Predicting Fine-Tuning Performance with Probing

Anonymous ACL submission

Abstract

Large NLP models have recently shown impressive performance in language understanding tasks, typically evaluated by fine-tuning tasks. Alternatively, probing has received increasing attention as being a lightweight method for interpreting the intrinsic mechanisms of large NLP models. In probing, post-hoc classifiers are trained on "out-of-domain" datasets that diagnose specific abilities. While probing the language models has led to insightful findings, they appear disjointed from the development of models. This paper explores the utility of probing deep NLP models to extract a proxy signal widely used in model developments, the fine-tuning performance. We find that it is possible to use the accuracies of only three probing results to predict the fine-tuning performance with errors 40% - 80% smaller than baselines. We further show the possibility of incorporating specialized probing datasets into developing deep NLP models.

1 Introduction

Large-scale neural models have recently demonstrated state-of-the-art performance in a wide variety of tasks, including sentiment detection, paraphrase detection, linguistic acceptability, and entailment detection (Devlin et al., 2019; Radford et al., 2019; Peters et al., 2018). Developing systems for these tasks usually involves two stages: a pre-training stage, where the large neural models gain linguistic knowledge from weak supervision signals in massive corpora, and a fine-tuning stage, where the models acquire task-specific knowledge from labeled data. The fine-tuning results are widely used to benchmark the performances of neural models and refine the models' development procedures.

However, these fine-tuning results are summary statistics and do not paint the full picture of deep neural models (Ethayarajh and Jurafsky, 2020; Bender and Koller, 2020). As researchers are increasingly concerned about interpreting the intrinsic mechanisms of deep neural models, many data-driven assessment methods have been developed. These assessments usually follow the route of compiling a targeted dataset and running post-hoc analyses. Until now, one of the most popular interpretation methods is referred to as *probing*. To probe a neural model, one uses a predictor to obtain the labels from the representations that are embedded using the neural model. Probing analyses on deep neural models revealed some low-dimensional syntactic structures (Hewitt and Manning, 2019), common-sense knowledge (Petroni et al., 2019) and (to some extent) human-like abilities, including being surprised upon witnessing linguistic irregularity (Li et al., 2021) and reasoning about space and time (Aroca-Ouellette et al., 2021).

From the viewpoint of data-driven assessments, both fine-tuning and probing can reveal the abilities of deep neural networks, but they appear to steer towards different directions:

In-domain vs. out-of-domain. Fine-tuning uses in-domain data – we evaluate the models on the same distributions as those where we deploy them. Probing, however, uses out-domain data: instead of simulating the deployment environment, the targeted datasets focus on diagnosing specific abilities.

Inclusive vs. specific. In fine-tuning, edge cases should be included, so the unexpected behavior after deployment can be minimized (Ribeiro et al., 2020) and the fine-tuning results can be stable (Zhang et al., 2021). On the contrary, the probing datasets are more specialized, so smaller datasets suffice[1].

High performances vs. faithful interpretations. While fine-tuning methods are mainly studied from

[1] Another viewpoint for the dataset requirement can be derived from learning theory. Loosely speaking, optimizing more parameters requires more data to reach stable results. Fine-tuning involves more parameters than probing. Anonymity (2022) provides a more quantitative discussion.

an algorithmic perspective to enhance the performance of language models, probing methods aim at assessing the faithfulness of language models. To fulfill the former objective, fine-tuning is accompanied by efforts in pre-training, collecting more data, building better representations, and exploring novel model architectures (He et al., 2021; Sun et al., 2021; Wang et al., 2021b; Jiang et al., 2020). Conversely, the latter goal is pursued by borrowing inspirations from a variety of other sources, including psycholinguistic assessment protocols (Futrell et al., 2019), information theory (Voita and Titov, 2020; Pimentel and Cotterell, 2021), and causal analysis (Slobodkin et al., 2021; Elazar et al., 2021).

In short, probing assessments are more specialized (therefore more flexible) and less computationally expensive. In contrast, the performance scores of fine-tuning assessments are more relevant to the design and training of deep neural models. **Can probing be used in the development of deep neural models?** This question involves two aspects:

- *Feasibility*: Are probing results relevant in the model development?
- *Operation*: How to set up probing analyses to get these useful results?

This paper attempts to answer both. For feasibility, we show that a crucial feedback signal in model development, the fine-tuning performance, can be predicted via probing results, indicating a positive answer to the feasibility question.

For operation, we run extensive ablation studies to simplify the probing configurations, leading to some heuristics to set up probing analyses. We start with a battery of probing tasks and evaluate the utilities both task-wise and layer-wise (§5.2 - §5.3). We then reduce the number of probing configurations, showing that as few as 3 configurations can predict fine-tuning results with RMSEs between 40% and 80% smaller than the control baseline (§5.5). To further answer the operation question, we run ablation studies on different probing configurations, including probing methods (§5.6) and the number of data samples (§5.7). We also analyze the uncertainty of the results (§5.8). Our analysis shows the possibility of using probing in developing high-performance deep neural models.

