal dependencies treebank (ud-pud, Nivre et al., 2016).

Dependency relation (DEP) predicts the syntactic dependency relation between two tokens, i.e.

[4]We ignore the BERT$_{\text{large}}$ because, during preliminary experiments, we found BERT$_{\text{large}}$ is highly unstable. The variance between different fine-tuning runs is so large that not comparable with other BERT models. This is consistent with the observations from Mosbach et al. (2020a).

[5]All the datasets we use in this work are publicly available under a creative commons or an open source license.

(w_{head} and w_{mod}). This task can help us understand if, and how well, a representation can characterize syntactic relationships between words. This task involves assigning a category to a *pair* of tokens. We concatenate their contextualized representations from BERT and treat the concatenation as the representation of the pair. We use the same dataset as the POS task for dependencies.

Preposition supersense disambiguation involves two categorization tasks of predicting *preposition's semantic role (**PS-role**)* and *semantic function (**PS-fxn**)*. These tasks are designed for disambiguating semantic meanings of prepositions. Following the previous work (Liu et al., 2019), we only train and evaluate on single-token prepositions from Streusle v4.2 corpus (Schneider et al., 2018).

Text classification, in general, is the task of categorizing sentences or documents. We use the **TREC-50** dataset (Li and Roth, 2002) with 50 semantic labels for sentences. As is the standard practice, we use the representation of the `[CLS]` token as the sentence representation. This task can show how well a representation characterizes a sentence.

3.3 Fine-tuning Setup

We fine-tune the models in §3.1 on the five tasks from §3.2 separately.[6] The fine-tuned models (along with the original models) are then used to generate contextualized representations. The probing techniques described in §2 are applied to study both original and fine-tuned representations.

Our preliminary experiments showed that the commonly used 3-5 epochs of fine-tuning are insufficient for the smaller representations, such as $\text{BERT}_{\text{tiny}}$, and they require more epochs. We fine-tuned all the representations for 10 epochs except $\text{BERT}_{\text{base}}$, which we fine-tuned for the usual three epochs. Note that the fine-tuning phase is separate from the classifier training phase for probing; for the probe classifiers, we train two-layer neural networks (described in §2.1) from scratch on both original and fine-tuned representations[7], ensuring a fair comparsion between them.

4 Observations and Analysis

In this section, we will use classifier probes to examine if fine-tuning always improves classifier performance (§4.1). Then we propose a geometric explanation for *why* fine-tuning improves classification performance using DIRECTPROBE (§4.2 and §4.3). Next, we will confirm this geometric explanation by investigating cross-task fine-tuning (§4.4). Finally, we will analyze how fine-tuning changes the geometry of different layers of $\text{BERT}_{\text{base}}$ (§4.5).

4.1 Fine-tuned Performance

It is commonly accepted that the fine-tuning improves task performance. Does this always hold? Table 2 summarizes the relevant observations from our experiments. Appendix C presents the complete fine-tuning results.

Fine-tuning diverges the training and test set. In Table 2, the last column shows the spatial similarity between the training and test set for each representation. We apply DIRECTPROBE on the training and test set separately. The spatial similarity is calculated as the Pearson correlation coefficient between the distance vectors of training and test set (described in §2). We observe that after fine-tuning, all the similarities decrease, implying that the training and test set diverge as a result of fine-tuning. In most cases, this divergence is not severe enough to decrease the performance.

There are exceptions, where fine-tuning hurts performance. An interesting observation in Table 2 is that $\text{BERT}_{\text{small}}$ does not show the improvements on the PS-fxn task after fine-tuning, which breaks the well-accepted impression that fine-tuning always improve the performance. However, only one such exception is observed across all our experiments (see Appendix C). It is insufficient to draw any concrete conclusions about why this is happening. We do observe that $\text{BERT}_{\text{small}}$ shows the smallest similarity (0.44) between the training and test set after fine-tuning on PS-fxn task. We conjecture that controlling the divergence between the training and test sets can help ensure that fine-tuning helps. Verifying or refuting this conjecture requires further study.

4.2 Linearity of Representations

Next, let us examine the geometry of the representations before and after fine-tuning using DIRECTPROBE and counting the number of clusters. We will focus on the overwhelming majority of cases where fine-tuning does improve performance.

[6]More detailed settings can be found in Appendix A

[7]When the fine-tuned representations are probed, their weights are frozen. Essentially, after fine-tuning, we treat the fine-tuned representations as a black-box that produces embeddings for analysis.

Task		Acc	Sim
POS	original	94.25	0.96
	tuned	94.43	0.72
DEP	original	92.93	0.93
	tuned	94.48	0.78
PS-fxn	original	86.26	0.82
	tuned	*85.08*	**0.44**
PS-role	original	74.22	0.84
	tuned	74.57	0.54
TREC-50	original	81.32	-
	tuned	89.60	-

Table 2: Fine-tuned performances of $\text{BERT}_{\text{small}}$ based on the last layers. The last column shows the spatial similarity (described in §2) between the training and test set. A complete table of all representations and tasks can be found in Appendix C.

Smaller representations require more complex classifiers. Table 3 summarizes the results. For brevity, we only present the results on $\text{BERT}_{\text{tiny}}$. The full results are in Appendix C. We observe that before fine-tuning, small representations (i.e., $\text{BERT}_{\text{tiny}}$) are non-linear for most tasks. Although a non-linearity does not imply poor generalization, it represents a more complex spatial structure, and requires a more complex classifier. This suggests that to use small representations (say, due to limited resources), it would be advisable to use a non-linear classifier rather than a simple linear one.

Fine-tuning makes the space simpler. In Table 3, we observe that the number of clusters decreases after fine-tuning. This tells us that after fine-tuning, the points associated with different labels are in a simpler spatial configuration. The same trend holds for TREC-50 (Table 4), even when the final representation is *not* linearly separable.

Task		#clusters	is linear	Acc
POS	original	30	N	90.76
	tuned	18	N	91.67
DEP	original	50	N	86.74
	tuned	46	Y	89.04
PS-fxn	original	42	N	74.14
	tuned	40	Y	74.40
PS-role	original	46	Y	58.38
	tuned	46	Y	60.31
TREC-50	original	58	N	68.12
	tuned	51	N	84.04

Table 3: The linearity of the last layer of $\text{BERT}_{\text{tiny}}$ for each task. Other results are in Appendix C.

