LIFT: Language-Interfaced FineTuning for Non-Language Machine Learning Tasks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Finetuning pretrained language models (LMs) has become a norm for learning various downstream tasks. While it is feasible to finetune LMs for language downstream tasks without making any architectural changes, most existing finetuning approaches for non-language tasks rely on task-specific designs for input, output layers, and loss functions. A natural question arises – Can language model finetuning solve non-language downstream tasks without changing models' architecture or loss function? To answer this question, we study the efficacy and limitations of **Language-Interfaced FineTuning (LIFT)** for non-language tasks by conducting an extensive empirical study on a suite of non-language classification and regression tasks. LIFT does not make *any* changes to the model architecture or loss function, and it solely relies on the *natural language interface*, truly enabling "no-code machine learning with language models". We find that LIFT performs relatively well across a wide range of low-dimensional classification and regression tasks, matching the performances of the best models in many cases, especially for the classification tasks. We thoroughly study fundamental properties of LIFT, including the inductive bias, sample efficiency, ability to extrapolate, robustness to noises and corrupted labels, and adversarial robustness. We also analyze a few unique properties specific to LIFT – non-deterministic predictions and how to use them, and sample-efficient context-aware learning via appropriate prompting or two-stage finetuning. We provide discussions on limitations and open questions toward making LIFT more effective and efficient.

1 Introduction

Deep neural networks have accomplished great success in multiple domains, from computer vision [1, 2], natural language processing [3, 4] to game playing [5, 6]. Most advancements in deep learning have come with a variety of domain-specific designs for network architectures, such as convolutional neural networks [7, 8, 9] for vision tasks or recurrent neural networks [10, 11] and transformers [12, 13] in the language domains. A domain-and-modality agnostic model that can be adapted to solve tasks across modalities or domains has become a desideratum [14], motivating great efforts in transfer learning [15] and multi-modal learning [16]. Recently, transformer-based language models (LMs) [13, 17, 18, 19] shed light on enabling the universal model with their incredible versatility across different domains and modalities. They adapt impressively well to not only language domains [20], such as sentiment analysis [21, 22], question answering [23], or commonsense reasoning [24], but also non-language modalities [18]. For instance, GPT2 [17] pretrained on language data can be efficiently finetuned to perform image classification and numerical computation [18].

The adaptation of pretrained LMs to downstream language tasks can be achieved without modifying the models' architecture via finetuning [25, 26, 27, 28] or in-context few-shot learning methods [29, 30]. However, this may not be the case for non-language tasks. As their input and output formats are not necessarily in linguistic forms, adapting LMs to these domains usually requires architectural changes, especially in the designs of the input layer, output layer, and loss functions. For instance, to adapt GPT2 [23] to other modalities, the frozen pretrained transformer [18] adds an input layer and

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

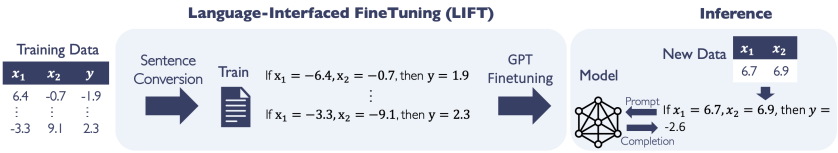

Figure 1: A high-level illustration of Language-Interfaced FineTuning (LIFT). The LIFT framework consists of two phases: dataset conversion and language model finetuning. To obtain a model for a given task, all you need with LIFT is designing proper sentence templates and converting the labeled data to a sentence dataset. No changes to architecture or loss functions are needed.

an output layer for learning the input and output transformations before finetuning. Such changes require a good understanding of the underlying principles of LMs and some modifications to the program code. A natural question that arises is whether such changes are necessary. In other words,

Can LM finetuning solve non-language tasks
***without** changing the architecture or the loss function?*