2 Related Work

Performance prediction Xia et al. (2020) proposed a framework that predicts task performance using a collection of features, including the hyperparameters of the model and the percentage of text overlap between the source and target datasets. Srinivasan et al. (2021) extended this framework into a multilingual setting. Ye et al. (2021) considered the reliability of performance – an idea similar to that of Dodge et al. (2019). Our experiments differ from the performance prediction literature, mainly regarding the features used. Instead of deriving features from the datasets and the model hyperparameters, we focus on the utility of probing results as features themselves.

Domain generalization The domain generalization literature provides a variety of methods to improve the performance of out-of-domain classification. We defer to Wang et al. (2021a) for a summary. Gulrajani and Lopez-Paz (2020) ran empirical comparisons on many algorithms, and some theoretical analyses bound the performance of out-of-domain classification (Li et al., 2022; Minsker and Mathieu, 2019). In our setting, the probing and the fine-tuning datasets can be considered different domains, but we aim to predict performance.

Probing, and the utility of LODNA The probing literature reveals various abilities of deep neural models, as summarized by Rogers et al. (2020); Manning et al. (2020); Belinkov (2021); Pavlick (2022). There have been some discussions on the utility of probing results. Baroni (2021) argued that these linguistic-oriented deep neural network analyses (LODNA) should treat deep neural models as algorithmic linguistic theories; otherwise, LODNA has limited relevance to theoretical linguists. Recent literature in LODNA drew interesting findings by comparing the mechanisms in which algorithms and humans respond to external stimuli, including the relative importance of sentences (Hollenstein and Beinborn, 2021). Probing results, when used jointly with evidence from datasets, can also be used to predict the inductive bias of neural models (Lovering et al., 2021; Immer et al., 2021). As we show, probing results can explain the variance in and even predict the fine-tuning performance of neural NLP models.

Fine-tuning and probing There have been multiple papers that explored fine-tuning and probing paradigms. Probing is used as a post-hoc method to interpret linguistic knowledge in deep neural models during pre-training (Liu et al., 2019a), fine-tuning (Miaschi et al., 2020; Mosbach et al., 2020;

Durrani et al., 2021; Yu and Ettinger, 2021; Zhou and Srikumar, 2021), and other stages of model development (Ebrahimi et al., 2021). From a performance perspective, probing can sometimes result in higher performance metrics (e.g., accuracy) than fine-tuning (Liu et al., 2019a; Hall Maudslay et al., 2020). We take a different perspective, considering how the signals of probing can be helpful towards developing large neural models.

3 Methods

We present the overall analysis method and evaluation metric in this section. §5 elaborates the detailed experiment settings.

Predicting fine-tuning performance A deep neural model M can be fine-tuned on task T to achieve performance $\mathcal{A}_T$. Let $\mathbf{S} \in \mathbb{R}^N$ be the test accuracies of probing classifications on model M, using N configurations. For example, a deep neural model M = RoBERTa can be fine-tuned to reach performance $\mathcal{A}_T = 0.85$ on a T = RTE task. With post-hoc classifiers applied to the 12 layers of M, we can probe for 12 test accuracies on a probing task (e.g., detecting the past vs. present tense), which constitute of $\mathbf{S}$.

There are K models in total. The collected probing results $\{\mathbf{S}^{(k)}\}_{k=1}^{K}$ can be used to predict the fine-tuning performance $\{\mathcal{A}_T^{(k)}\}_{k=1}^{K}$ via regression. Formally, this procedure optimizes for $N+1$ parameters, $\theta \in \mathbb{R}^{N+1}$ so that:

$$\theta_* = \text{argmin}_\theta \Sigma_k ||\theta^T \mathbf{S}^{(k)} - \mathcal{A}_T^{(k)}||^2 \quad (1)$$

This procedure has closed-form solutions that are implemented in various scientific computation toolkits (e.g., `R` and `scipy`). The minimum reachable RMSE is therefore:

$$\text{RMSE} = \sqrt{\frac{1}{K}\Sigma_k ||{\theta_*}^T \mathbf{S}^{(k)} - \mathcal{A}_T^{(k)}||^2} \quad (2)$$

RMSE-reduction While RMSE can evaluate the quality of this regression, it is insufficient for measuring the informativeness of $\mathbf{S}$ due to the discrepancy among the fine-tuning tasks T. Suppose we have two tasks, T_1 and T_2, where the probing results $\mathbf{S}$ can support high-precision regressions to RMSE $= 0.01$ on both tasks. However, on T_1, even features drawn from random distributions[2] might be sufficient to reach RMSE $= 0.02$, while on the more difficult task, T_2, random features could only reach RMSE $= 0.10$ maximum. The probing results $\mathbf{S}$ is more useful for T_2 than T_1, but RMSE itself does not capture this difference.