Rep		#clusters	is linear	Acc
$\text{BERT}_{\text{tiny}}$	original	58	N	68.12
	tuned	51	N	84.04
$\text{BERT}_{\text{mini}}$	original	52	N	74.12
	tuned	52	N	88.36
$\text{BERT}_{\text{small}}$	original	52	N	81.32
	tuned	51	N	89.60
$\text{BERT}_{\text{medium}}$	original	52	N	80.68
	tuned	52	N	89.80
$\text{BERT}_{\text{base}}$	original	52	N	85.24
	tuned	51	N	90.36

Table 4: The linearity of the last layer of all models on TREC-50 task. The number of clusters is always more than the number of labels (50).

4.3 Spatial Structure of Labels

To better understand the changes in spatial structure, we apply DIRECTPROBE to *every* intermediate representation encountered during fine-tuning. Here, we focus on the $\text{BERT}_{\text{base}}$. Since all representations we considered are linearly separable[8], the number of clusters equals the number of labels. As a result, each cluster exclusively corresponds to one label. Going ahead, we will use clusters and labels interchangeably.

Fine-tuning pushes each label far away from each other. This confirms the observation of Zhou and Srikumar (2021), who pointed out that the fine-tuning pushes each label away from each other. However, they use the global minimum distance between clusters to support this argument, which only partially supports the claim: the distances between some clusters might increase despite the global minimum distance decreasing.

We track the minimum distance of each label to all other labels during fine-tuning. We find that all the minimum distances are increasing. Figure 2 shows how these distances change in the last layer of $\text{BERT}_{\text{base}}$ for the PS-role and POS tagging tasks. Appendix D includes the plots for all tasks. For clarity, we only show the three labels where the distance increases the most, and the three where it increases the least. We also observe that although the trend is increasing, the minimum distance associated with a label may decrease during the course of fine-tuning, e.g., the label STUFF in PS-role task, suggesting a potential instability of fine-tuning.

[8]In this part, we exclude the TREC-50 task because it is non-linear even after fine-tuning. It is difficult to track the minimum distances between clusters when the clusters are

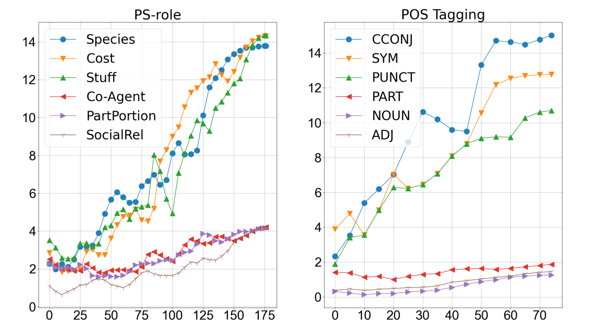

Figure 2: The dynamics of the minimum distances of the three labels where the distance increases the most, and the three where it increases the least. The horizontal axis is the number of fine-tuning updates; the vertical axis is chosen label's minimum distance to other labels. These results come from the last layer of BERT$_{\text{base}}$. A full plots of four tasks can be found in Appendix D.

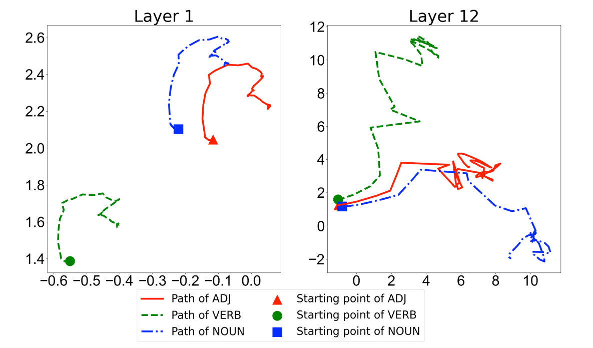

Figure 3: The PCA projection of three closest labels in POS tagging task based on the first (left) and last (right) layer of BERT$_{\text{base}}$. These lines show the paths of the centroids of each label cluster during the fine-tuning. The markers indicate the starting points. This figure is best seen in color.

To further see how labels move during the fine-tuning, we track the centroids of each cluster. We select three closest labels from the POS tagging task and track the paths of the centroids of each label cluster in the last layer of BERT$_{\text{base}}$ during the fine-tuning. Figure 3 (right) shows the 2D PCA projection of these paths. We observe that before fine-tuning, the centroids of all these three labels are close to each other. As fine-tuning proceeds, the centroids move around in different directions, away from each other.

We conclude that fine-tuning enlarges the gaps between label clusters and admits more classifiers consistent with the labels, allowing for better generalization. Note that neither the loss nor the optimizer *explicitly* mandates this change. Indeed, since the labels were originally linearly separable, the learner need not adjust the representation at all.

merging during fine-tuning.

4.4 Cross-task Fine-tuning

In §4.3, we hypothesized that fine-tuning improves the performance because it enlarges the gaps between label clusters. A natural inference of this hypothesis is that the process may shrink the gaps between labels of an unrelated task, and its performance can decrease. In this subsection, we investigate how fine-tuning for one task affects another.

We fine-tune the BERT$_{\text{base}}$ on PS-role and POS tagging tasks separately and use the fine-tuned models to generate contextualized representations for the PS-fxn task. Our choice of tasks in this experimental design is motivated by the observation that PS-role and PS-fxn are similar tasks that seek to predict supersense tags for prepositions. On the other hand, POS tagging can adversely affect the PS-fxn task because POS tagging requires all the prepositions to be grouped together (label ADP) while PS-fxn requires different prepositions to be far away from each other. We apply DIRECTPROBE on both representations to analyze the geometric changes[9] with respect to PS-fxn.

The effects of cross-task fine-tuning depends on how close two tasks are. The third and fourth columns of Table 5 indicate the number of labels whose minimum distance is increased or decreased after fine-tuning. The second column from the right shows the average distance change over all labels, e.g. fine-tuning on POS results in the minimum distances of the PS-fxn labels decreasing by 1.68 on average. We observe that fine-tuning on the same dataset (PS-fxn) increases the distances between labels (second row), which is consistent with observations from §4.3; fine-tuning on a similar task also increases the distances between clusters (third row) but to a lesser extent. However, fine-tuning on a "opposing" task decreases the distances between clusters (last row). These observations suggest that cross-task fine-tuning could add or remove information from the representation, depending on how close the source and target task are.