To find the answer to this question, we consider a simple finetuning procedure for pretrained LMs, dubbed **Language-Interfaced FineTuning (LIFT)**. The LIFT framework can be used to learn a predictor for any classification or regression tasks. LIFT consists of two phases: (1) converting labeled samples into sentences and (2) finetuning pretrained LMs on the sentence dataset with the same decoder architecture and the language modeling loss. Fig. 1 illustrates how we finetune GPT with LIFT to solve a regression task. Here, input data has two features $\mathtt{x}_1, \mathtt{x}_2$ and the goal is to predict $\mathtt{y}$. LIFT first converts each labeled sample to a sentence. In this particular example, we use a specific template "`If` $\mathtt{x}_1 = \text{value}_1, \mathtt{x}_2 = \text{value}_2$, `then` $\mathtt{y} = \text{value}_3$" for sentence as shown in the figure. We then finetune LMs without changing neither architecture nor loss function. Once the model is finetuned, we perform inference as follows. It first converts test samples into sentences using the same template while leaving the $\mathtt{y}$ value empty and then feeds the converted sentences as prompts to the finetuned model. The output tokens are parsed to provide our prediction.

In this paper, we empirically show that LIFT can solve a variety of non-language tasks. Fig. 2 visualizes examples of numeric functions learned by GPT-J models finetuned with our LIFT framework. Since LIFT does not require any changes in the architecture or loss function, this implies that changes to architecture or loss function are not necessary even when finetuning LMs on non-language tasks. Thus, LIFT can be almost viewed as a 'no-code machine learning' framework as the data-to-sentence conversion is extremely straightforward even for people without programming backgrounds.

More specifically, we investigate the efficacy and limitations of LIFT on non-language tasks by conducting an extensive empirical study on a suite of non-language classification and regression tasks. *First*, LIFT is observed to perform well across a wide range of low-dimensional classification and regression tasks. Though the performance of LIFT is not the best overall, it almost matches the performance of the best baseline models in most cases. *Secondly*, for a better understanding of LIFT, we thoroughly analyze its fundamental learning properties, including inductive bias, sample efficiency, ability to extrapolate, noise robustness, and adversarial robustness. LIFT is observed to be robust to outliers and label corruption. *Thirdly*, we also study a few unique properties specific to LIFT – non-deterministic predictions and how we can use them, and sample-efficient context-aware learning via appropriate prompting. LIFT is observed to behave similarly to the Bayesian prediction and utilize feature names for faster adaptation to the tasks. *Lastly*, improving upon the basic finetuning, we employ more techniques for more tasks: two-stage intermediate finetuning with synthetic pretext tasks and data augmentation with mixup. We find that two-stage finetuning improves the performance of LIFT in the regime of low training samples. We discuss the current limitations of LIFT and future investigations towards making LIFT more effective and efficient.

Scope of study The goal of this work is to understand: (i) what LIFT can and cannot, (ii) properties of finetuned models obtained via LIFT, and (iii) several techniques to improve LIFT. We emphasize that LIFT is not meant to achieve impressive performance compared to the baseline algorithms.

2 Related Work

Pretraining and finetuning large-scale language models Large-scale language models renovate and sharpen the field of natural language processing [4]. Starting from BERT [31], pretraining LMs and finetuning on various downstream tasks has become the standard practice, achieving astonishing performances across multiple tasks [32, 33, 21, 23, 34, 35]. Finetuning LMs also benefit downstream tasks across different domains [36]. So far, finetuning LMs is limited to NLP tasks. In this work, we explore whether finetuning LMs can be beneficial for non-language tasks across different modalities. Finetuning methods for LMs have been widely studied, ranging from simple architecture modifications: adding linear layers [37, 38], and freezing parts of the network [18, 39, 40];