Considering this, we should further adjust against a baseline, the minimum reachable RMSE using random features.

$$\theta_{c*} = \text{argmin}_\theta \Sigma_k ||\theta^T \epsilon^{(k)} - \mathcal{A}_T^{(k)}||^2, \quad (3)$$

where the random features ϵ are drawn from $\mathcal{N}(0, 0.1)$. Overall, the RMSE and the reduction from the baseline are computed as:

$$\text{RMSE}_c = \sqrt{\frac{1}{K}\Sigma_k ||\theta_{c*}^T \epsilon^{(k)} - \mathcal{A}_T^{(k)}||^2} \quad (4)$$

$$\text{RMSE_reduction} = \frac{\text{RMSE}_c - \text{RMSE}}{\text{RMSE}_c} \times 100 \quad (5)$$

In the experiments, all RMSE and RMSE$_c$ values follow 5-fold cross validation. We report the RMSE_reduction as the score that measures the utility of $\mathbf{S}$.

4 Data and Material

4.1 Fine-tuning tasks

We consider 6 binary classification tasks in GLUE (Wang et al., 2019) as fine-tuning tasks: `RTE` consists of a collection of challenges recognizing textual entailment. Given two sentences, the model decides whether a sentence entails the other. `COLA` (Warstadt et al., 2019) requires the model to determine if a sentence is linguistically acceptable. `MRPC` (Dolan and Brockett, 2005) requires the model to identify if a pair of sentences are paraphrases. `SST2` (Socher et al., 2013) asks the model to output the sentiment positivity of movie reviews. `QNLI` contains questions and answers parsed from SQuAD (Rajpurkar et al., 2016). This task requires the model to decide whether the *answer* answers the *question*. `QQP`[3] tests if the model can correctly output whether a pair of Quora questions are synonymous.

4.2 Probing tasks

We use 7 probing tasks in SentEval (Conneau and Kiela, 2018):

[2] Considering the small data sizes (i.e., the total number of models studied), even the "random features" drawn from random noises contain artefacts – patterns that can be used to regress the results.

[3] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

- Syntactic: bigram shift (BShift), and tree depth (TreeDepth)
- Semantic: past present (Tense), subject number (SubjNum), object number (ObjNum), semantic odd-man out (SOMO), and coordination inversion (CoordInv)

These probing tasks span across a range of linguistic abilities. In general, layers closer to the inputs (lower layers) in BERT contain more surface-level information, whereas higher layers contain more syntactic and semantic information (Tenney et al., 2019; Jawahar et al., 2019), but the actual location of different linguistic features may vary (Miaschi et al., 2020).

For each probing task, we randomly sample 1200 data points per class, corresponding to around 1% of the original SentEval data.

4.3 Pre-trained Language Models

We use several pre-trained language models as foundation models (Bommasani et al., 2021) for fine-tuning and probing. We refer to the models by their names on the Huggingface Model Hub[4].

`roberta-base` (Liu et al., 2019b) pretrains BERT (Devlin et al., 2019) on over 160GB of English corpora, using improved techniques including dynamic masking, large mini-batches and masked language modeling without next-sentence-prediction.

`xlm-roberta-base` (Conneau et al., 2020) is pre-trained on 2.5TB of Common Crawl data from over 100 languages. The multiple languages sources improve the transferability across languages while compromising only a little accuracy on the English GLUE tasks (compared to the monolingual RoBERTa).

`albert-base-v2` (Lan et al., 2020) shares parameters across layers and decomposes the vocabulary matrices into smaller matrices. These parameter-reducing techniques reduce the computation resource requirements, which allows the model pretraining to further scale-up.

`microsoft/deberta-base` (He et al., 2021) uses separate attention vectors to model the content and the positions of each word. During fine-tuning, DeBERTa adds adversarial perturbations to the normalized embeddings.

`xlnet-base-cased` (Yang et al., 2019) models different permutation orders of the contexts during pre-training. XLNet additionally uses attentions to keep track of previous states, allowing the model to process the contexts extending beyond fixed lengths.

`Corrupted models`. To increase the diversity of models, we corrupt the foundation models on a masked language modeling task using scrambled Wikipedia for 500, $1k$, $2k$, $4k$, and $6k$ steps. This "model augmentation" procedure does not apply to XLNet because scrambling the corpus produces a permutation of context, which XLNet already models. In total, there are 25 foundation models, each containing 12 layers.

4.4 Fine-tuning methods

For all fine-tuning classifications, we use the *AutoModelForSequenceClassification* framework by huggingface Transformers (Wolf et al., 2020). The model is trained with an AdamW optimizer with a collection of initial learning rates[5] and a batch size of 4. Since the GLUE tasks do not publicize the test set labels, we use the best dev set performance as the fine-tuning results. For reproducibility, we set the random seed to 42 in PyTorch (Paszke et al., 2019). Additional details, including runtime and computation resources, are in Appendix A.

4.5 Probing methods

There are many methods to probe a neural network. In this paper, we use a post-hoc classifier to predict a target ("probing task target") from the representations of the neural network. We run through a collection of scikit-learn (Pedregosa et al., 2011) classifiers[6], choose the best one by the validation accuracy, and take its test accuracy as the probing result. Additional details, including runtime and computation resources, are in Appendix A.