Small distances between label clusters indicate a poor performance. Based on our conclusion in §4.3 that a larger gap between labels leads to better generalization, we expect that the performance

[9]The PS-fxn task is still linearly separable even after fine-tuning on PS-role or POS tagging tasks.

fine-tuning	probing	#inc	#dec	average inc	Acc
-	PS-fxn	-	-	-	87.75
PS-fxn	PS-fxn	40	0	5.29	89.58
PS-role	PS-fxn	27	13	1.02	88.53
POS	PS-fxn	0	40	-1.68	83.24

Table 5: Classification performances for PS-fxn task using the last layer of $\text{BERT}_{\text{base}}$ when fine-tuning on different tasks. First row indicates the untuned version. The third and forth column indicate the number of labels whose minimum distance is increased or decreased after fine-tuning. The second last column (average inc) shows the average change of the minimum distance over all the labels. The last column indicates the probing accuracy.

of PS-fxn after fine-tuning on PS-role would be higher than the performance after fine-tuning on POS tagging. To verify this, we train two-layer neural networks on PS-fxn task using the representations that are fine-tuned on PS-role and POS tagging tasks. Importantly, we do not further fine-tune the representations for PS-fxn. The last column of Table 5 shows the results. Fine-tuning on PS-fxn enlarges gaps between *all* PS-fxn labels, which justifies the highest performance; fine-tuning on PS-role enlarges gaps between *some* labels in PS-fxn, leading to a slight improvement; fine-tuning on POS tags shrinks the gaps between *all* labels in PS-fxn, leading to a decrease in performance.

In summary, based on the results of §4.2, §4.3 and §4.4, we conclude that fine-tuning injects or removes task-related information from representations by adjusting the distances between label clusters *even if* the original representation is linearly separable (i.e., when there is no need to change the representation). When the original representation does not support a linear classifier, fine-tuning tries to group points with the same label into a small number of clusters, ideally one cluster.

4.5 Layer Behavior

Previous work (Merchant et al., 2020; Mosbach et al., 2020b) showed that during fine-tuning, lower layers changed little compared to higher layers. In the following experiments, we confirm their findings and further show that: (i) fine-tuning does not change the representation arbitrarily, even for higher layers; (ii) an analysis of the changes of different layers by a visual comparison between lower and higher layers. Here, we focus on the POS tagging task with $\text{BERT}_{\text{base}}$. Our conclusions extend to other tasks, whose results are in Appendix E.

Higher layers do not change arbitrarily. Although previous work (Mosbach et al., 2020b) shows that higher layers change more than the lower layers, we find that higher layers still remain close to the original representations. To study the dynamics of fine-tuning, we compare each layer during fine-tuning to its corresponding original pre-trained one. The spatial similarity between two representations is calculated as the Pearson correlation coefficient of their distance vectors as described in §2. Intuitively, a classifier learns a decision boundary that traverses the region between clusters, which makes the distances between clusters more relevant to our analysis (as opposed to the spatial structure of points within each cluster).

Figure 4 shows the results for all four tasks.[10] To avoid visual clutter, we only show the plots for every alternate layer. For the higher layers, we find that the Pearson correlation coefficient between the original representation and the fine-tuned one is surprisingly high (more than 0.5), reinforcing the notion that fine-tuning does not change the representation arbitrarily. Instead, it attempts to preserve the relative positions the labels. This means the fine-tuning process encodes task-specific information, yet it largely preserves the pre-trained information encoded in the representation.

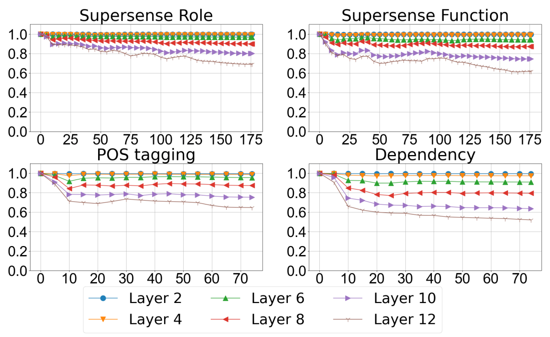

Figure 4: Dynamics of spatial similarity during the fine-tuning process based on $\text{BERT}_{\text{base}}$. The horizontal axis is the number of updates during fine-tuning. The vertical axis is the Pearson correlation coefficient between current space and its original version (before fine-tuning).

The labels of lower layers move only in a small region and almost in the same directions. The unchanged nature of lower layers raises the question: do they not change at all? To answer this question, for every label, we compute difference between its centroids before and after fine-tuning.

[10]We exclude the TREC-50 task because it is non-linear. We cannot have the distance vectors for non-linear representations.

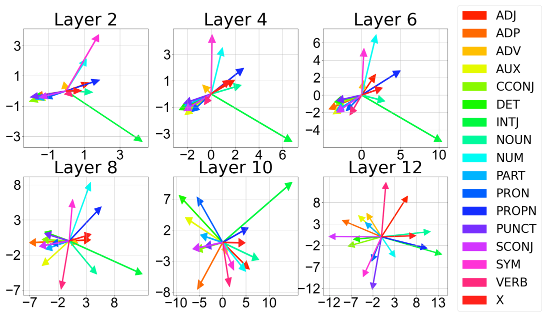

Figure 5: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on POS tagging task and BERT$_{\text{base}}$. Lower layers have a much smaller projection range than the higher layers. This figure is best seen in color.

Figure 5 shows the PCA projection in 2D of these difference vectors. For brevity, we only present the plots for every alternative layer. A plot with all layers can be found in Appendix E. We observe that the movements of labels in lower layers concentrate in a few directions compared to the higher layers, suggesting the labels in lower layers do change, but do not separate the labels as much as higher layers. Also, we observe that the labels INTJ and SYM have distinctive directions in the lower layers.

Note that, in Figure 5, the motion range of lower layers is much smaller than the higher layers. The projected two dimensions range from -1 to 3 and from -3 to 3 for layer two, while for layer 12 they range from -12 to 13 and -12 to 8, suggesting that labels in lower layers only move in a small region compared to higher layers. Figure 3 shows an example of this difference. Compared to the layer 12 (right) paths, we see that the layer 1 paths (left) traverse almost the same trajectories, which is consistent with the observations from Figure 5.