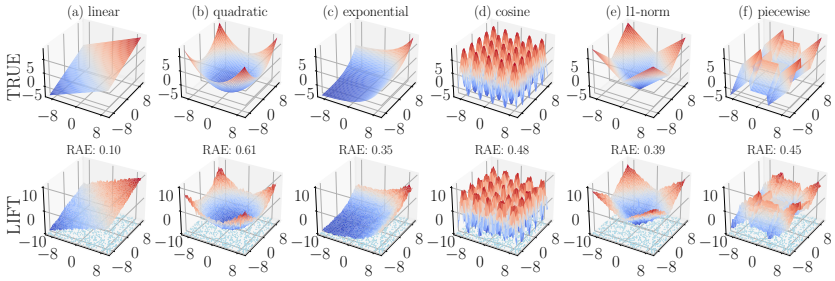

Figure 2: Performance of GPT-J (finetuned with LIFT) in approximating various functions. The first row visualizes the true values of the functions, and the second row visualizes the predicted values of GPT-J after finetuning for the corresponding regression tasks with 1,000 training samples, using suggested LIFT framework. The blanks shown on the surfaces produced by GPT-J represent invalid outputs. The blue points visualize the input vector of the training samples. See Appendix for experiments on smaller number of samples.

to more advanced methods such as adapter networks [26, 27, 28], low-rank adaptation of network parameters [25], and intermediate finetuning [41, 42, 43]. These methods can be adopted for non-language tasks with modifications over the input/output layers and the loss functions. However, these modifications might steer pretrained parameters into unwanted directions, leading to undesired behaviors like catastrophic forgetting. In-context few-shot learning paradigm [19, 44, 29, 30, 45] suggests modifying only LM inputs by adding a few examples of the downstream task. These models work well for NLP downstream tasks, but it is unclear whether the same is applied to downstream tasks of other modalities. Our work studies the feasibility of directly finetune LMs for non-language tasks, further pushing the application boundaries of large-scale language models.

Adaptation of LMs to non-language tasks Transformer-based architectures have been successfully applied to non-language domains and modalities, such as computer vision [46, 2], protein sequences [47], reinforcement learning [48], and tabular data [49, 50, 51]. Multiple works study the transferring of these models for downstream tasks in the same modality, such as language [52], vision [2] or different modalities, such as CLIP for language-image association [53] or between English and programming languages [54]. However, directly finetuning LMs for other modalities is just recently visited. Recent works [18] empirically show that GPT2 [23] pretrained on language tasks and data can be efficiently finetuned for different modalities and domains. By freezing most pretrained parameters except the layer normalization layers and adding input and output layers, they can finetune to achieve competitive performance on various datasets. We note that these works require some changes in the architecture and losses to adapt to different data representations [18], while LIFT requires no such changes. For **tabular data**, to the best of our knowledge, we are one of the first works studying the adaptation of LMs for non-language tasks. We note that though achieving tremendous success with natural data, deep learning still faces difficulties with standard machine learning tasks, especially on tabular data, probably because of its heterogeneous nature with features being dense numerical and weaker in correlation compared to natural image-language data [49]. Furthermore, transformer-based language models are observed not to work well on complex numeric tasks [55, 56, 57] and are sensitive to different formats of numeracy [55]. These transformer-based models for tabular data [50, 51] have different architecture designs from the LMs. Nevertheless, standard learning algorithms (Random Forest [58] or XGBoost [59]) are found to still outperform deep learning models on various tabular datasets [49].

3 Methodology and Experimental Setup

We describe the methodology used in our empirical study. Our code is publicly available here.[1]

LIFT To finetune a pretrained LM with LIFT on a supervised dataset, we apply two steps: (1) convert each sample into a sentence with a fixed template, (2) perform standard finetuning on LMs with sentence datasets. We use the default cross-entropy loss for token prediction in LMs. Our template is "`When we have` $\mathbf{x}_1 = v_1, \ldots, \mathbf{x}_p = v_p$, `what should be the` $\mathbf{y}$`?### y =`$v_{\mathbf{y}}$`@@@`", where $v_1, \ldots, v_p$ are feature values and $v_{\mathbf{y}}$ is the response value. For the testing sample, we leave the value $v_{\mathbf{y}}$ blank. The predicted output is parsed from the output tokens generated from finetuned LMs.