5 Experiments

5.1 Fine-tuning performance

As an exploratory analysis, Figure 1 plots the distributions of the GLUE fine-tuning performances. The additional corruption steps result in more significant fine-tuning performance drops on RTE and MRPC than other tasks. Moreover, the dev accuracies of `roberta-base` (and its corrupted mod-

[4] `https://huggingface.co/`

[5] 1e-4, 3e-5, 1e-5, 3e-6, 1e-6, 3e-7. Note that most highest-performing classification runs are reached with either 1e-5 or 3e-6.

[6] Logistic Regression, MLP with 10 and 20 hidden nodes, Random Forest with 100 and 10 estimators, Decision Tree, and SVM. There are 7 classifiers in total.

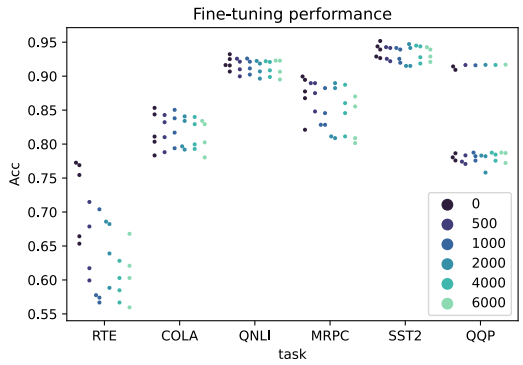

Figure 1: Fine-tuning performance. The color coding reflects the number of corruption steps on scrambled wikipedia, with 0 corresponding to the "vanilla" foundation models.

els) are larger than 0.90, while most other models have around 0.80 dev accuracies.

5.2 Which probing task is most informative?

We start with testing the predictability of using the results from only one probing task. For each probing task, we concatenate the 12 probing results as features and predict the fine-tuning performance using linear regression[7].

Table 1 shows the percentage of RMSE reduction from baseline, using all layers from one probing task. There is no definitive answer towards "which probing task best predicts all fine-tuning tasks" but, depending on the linguistic abilities that each task targets, there are some regularities. For example, the 'number counting' probing tasks do not predict the fine-tuning performances on RTE, the textual entailment recognition task. In other fine-tuning tasks (COLA, QNLI, MRPC, SST2, QQP), however, each probing task shows positive RMSE reduction, signaling the ability to predict fine-tuning performance.

5.3 Which layers are the most indicative?

In the regression experiments of §5.2, we considered each feature equally important. However, a one-way ANOVA shows that some layers are more indicative than others, as Table 2 shows. For example, the probing results of tree_depth (at layer 1) and object_number (at layer 1) explain significant variance on all fine-tuning tasks.

Note that the layers with the most predictability should not be confused with those containing the richest linguistic knowledge. The former corresponds to the probing results that explain the most variances, while the latter corresponds with probing with the highest accuracy.

5.4 Only one layer per probing task

Instead of probing all 12 layers, could using the probing results from only one layer for each probing task be beneficial? Following Table 2, we use the layers that are shown to explain significant portions of variance[8].

The results are also included in Table 1. When reducing the number of features into around half (12 to 7), "one layer per probing task" can reduce more RMSE in RTE and SST2[9]. However, the results in other fine-tuning tasks indicate that alternative feature selection methods might help find a more predictive feature set.

5.5 Can we predict with only 3 features?

This experiment further reduces the number of features used while maximizing the MSE reductions. We iterate through all possible combinations of the $12 \times 7 = 84$ probing features for each fine-tuning task and report the largest RMSE reduction in predicting the fine-tuning performance.

Table 1 shows the results and the corresponding features. The prediction with three features can reduce the most RMSE on RTE, SST2, QNLI, and QQP. On COLA and MRPC, the RMSE reductions by the top three features are at most 6% smaller than those of the best previous configurations, which involved many more probing features.

The results show the utility of probing. It is possible to predict the fine-tuning performances by probing on as few as three configurations (each configuration using one probing task on one layer).

5.6 Ablation: probing configuration

To further simplify the probing procedure, we run this ablation study. Instead of probing using a battery of post-hoc classifiers, we just specify one classifier as probe, and test if the probing results can reproduce the findings of §5.2 - §5.5.

[7] `lm` method in the `caret` R package.

[8] Namely, layers 5, 6, 1, 5, 1, 1, 1 from the 7 probing tasks respectively.

[9] Note that the ANOVA in §5.3 use all data samples, so the choice of the features contain information propagated from the validation set. To ensure fair comparisons, we do *not* include the results from "one-layer-per-task" setting when finding the highest RMSE reductions in subsequent analysis. The results from this setting do not outperform the bold-font results even once, though.