5 Discussion

Does fine-tuning always improve performance? Indeed, fine-tuning almost always improves task performance. However, rare cases exist where fine-tuning decreases the performance. Fine-tuning introduces a divergence between the training set and unseen examples (§4.1). However, it is unclear how this divergence affects the generalization ability of representations, e.g. does this divergence suggest a new kind of overfitting that is driven by representations rather than classifiers?

How does fine-tuning alter the representation to adjust for downstream tasks? Fine-tuning alters the representation by grouping points with the same label into small number of clusters (§4.2) and pushing each label cluster away from the others (§4.3). We hypothesize that the distances between label clusters correlate with the classification performance and confirm this hypothesis by investigating cross-task fine-tuning (§4.4). Our findings are surprising because fine-tuning for a classification task does not need to alter the geometry of a representation if the data is already linearly separable in the original representation. What we observe reveals geometric properties that characterize good representations. We do not show theoretical analysis to connect our geometric findings to representation learnability, but the findings in this work may serve as a starting point for a learning theory for representations.

How does fine-tuning change the underlying geometric structure of different layers? It is established that higher layers change more than the lower ones. In this work, we analyze this behavior more closely. We discover that higher layers do not change arbitrarily; instead, they remain similar to the untuned version. Informally, we can say that fine-tuning only "slightly" changes even the higher layers (§4.5). Nevertheless, our analysis does not reveal why higher layers change more than the lower layers. A deeper analysis of model parameters during fine-tuning is needed to understand the difference between lower and higher layers.

Limitations of this work. Our experiments use the BERT family of models for English tasks. Given the architectural similarity of transformer language models, we may be able to extrapolate the results to other models, but further work is needed to confirm our findings to other languages or model architectures. In our analysis, we ignore the structure within each cluster, which is another information source for studying the representation. We plan to investigate these aspects in future work. We make our code available for replication and extension by the community.

6 Related Work

There are many lines of work that focus on analyzing and understanding representations. The most commonly used technique is the classifier-based method. Early work (Alain and Bengio, 2017; Kulmizev et al., 2020) starts with using linear classifiers as the probe. Hewitt and Liang (2019) pointed out that a linear probe is not sufficient to evaluate a representation. Some recent work also employ non-linear probes (Tenney et al., 2019; Eger et al., 2019). There are also efforts to inspect the representations from a geometric persepctive (e.g. Ethayarajh, 2019; Mimno and Thompson, 2017), including the recently proposed DIRECTPROBE (Zhou and Srikumar, 2021), which we use in this work. Another line of probing work designs control tasks (Ravichander et al., 2021; Lan et al., 2020) to reverse-engineer the internal mechanisms of representations (Kovaleva et al., 2019; Wu et al., 2020). However, in contrast to our work, most studies (Zhong et al., 2021; Li et al., 2021; Chen et al., 2021) focused on pre-trained representations, not fine-tuned ones.

While fine-tuning pre-trained representations usually provides strong empirical performance (Wang et al., 2018; Talmor et al., 2020), how fine-tuning manage to do so has remained an open question. Moreover, the instability (Mosbach et al., 2020a; Dodge et al., 2020; Zhang et al., 2020) and forgetting problems (Chen et al., 2020; He et al., 2021) make it harder to analyze fine-tuned representations. Despite these difficulties, previous work (Merchant et al., 2020; Mosbach et al., 2020b; Hao et al., 2020) draw valuable conclusions about fine-tuning. This work extends this line of effort and provides a deeper understanding of how fine-tuning changes representations.

7 Conclusions

In this work, we take a close look at how fine-tuning a contextualized representation for a task modifies it. We investigate the fine-tuned representations of several BERT models using two probing techniques: classifier-based probing and DIRECTPROBE. First, we show that fine-tuning introduces divergence between training and test set, and in at least one case, hurts generalization. Next, we show fine-tuning alters the geometry of a representation by pushing points belonging to the same label closer to each other, thus simpler and better classifiers. We confirm this hypothesis by cross-task fine-tuning experiments. Finally, we discover that while adjusting representations to downstream tasks, fine-tuning largely preserves the original spatial structure of points across all layers. Taken collectively, the empirical study presented in this work can not only justify the impressive performance of fine-tuning, but may also lead to a better understanding of learned representations.

Acknowledgments

We thank the ARR reviewers and the Utah NLP group for their constructive feedback. This work is partially supported by NSF grants #1801446 (SaTC) and #1822877 (Cyberlearning), and a generous gift from Verisk Inc.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7319–7328, Online. Association for Computational Linguistics.

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Yonatan Belinkov. 2021. Probing classifiers: Promises, shortcomings, and alternatives. *CoRR*, abs/2102.12452.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27.

Boli Chen, Yao Fu, Guangwei Xu, Pengjun Xie, Chuanqi Tan, Mosha Chen, and Liping Jing. 2021. Probing BERT in hyperbolic spaces. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of* *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305.

Steffen Eger, Andreas Rücklé, and Iryna Gurevych. 2019. Pitfalls in the evaluation of sentence embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 55–60, Florence, Italy. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2020. Investigating learning dynamics of BERT fine-tuning. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 87–92, Suzhou, China. Association for Computational Linguistics.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. 2021. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1121–1133, Online. Association for Computational Linguistics.

John Hewitt, Kawin Ethayarajh, Percy Liang, and Christopher D. Manning. 2021. Conditional probing: measuring usable information beyond a baseline. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1626–1639. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Mael Jullien, Marco Valentino, and André Freitas. 2022. Do transformers encode a foundational ontology? probing abstract classes in natural language. *CoRR*, abs/2201.10262.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online. Association for Computational Linguistics.

Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel R. Bowman, and Ellie Pavlick. 2019. Probing what different NLP tasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 235–249, Minneapolis, Minnesota. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Katarzyna Krasnowska-Kieraś and Alina Wróblewska. 2019. Empirical linguistic study of sentence embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5729–5739, Florence, Italy. Association for Computational Linguistics.

Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do neural language models show preferences for syntactic formalisms? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4077–4091, Online. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bai Li, Zining Zhu, Guillaume Thomas, Yang Xu, and Frank Rudzicz. 2021. How is BERT surprised? layerwise detection of linguistic anomalies. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4215–4228, Online. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to BERT embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 33–44, Online. Association for Computational Linguistics.

David Mimno and Laure Thompson. 2017. The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2873–2878, Copenhagen, Denmark. Association for Computational Linguistics.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020a. On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. *CoRR*, abs/2006.04884.