	RTE	COLA	MRPC	SST2	QNLI	QQP
All layers one task (§5.2)						
BShift	6.24	52.80	**53.18**	29.78	55.29	51.64
CoordInv	2.10	66.59	18.18	44.24	56.35	56.57
ObjNum	2.19	44.20	28.02	53.15	60.64	72.38
SOMO	30.90	44.75	29.39	29.28	38.64	55.68
Tense	3.07	48.42	34.65	22.29	41.37	75.58
SubjNum	-19.66	**78.56**	34.48	47.75	64.74	51.50
TreeDepth	4.37	53.03	9.54	46.98	62.79	54.67
One layer per task (§5.4)	36.12	62.66	25.78	49.87	59.79	26.73
Only three features (§5.5)	**41.69**	75.66	47.56	**72.59**	**80.52**	**76.77**
	CoordInv_1	ObjNum_2	TreeDepth_1	SubjNum_1	SubjNum_2	TreeDepth_6
	TreeDepth_1	SubjNum_2	SOMO_4	BShift_ 3	Tense_8	Tense_8
	BShift_12	TreeDepth_12	ObjNum_7	CoordInv_10	CoordInv_9	Tense_12

Table 1: RMSE reduction from baseline. A larger value shows the probing results more indicative of the fine-tuning performance. The **bold-font** configurations are those with the highest RMSE reductions for predicting each fine-tuning task.

	RTE	COLA	MRPC	SST2	QNLI	QQP
bigram shift (BShift)	4,5	2,4,5	2,4,5,9	2,5,6	2,4,5	2,4,5
coordination inversion (CoordInv)	5,6,12	1,2,4,6	1,6	1,4,6	1,4-6	2-4,6
object number (ObjNumber)	1	1,3,8,11	1,3	1,3-5,8,11	1,3,8,11	1-5,12
semantic odd man out (SOMO)	4,5,8,12	2-6	3,4	3,5,6	2-6	2,5-9,12
past present (Tense)	1	1,3,5	1,5,6	1,11	1,3,5,8	1-5,8-11
subject number (SubjNum)	None	1,3-6,9	1	1,4	1	1,2,3,4
tree depth (TreeDepth)	1	1	1	1,3,5	1	1-3,7,8,11

Table 2: Layers with significant probing results ($p < .05$ from one-way ANOVA) with residual dof $= 12$.

Table 3 shows the maximum MSE reductions using different choices of probes. A perhaps surprising finding is that the highest-accuracy probes do not always produce the most valuable results. To predict fine-tuning performances, MLP-20 and RandomForest-100 are instead more recommended.

As a side note, among the 48 results presented in Table 3, only 9 are not achieved by the "best-3-features" methods (including the 2 shown in Table 1). This contrast emphasizes the importance of feature selection when configuring probes.

5.7 Ablation: dataset size

The SentEval probing suite provides more than $100k$ data samples per task but, according to Anonymity (2022), much smaller probing datasets could lead to reliable results. This is supported by the findings in §5.2 - §5.6, where as few as 1,200 samples per class (around 1% of total data) are sufficient to provide useful findings. What if we further reduce the sizes of probing datasets? Here, we repeat §5.2 and §5.5 with probing results from only 400 samples per class. While we can also reduce RMSE with only 400 samples per class, probing results are generally not as useful as those from 1,200 samples. Among the 48 configurations, the probing results from 400 samples have worse RMSE reductions in 11 configurations, but better in 5. Detailed results are included in Table 5.

5.8 Uncertainty analysis

Our method involves comparing the maximum RMSE reductions against the baseline (regressing from features drawn from Gaussian) RMSE_c, which may be affected by the random seeds. Here we describe an error analysis on the baseline regressor results of §5.2 - §5.5.

We run $N = 100$ Monte Carlo simulations on each configuration of regression from 3, 7, and 12 features, respectively, record the RMSE_c, and analyze the uncertainty. We use the variation of RMSE_c (as measured by $\text{Std}(\text{RMSE}_c)$) relative to the scale (as measured by $\text{Mean}(\text{RMSE}_c)$) to describe the uncertainty. As shown in Table 4, the uncertainty remains relatively stable across the choice

	RTE	COLA	MRPC	SST2	QNLI	QQP
Highest-accuracy probe in §5.2 - §5.5	41.69	78.56	53.18	**72.59**	80.52	76.77
Specify one probing method (§5.6)						
DecisionTree	51.98	68.48	54.31	70.90	74.35	52.85
LogReg	45.28	78.34	44.87	70.26	83.13	73.98
MLP-10	48.50	72.12	45.88	65.87	73.82	81.97
MLP-20	47.37	74.94	**63.79**	69.22	79.10	**82.67**
RandomForest-10	50.64	74.08	50.17	68.2	75.19	59.66
RandomForest-100	**53.94**	**79.20**	53.21	71.60	**83.25**	72.72
SVM	51.71	74.01	57.92	71.44	76.78	73.03

Table 3: Maximum MSE reductions using different probing configurations. The **bold-font** numbers are the maximum values in each column.

	3 features	7 features	12 features
RTE	5.71%	9.00%	15.16%
COLA	5.46%	10.00%	13.60%
MRPC	5.21%	9.47%	15.63%
SST2	5.03%	9.41%	14.70%
QNLI	5.37%	10.01%	14.07%
QQP	5.80%	9.29%	14.97%

Table 4: The relative uncertainties ($\frac{\text{Std}(\text{MSE}_c)}{\text{Mean}(\text{MSE}_c)}$) using 3, 7, and 12 features to regress the 6 fine-tuning task performances.

of regression tasks but increases with the number of features. This result favors the use of fewer probing results as features.