Marius Mosbach, Anna Khokhlova, Michael A. Hedderich, and Dietrich Klakow. 2020b. On the interplay between fine-tuning and sentence-level probing for linguistic knowledge in pre-trained transformers. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 68–82, Online. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. 2021. Probing the probing paradigm: Does probing accuracy entail task relevance? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3363–3377, Online. Association for Computational Linguistics.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R. Moeller, Aviram Stern, Adi Bitan, and Omri Abend. 2018. Comprehensive supersense disambiguation of English prepositions and possessives. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 185–196, Melbourne, Australia. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. olmpics - on what language model pre-training captures. *Trans. Assoc. Comput. Linguistics*, 8:743–758.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International*

Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

William F Whitney, Min Jae Song, David Brandfonbrener, Jaan Altosaar, and Kyunghyun Cho. 2021. Evaluating representations by the complexity of learning low-loss predictors. In *Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics.

Yadollah Yaghoobzadeh, Katharina Kann, T. J. Hazen, Eneko Agirre, and Hinrich Schütze. 2019. Probing for semantic classes: Diagnosing the meaning content of word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5740–5753, Florence, Italy. Association for Computational Linguistics.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online. Association for Computational Linguistics.

Yichu Zhou and Vivek Srikumar. 2021. DirectProbe: Studying representations without classifiers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5070–5083, Online. Association for Computational Linguistics.

A Fine-tuning Details

In this work, we fine-tune all tasks and representations using HuggingFace library. We use a linear weight schduler with a learning rate of $3e^{-4}$, which uses 10% of the total update steps as the warmup steps. The same schduler is used for all tasks. All the models are optimized by Adam (Kingma and Ba, 2015) with batch size of 32. All the fine-tuning is run on a single Titan GPU. The best hidden-layer sizes for each task are shown in Table 7.

B Summary of Tasks

In this work, we conduct experiments on five NLP tasks, which are chosen to cover different usages of the representations we study. Table 6 summarizes these tasks.

C Probing Performance

Table 7 shows the complete table of probing results in our experiments. The last column is the spatial similarity between the training set and test set. Some entries are missing because the similarity can only be computed on the representations that are linearly separable for the given task.

D Dynamics of Minimum Distances

Figure 6 shows the dynamics of minimum distances for labels on all four tasks. For clarity, we only present the distances for the three labels where the distances increase the most and the three where it decreases the most.

E PCA Projections of the Movements

Figures 7–10 show the PCA projections of the difference vector between the centroids of labels before and after fine-tuning based on BERT$_{\text{base}}$.

Task	#Training	#Test	#Labels	Token-based	Sentence-based	Pair-wise	Semantic	Syntax
Supersense-role	4282	457	47	√			√	
Supersense-function	4282	457	40	√			√	
POS	16 860	4323	17	√				√
Dependency Relation	16 054	4122	46			√		√
TREC-50	5452	500	50		√		√	

Table 6: Statistics of the five tasks with their different characteristics.

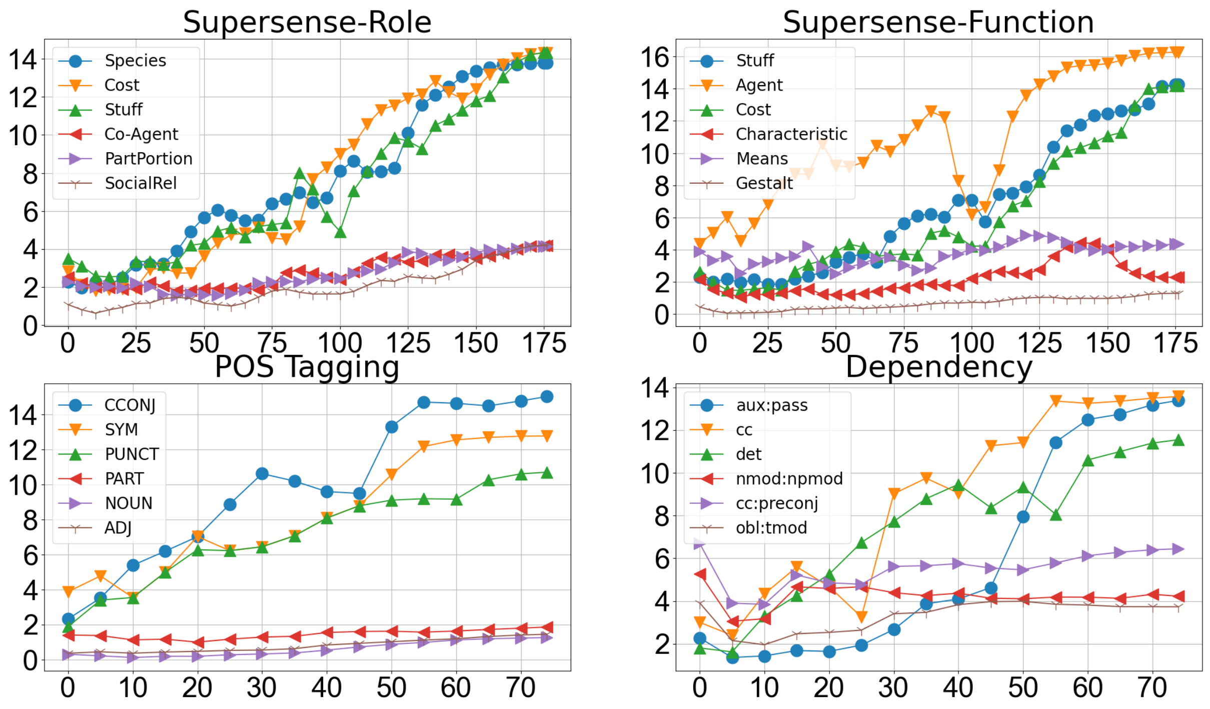

Figure 6: The dynamics of the minimum distance of the three labels where the distance increases the most, and three labels where is increases the least. The horizontal axis is the number of fine-tuning updates; the vertical axis is chosen label's minimum distance to other labels. These results come from the last layer of BERT$_{\text{base}}$.

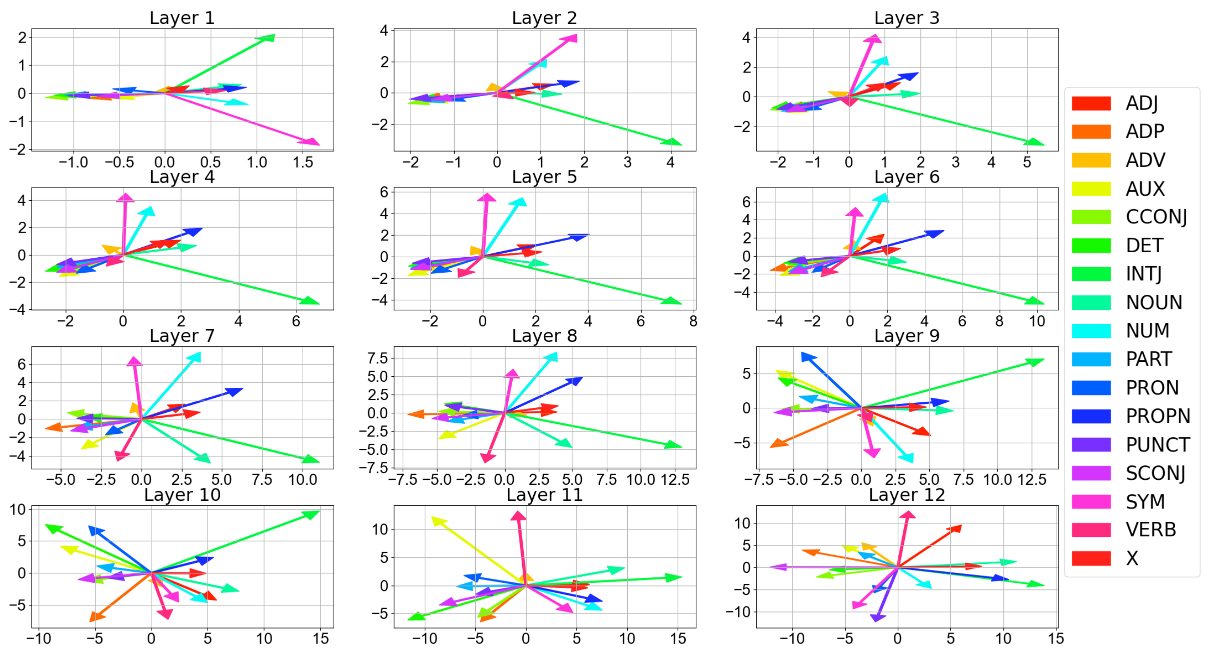

Figure 7: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on POS tagging task and BERT$_{\text{base}}$.

Representations	Task		Acc	Std	Best Layer Size	#Cluster	is Linear	Similarity
BERT$_{\text{tiny}}$	POS	original	90.76	0.24	(256, 64)	30	N	-
		fine-tuned	91.67	0.29	(64, 64)	18	N	-
	DEP	original	86.74	0.22	(256, 256)	50	N	-
		fine-tuned	89.04	0.20	(256, 256)	46	Y	0.88
	PS-fxn	original	74.14	1.42	(256, 256)	42	N	-
		fine-tuned	74.40	0.68	(256, 128)	40	Y	0.72
	PS-role	original	58.38	0.78	(256, 64)	46	Y	0.76
		fine-tuned	60.31	0.29	(64, 64)	46	Y	0.70
	TREC-50	original	68.12	0.82	(256, 256)	58	N	-
		fine-tuned	84.04	0.93	(256, 256)	51	N	-
BERT$_{\text{mini}}$	POS	original	93.81	0.10	(256, 32)	19	N	-
		fine-tuned	94.91	0.03	(256, 32)	17	Y	0.70
	DEP	original	91.82	0.09	(256, 128)	46	Y	0.93
		fine-tuned	93.55	0.07	(256, 128)	46	Y	0.86
	PS-fxn	original	82.45	1.07	(256, 256)	40	Y	0.77
		fine-tuned	84.25	0.39	(256, 128)	40	Y	0.53
	PS-role	original	68.05	1.08	(256, 256)	46	Y	0.81
		fine-tuned	71.90	1.06	(256, 64)	46	Y	0.59
	TREC-50	original	74.12	1.25	(256, 256)	52	N	-
		fine-tuned	88.36	0.50	(64, 32)	52	N	-
BERT$_{\text{small}}$	POS	original	94.26	0.13	(256, 32)	17	Y	0.96
		fine-tuned	95.43	0.06	(128, 64)	17	Y	0.72
	DEP	original	92.93	0.14	(256, 64)	46	Y	0.93
		fine-tuned	94.48	0.14	(256, 64)	46	Y	0.78
	PS-fxn	original	86.26	0.54	(256, 256)	40	Y	0.82
		fine-tuned	85.08	0.35	(256, 256)	40	Y	0.44
	PS-role	original	74.22	1.03	(256, 256)	46	Y	0.84
		fine-tuned	74.57	0.61	(128, 128)	46	Y	0.54
	TREC-50	original	81.32	0.61	(256, 128)	52	N	-
		fine-tuned	89.60	0.22	(256, 64)	51	N	-
BERT$_{\text{medium}}$	POS	original	94.40	0.08	(256, 128)	17	Y	0.97
		fine-tuned	95.56	0.05	(64, 32)	17	Y	0.67
	DEP	original	92.54	0.14	(256, 256)	46	Y	0.94
		fine-tuned	94.76	0.20	(128, 128)	46	Y	0.79
	PS-fxn	original	86.56	0.41	(256, 128)	40	Y	0.80
		fine-tuned	88.45	0.45	(128, 256)	40	Y	0.59
	PS-role	original	76.28	1.00	(256, 32)	46	Y	0.83
		fine-tuned	78.86	0.58	(128, 128)	46	Y	0.58
	TREC-50	original	80.68	1.16	(256, 64)	52	N	-
		fine-tuned	89.80	0.33	(32, 64)	52	N	-
BERT$_{\text{base}}$	POS	original	93.39	0.31	(256, 128)	17	Y	0.97
		fine-tuned	95.68	0.02	(128, 64)	17	Y	0.70
	DEP	original	89.39	0.08	(256, 128)	46	Y	0.92
		fine-tuned	94.76	0.05	(64, 256)	46	Y	0.76
	PS-fxn	original	87.75	0.41	(256, 128)	40	Y	0.84
		fine-tuned	89.58	0.67	(32, 256)	40	Y	0.57
	PS-role	original	74.49	0.84	(256, 128)	46	Y	0.82
		fine-tuned	81.14	0.26	(256, 128)	46	Y	0.52
	TREC-50	original	85.24	0.85	(256, 128)	52	N	-
		fine-tuned	90.36	0.32	(64, 32)	51	N	-

Table 7: A complete table of the probing results of five representations on five tasks.