Note that these uncertainty values are nontrivial. Let us take COLA as an example. To regress the fine-tuning performance, a 3-feature setting can achieve 75.66% RMSE reduction compared to RMSE_c, but RMSE_c itself has 5.46% uncertainty. This translates to around 7.22% uncertainty for the RMSE-reduction results (Table 1).

Can we reduce the uncertainty by using alternative evaluation metrics like the RMSE, or the percentage of explained variance (`ExplVar`), instead of introducing a control task? In addition to the adjustment for dataset artifacts, the control task provides a baseline to understand the utility. While the RMSE is always positive and `ExplVar` is almost always above 90%, RMSE reduction itself provides a clearer picture of the utility of probing features.

5.9 Can the probing results distinguish the originating foundation models?

Since the 25 models come from 5 foundation models (RoBERTa, XLM, ALBERT, DeBERTa, XLNet), one may wonder if it is possible to distinguish the originating models from the probing results – if so, this would form a potential confounder.

We use 5-class logistic regression from scikit-learn. For any combination of three features, we compute the accuracy following 5-fold cross validation. On all combinations of 3 features, the probing features can reach 0.0027 accuracy (sd=0.0109) better than the random features. This is statistically significant[10]. However, the maximum reachable accuracy is 0.08, whereas even a trivial predictor always outputting "RoBERTa" has an expected 0.24 accuracy. The small accuracies show that our "model augmentation" procedure (§4.3) produces sufficiently distinct models, and the potential confound can be discarded.

6 Discussion

Probing is computational-friendly Compared to fine-tuning, probing evaluations require less computation. Fine-tuning the 6 GLUE tasks takes around 30 GPU hours in total, while probing the 7 tasks (all 12 layers) takes 0.7 GPU hours to cache and 1.3 CPU hours to probe. We elaborate the computational budgets in Appendix A. Probing is far more efficient because it does not need to change the parameters in the neural model, and we only need one pass through the neural model and cache the representations. Fine-tuning needs the gradients to update the parameters in the neural models. There are some methods to reduce the computation costs[11], but, unfortunately, fine-tuning can not be

[10] One-sample one-sided t-test on dof $= 571,703$.

[11] One approach involves using momentum to accelerate the convergence (Kingma and Ba, 2015; Dozat, 2016). Alternatively, the memory usage can be reduced (Gomez et al., 2017; Behrmann et al., 2019). Empirically, limiting the precisions can also accelerate optimization (Shin et al., 2021). Prefix

	RTE	COLA	MRPC	SST2	QNLI	QQP
All layers one task						
BShift	21.00	53.66	19.65	35.60	51.32	57.55
CoordInv	4.49	31.28	22.83	38.93	6.30	25.51
ObjNum	29.51	56.10	39.94	65.95	72.30	70.54
SOMO	-5.09	-7.14	16.09	9.36	-0.71	46.36
Tense	1.30	51.79	33.63	13.52	49.43	73.01
SubjNum	9.89	76.32	47.19	48.62	65.36	49.42
TreeDepth	-11.49	66.06	28.98	27.13	59.93	43.10
Only three features	47.38	77.84	56.70	72.27	82.01	71.08
	Tense_1	SubjNum_1	BShift_2	SubjNum_1	SubjNum_1	Tense_3
	SubjNum_11	BShift_6	ObjNum_7	Tense_2	SubjNum_8	BShift_4
	CoordInv_12	TreeDepth_8	SOMO_9	CoordInv_6	BShift_9	BShift_8

Table 5: RMSE reduction from baseline, using probing results with 400 data samples per class. The colored results are different from (better than or worse than) the results with 1,200 data samples (Table 1) by more than the estimated uncertainty margins in §5.8, i.e., 5% and 15% for 3 and 12 features, respectively.

accelerated by caching to such an extent.

Fine-tuning tasks need more specifications Currently, the most popular leaderboards for natural language understanding constitute fine-tuning tasks. A criticism towards these leaderboards is underspecification (D'Amour et al., 2020) – the short descriptions of the tasks can hardly be inclusive enough to specify the precise abilities required to complete the tasks. To further understand the underspecification problem, researchers recently developed probing datasets (McCoy et al., 2019; Warstadt et al., 2020). Probing results on these datasets have been (indirectly) used in developing deep neural models – performance prediction is a more direct application.

Probing tests can provide fine-grained insights Leaderboard tasks should be customized to the users (Ethayarajh and Jurafsky, 2020). The diversity of probing datasets offers flexible choices to NLP researchers, supporting the diversified considerations to the consumers. Some recently proposed fine-grained leaderboards allow researchers to answer questions like "where does model A outperform model B" (Ma et al., 2021; Narayan et al., 2021; Ruder et al., 2021; Liu et al., 2021a). The probing literature can provide many more datasets for building diverse leaderboards.

Incorporating probing in model developments While probing results are useful for predicting fine-tuning performance, there might be alternative ways to incorporate probing results in model development, with or without using fine-tuning results as proxies. Either way, completing this "feedback loop" can improve neural model development. Considering that probing results come from out-of-domain data, it might be more desirable to use probing results to optimize hyper-parameters instead of parameters themselves. For example, after pre-training a deep neural model for 1000 steps on Wikipedia, is it more beneficial to continue training on Wikipedia or proceed to Common Crawl? The probing results could help answer questions of this kind, which we leave to future work.