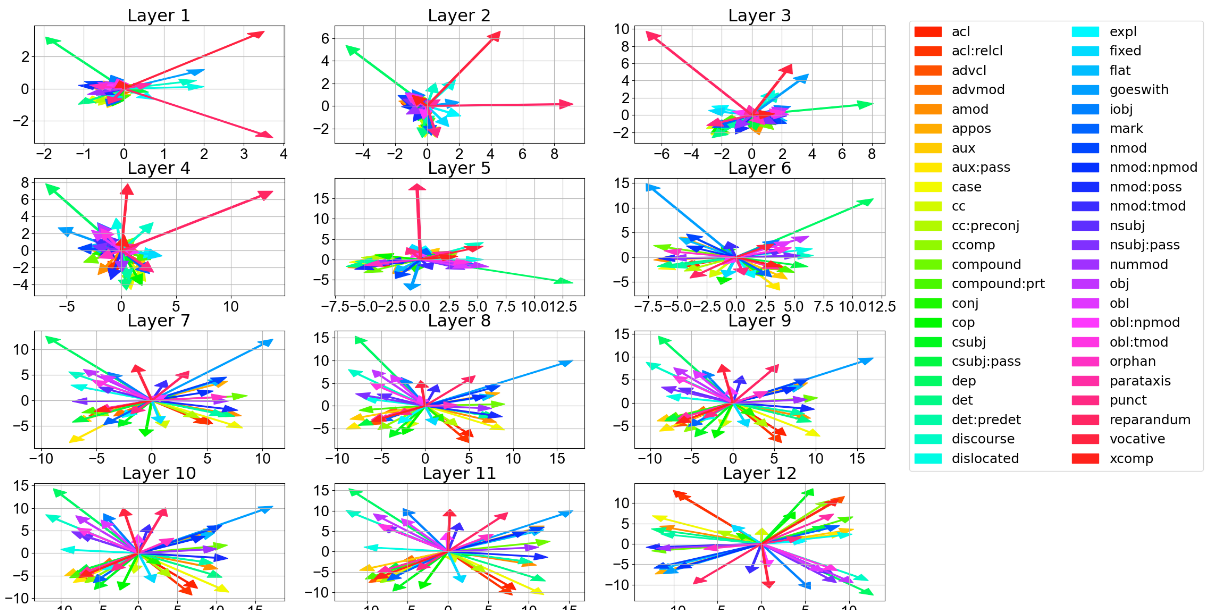

Figure 8: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on dependency prediction task and BERT$_{\text{base}}$.

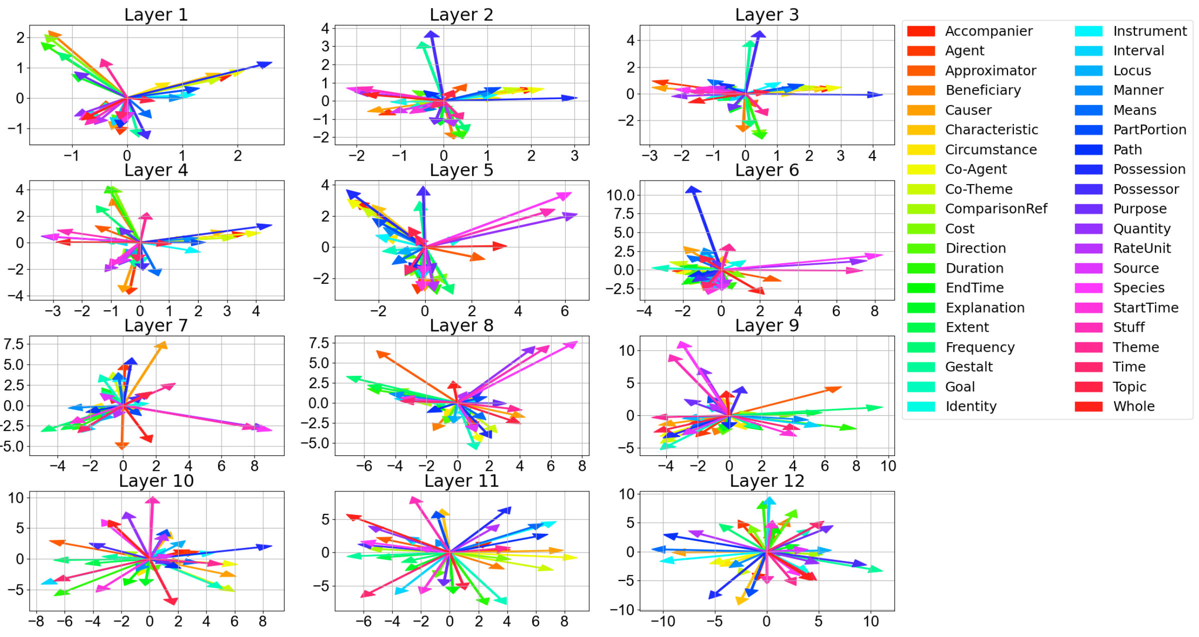

Figure 9: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on Supersense function task and BERT$_{\text{base}}$.

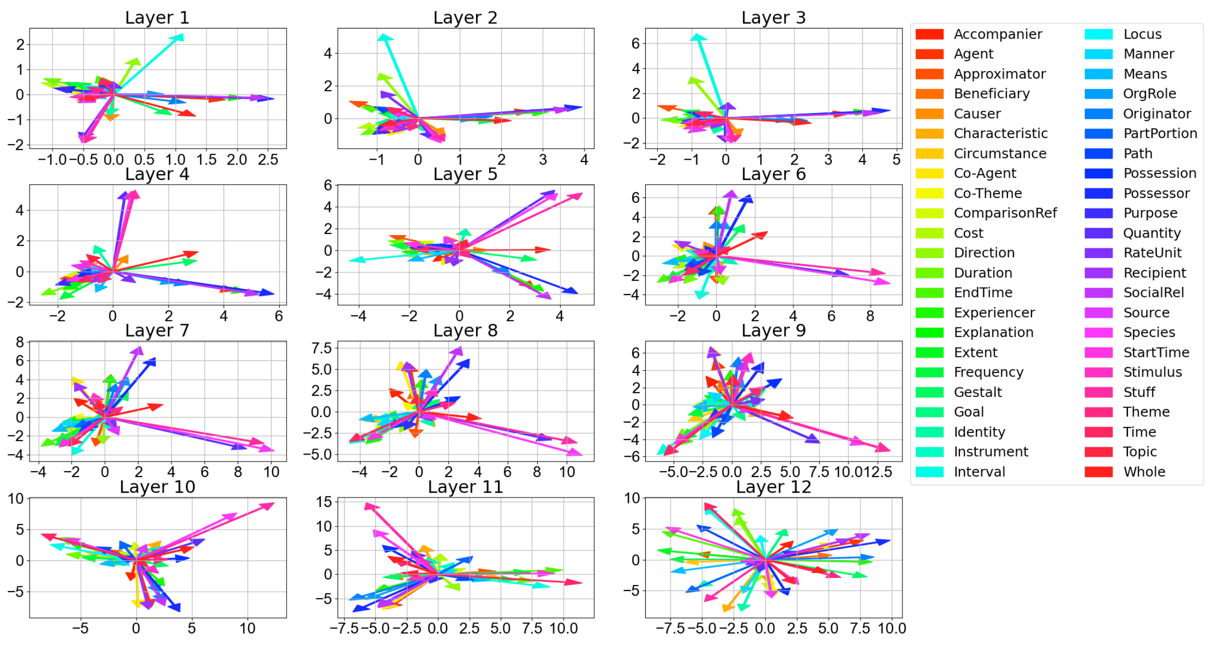

Figure 10: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on Supersense role task and BERT$_{\text{base}}$.

F Cluster Number Revision

We discovered a bug in the implementation of DIRECTPROBE which causes the merging to stop early while the remaining clusters are still mergeable. The main paper (Table 3, Table 4, and Table 7) has been updated to report the correct results. Table 8 shows the original results.

This bug does not change the natural of the linearity of datasets and representations. All the findings from original experiments remain the same. This bug only affects the number of clusters when the representation is non-linear for a given task.

Representations	Task		Acc	Std	Best Layer Size	#Cluster	is Linear	Similarity
BERT$_{\text{tiny}}$	POS	original	90.76	0.24	(256, 64)	3936	N	-
		fine-tuned	91.67	0.29	(64, 64)	20	N	-
	DEP	original	86.74	0.22	(256, 256)	653	N	-
		fine-tuned	89.04	0.20	(256, 256)	46	Y	0.88
	PS-fxn	original	74.14	1.42	(256, 256)	402	N	-
		fine-tuned	74.40	0.68	(256, 128)	40	Y	0.72
	PS-role	original	58.38	0.78	(256, 64)	46	Y	0.76
		fine-tuned	60.31	0.29	(64, 64)	46	Y	0.70
	TREC-50	original	68.12	0.82	(256, 256)	399	N	-
		fine-tuned	84.04	0.93	(256, 256)	51	N	-
BERT$_{\text{mini}}$	POS	original	93.81	0.10	(256, 32)	2429	N	-
		fine-tuned	94.91	0.03	(256, 32)	17	Y	0.70
	DEP	original	91.82	0.09	(256, 128)	46	Y	0.93
		fine-tuned	93.55	0.07	(256, 128)	46	Y	0.86
	PS-fxn	original	82.45	1.07	(256, 256)	40	Y	0.77
		fine-tuned	84.25	0.39	(256, 128)	40	Y	0.53
	PS-role	original	68.05	1.08	(256, 256)	46	Y	0.81
		fine-tuned	71.90	1.06	(256, 64)	46	Y	0.59
	TREC-50	original	74.12	1.25	(256, 256)	127	N	-
		fine-tuned	88.36	0.50	(64, 32)	52	N	-
BERT$_{\text{small}}$	POS	original	94.26	0.13	(256, 32)	17	Y	0.96
		fine-tuned	95.43	0.06	(128, 64)	17	Y	0.72
	DEP	original	92.93	0.14	(256, 64)	46	Y	0.93
		fine-tuned	94.48	0.14	(256, 64)	46	Y	0.78
	PS-fxn	original	86.26	0.54	(256, 256)	40	Y	0.82
		fine-tuned	85.08	0.35	(256, 256)	40	Y	0.44
	PS-role	original	74.22	1.03	(256, 256)	46	Y	0.84
		fine-tuned	74.57	0.61	(128, 128)	46	Y	0.54
	TREC-50	original	81.32	0.61	(256, 128)	113	N	-
		fine-tuned	89.60	0.22	(256, 64)	51	N	-
BERT$_{\text{medium}}$	POS	original	94.40	0.08	(256, 128)	17	Y	0.97
		fine-tuned	95.56	0.05	(64, 32)	17	Y	0.67
	DEP	original	92.54	0.14	(256, 256)	46	Y	0.94
		fine-tuned	94.76	0.20	(128, 128)	46	Y	0.79
	PS-fxn	original	86.56	0.41	(256, 128)	40	Y	0.80
		fine-tuned	88.45	0.45	(128, 256)	40	Y	0.59
	PS-role	original	76.28	1.00	(256, 32)	46	Y	0.83
		fine-tuned	78.86	0.58	(128, 128)	46	Y	0.58
	TREC-50	original	80.68	1.16	(256, 64)	110	N	-
		fine-tuned	89.80	0.33	(32, 64)	52	N	-
BERT$_{\text{base}}$	POS	original	93.39	0.31	(256, 128)	17	Y	0.97
		fine-tuned	95.68	0.02	(128, 64)	17	Y	0.70
	DEP	original	89.39	0.08	(256, 128)	46	Y	0.92
		fine-tuned	94.76	0.05	(64, 256)	46	Y	0.76
	PS-fxn	original	87.75	0.41	(256, 128)	40	Y	0.84
		fine-tuned	89.58	0.67	(32, 256)	40	Y	0.57
	PS-role	original	74.49	0.84	(256, 128)	46	Y	0.82
		fine-tuned	81.14	0.26	(256, 128)	46	Y	0.52
	TREC-50	original	85.24	0.85	(256, 128)	162	N	-
		fine-tuned	90.36	0.32	(64, 32)	51	N	-

Table 8: Original table of the probing results of five representations on five tasks. These results were in the original version of the paper before we found a bug in the implementation of DIRECTPROBE. The updated results are in Table 7. See Appendix C for details.