7 Conclusion

This paper shows that analyzing probing results can be relevant to developing deep NLP models via predicting a proxy signal, i.e., fine-tuning performance. We show that as few as three probing accuracy scores can be useful to predict fine-tuning results with RMSEs 40% - 80% smaller than baselines. This can dramatically improve the efficiency of deep learning pipelines. Given several ablation studies, we recommend MLP-20 and RandomForest-100 over other probing methods and show that the probing results from as few as 400 per class may still contain predictability. Probing analysis contain rich resources, and we show their results are closely related to fine-tuning performances. We call for further applications of probing into the developments of deep NLP models.

tuning and prompt tuning are also lightweight alternatives to fine-tuning (He et al., 2022; Le Scao and Rush, 2021; Li and Liang, 2021). Liu et al. (2021b) summarizes many approaches related to prompt.

References

In Anonymity. 2022. On the data requirements of probing.

Stéphane Aroca-Ouellette, Cory Paik, Alessandro Roncone, and Katharina Kann. 2021. PROST: Physical reasoning about objects through space and time. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4597–4608, Online. Association for Computational Linguistics.

Marco Baroni. 2021. On the proper role of linguistically-oriented deep net analysis in linguistic theorizing. *ArXiv:2106.08694*.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Joern-Henrik Jacobsen. 2019. Invertible Residual Networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 573–582. PMLR.

Yonatan Belinkov. 2021. Probing Classifiers: Promises, Shortcomings, and Alternatives. *arXiv:2102.12452*.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An Evaluation Toolkit for Universal Sentence Representations. In *LREC*. ArXiv: 1803.05449.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. 2020. Underspecification presents challenges for credibility in modern machine learning. *arXiv preprint arXiv:2011.03395*.

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, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. 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 2185–2194, Hong Kong, China. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Timothy Dozat. 2016. Incorporating Nesterov momentum into Adam. *OpenReview*.

Nadir Durrani, Hassan Sajjad, and Fahim Dalvi. 2021. How transfer learning impacts linguistic knowledge in deep NLP models? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4947–4957, Online. Association for Computational Linguistics.

J. Ebrahimi, Hao Yang, and Wei Zhang. 2021. How Does Adversarial Fine-Tuning Benefit BERT? *ArXiv*, abs/2108.13602.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic Probing: Behavioral Explanation with Amnesic Counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

Kawin Ethayarajh and Dan Jurafsky. 2020. Utility is in the eye of the user: A critique of NLP leaderboards. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4846–4853, Online. Association for Computational Linguistics.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural language models as psycholinguistic subjects: Representations of syntactic state. 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 32–42, Minneapolis, Minnesota. Association for Computational Linguistics.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. 2017. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30.

Ishaan Gulrajani and David Lopez-Paz. 2020. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*.

Rowan Hall Maudslay, Josef Valvoda, Tiago Pimentel, Adina Williams, and Ryan Cotterell. 2020. A Tale of a Probe and a Parser. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7389–7395, Online. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a Unified View of Parameter-Efficient Transfer Learning. In *International Conference on Learning Representations*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. In *International Conference on Learning Representations*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word 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 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Nora Hollenstein and Lisa Beinborn. 2021. Relative importance in sentence processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 141–150, Online. Association for Computational Linguistics.

Alexander Immer, Lucas Torroba Hennigen, Vincent Fortuin, and Ryan Cotterell. 2021. Probing as quantifying the inductive bias of pre-trained representations.

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.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*. ArXiv:1412.6980.

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 *International Conference on Learning Representations*.

Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

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.

Da Li, Henry Gouk, and Timothy Hospedales. 2022. Finding lost dg: Explaining domain generalization via model complexity. *arXiv preprint arXiv:2202.00563*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, Online. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. 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.

Pengfei Liu, Jinlan Fu, Yang Xiao, Weizhe Yuan, Shuaichen Chang, Junqi Dai, Yixin Liu, Zihuiwen Ye, and Graham Neubig. 2021a. ExplainaBoard: An explainable leaderboard for NLP. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 280–289, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021b. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692*. ArXiv: 1907.11692.

Charles Lovering, Rohan Jha, Tal Linzen, and Ellie Pavlick. 2021. Predicting Inductive Biases of Pre-Trained Models. In *ICLR*.

Zhiyi Ma, Kawin Ethayarajh, Tristan Thrush, Somya Jain, Ledell Wu, Robin Jia, Christopher Potts, Adina Williams, and Douwe Kiela. 2021. Dynaboard: An Evaluation-As-A-Service Platform for Holistic Next-Generation Benchmarking. In *Conference on Neural Information Processing Systems*.

Christopher D. Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Alessio Miaschi, Dominique Brunato, Felice Dell'Orletta, and Giulia Venturi. 2020. Linguistic Profiling of a Neural Language Model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 745–756, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Stanislav Minsker and Timothée Mathieu. 2019. Excess risk bounds in robust empirical risk minimization. *arXiv preprint arXiv:1910.07485*.

Marius Mosbach, Anna Khokhlova, Michael A. Hedderich, and Dietrich Klakow. 2020. On the Interplay Between Fine-tuning and Sentence-level Probing for Linguistic Knowledge in Pre-trained Transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2502–2516, Online. Association for Computational Linguistics.

Avanika Narayan, Piero Molino, Karan Goel, Willie Neiswanger, and Christopher Ré. 2021. Personalized Benchmarking with the Ludwig Benchmarking Toolkit. In *Conference on Neural Information Processing Systems*.

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 *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Ellie Pavlick. 2022. Semantic Structure in Deep Learning. *Annual Review of Linguistics*, 8. Publisher: Annual Reviews.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, page 6.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Tiago Pimentel and Ryan Cotterell. 2021. A Bayesian framework for information-theoretic probing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2869–2887, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Sebastian Ruder, Noah Constant, Jan Botha, Aditya Siddhant, Orhan Firat, Jinlan Fu, Pengfei Liu, Junjie Hu, Dan Garrette, Graham Neubig, and Melvin Johnson. 2021. XTREME-R: Towards More Challenging and Nuanced Multilingual Evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10215–10245, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dongyeob Shin, Geonho Kim, Joongho Jo, and Jongsun Park. 2021. Low complexity gradient computation techniques to accelerate deep neural network training. *IEEE Transactions on Neural Networks and Learning Systems*.

Aviv Slobodkin, Leshem Choshen, and Omri Abend. 2021. Mediators in determining what processing BERT performs first. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 86–93, Online. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Anirudh Srinivasan, Sunayana Sitaram, Tanuja Ganu, Sandipan Dandapat, Kalika Bali, and Monojit Choudhury. 2021. Predicting the Performance of Multilingual NLP Models.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. 2021. ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Elena Voita and Ivan Titov. 2020. Information-Theoretic Probing with Minimum Description Length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*.

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Wenjun Zeng, and Tao Qin. 2021a. Generalizing to unseen domains: A survey on domain generalization. *arXiv preprint arXiv:2103.03097*.

Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021b. Towards zero-label language learning. *arXiv preprint arXiv:2109.09193*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Alex Warstadt, Yian Zhang, Xiaocheng Li, Haokun Liu, and Samuel R. Bowman. 2020. Learning which features matter: RoBERTa acquires a preference for linguistic generalizations (eventually). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 217–235, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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.

Mengzhou Xia, Antonios Anastasopoulos, Ruochen Xu, Yiming Yang, and Graham Neubig. 2020. Predicting performance for natural language processing tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8625–8646, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zihuiwen Ye, Pengfei Liu, Jinlan Fu, and Graham Neubig. 2021. Towards more fine-grained and reliable NLP performance prediction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3703–3714, Online. Association for Computational Linguistics.

Lang Yu and Allyson Ettinger. 2021. On the Interplay Between Fine-tuning and Composition in Transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2279–2293, Online. Association for Computational Linguistics.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting Few-sample BERT Fine-tuning. In *International Conference on Learning Representations*.

Yichu Zhou and Vivek Srikumar. 2021. A Closer Look at How Fine-tuning Changes BERT.

A Computational budget

Computation budget for fine-tuning The computation budget for fine-tuning varies for the tasks (primarily due to the sizes of the datasets). The time usages for different foundational models do not differ much – around 16 minutes for RTE, 17 minutes for MRPC, 45 minutes for COLA, 6.5 hours for SST2, 7 hours for QNLI, 18 hours for QQP. If we normalize by the sizes of the datasets, fine-tuning takes between 2 and 6 minutes of GPU time per 1,000 data samples.

Computation budget for probing Before probing, we cache the representations of SentEval data (taking around 60 hours on a GPU card) to avoid the feed-forward pass, which is the most time-consuming part. Note that we cached the whole SentEval data and then subsample 1,200 per class (around 1%). Caching only the subsampled 1% would take 40 minutes on GPU.

The probing classifications take around 16 hours of CPU time for each of the 25 models. This includes 7 (probing tasks) $\times$ 12 (layers) $\times$ 7 (probing methods) = 588 (probing configurations).

Consider a configuration of one probing method, 12 layers, and 7 probing tasks. Acquiring 84 probing features would take 40 minutes (caching) + 80 minutes (probing), which is about two hours, where only 1/3 of the two hours need GPU.

Computation in analysis For §5.5, the feature selection procedure takes between 350 and 450 seconds for one fine-tuning task when running in RStudio on a laptop (with i7 CPU). All 6 fine-tuning tasks take around 1 hour.

For §5.6, the computation cost is 7 times that of §5.2 to §5.5, totaling around 7 hours.

For §5.7, the probing time is around 15 hours. Subsequent analysis time on RStudio equals §5.2 to §5.5, i.e., around 1 hour.

For §5.8, a Monte Carlo simulation for the uncertainty analysis takes around 15 minutes for all 6 fine-tuning tasks on a laptop CPU.

For §5.9, running through the features to find the top 3 for distinguishing the foundation models take 50 minutes on a desktop CPU.