Pretrained language models We apply LIFT on two pretrained language models: GPT-J [60] and GPT-3 [19]. For GPT-J, we apply Lora [25] for efficient finetuning that essentially approximates the weight update of each layer by a low-rank matrix. For experiments on GPT-J, we used `p3.8xlarge` and `p3.2xlarge` instances from AWS and RTX3090 GPUs in the local server. Since GPT-3 is not fully publicly available, we use the API provided by OpenAI to perform black-box GPT-3 finetuning.

[1] `https://anonymous.4open.science/r/LanguageInterfacedFineTuning`

Property	Conclusion
Overall performance	For classification tasks, LIFT can achieve comparable performances to strong baselines (see Table 2) on various datasets. For regression tasks, LIFT can approximate different types of functions in the low-dimensional cases (see Fig. 2), but does not work well in the high-dimensional cases (see Appendix).
Robustness	In general, LIFT does not show clear robustness over label corruption, adversarial examples, etc. For regression tasks, LIFT is robust to outliers (Fig. 23).
Context-aware learning	Performance of LIFT can be further improved by providing the context (feature names) of tasks. The improvement is more significant on the datasets in which common knowledge helps. (Tables 19, 5)
Intermediate finetuning	When applying synthetic pretext tasks with LIFT, the number of required samples for achieving high performance is reduced in both classification and regression tasks (Fig. 7).

Table 1: Summary of main findings on LIFT.

Evaluation We evaluate the performance of LIFT on various non-language classification/regression tasks. First, we compare the *overall performance* of LIFT with popular machine learning algorithms. Second, we evaluate the *robustness* of LIFT under the corruption of labels, outliers, and adversarial examples. Finally, we design experiments to investigate the unique properties of LIFT. Note that GPT may produce *invalid outputs*, such as non-numerical outputs. For classification, we consider invalid outputs as misclassifications; for regression, we increase the generation randomness and regenerate five times to obtain valid outputs. For the cases in that valid outputs are not available, we measure the regression error by replacing the invalid outputs with the mean of the observed outcome of the training dataset. In terms of *evaluation metrics*, we measure accuracy on classification tasks, and RMSE, RAE errors on regression tasks, where RAE $= \frac{\sum_{i=1}^{n} |\hat{\mathrm{y}}_i - \mathrm{y}_i|}{\sum_{i=1}^{n} |\bar{\mathrm{y}} - \mathrm{y}_i|}$ and RMSE $= \sqrt{\frac{\sum_{i=1}^{n} (\hat{\mathrm{y}}_i - \mathrm{y}_i)^2}{n}}$ on each dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{n}$ with n samples, features $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$, and outcome y.

Datasets We design and select a wide range of datasets to better understand the behavior of LIFT. For *classification*, we use three types of non-language data: low-dimensional synthetic datasets, real tabular datasets in OpenML [61] and vision datasets (MNIST [62], Fashion-MNIST [63] and their permuted variant [64]). For *regression*, we use both synthetic and real datasets. For synthetic datasets, we defined samples $(\mathbf{x}_i, \mathrm{y}_i)$ of input-output pair as $\mathrm{y}_i \sim f(\mathbf{x}_i) + \mathcal{N}(0, \sigma^2)$, where $\sigma^2 \geq 0$ is the noise level. Unless otherwise stated, we sample the feature $\mathbf{x}_i$ uniformly from a hypercube $[L, U]^p$, where L and U are minimum/maximum feature values, and p is the number of features. Following the suggestion by [65], we consider various functions f for regression tasks: (i) linear function, (ii) quadratic function, (iii) exponential function, (iv) cosine function, (v) $\ell 1$-norm function, and (vi) piece-wise function. We also use three real datasets: Medical Insurance [66], Combined Cycle Power Plant (CCPP) [67], Servo [68] and student [69]. More details are included in Appendix.

Baselines We consider standard machine learning algorithms [70, 71]. The classification baselines are logistic regression (*LogReg*), decision tree (*DT*), k-nearest neighbor (*KNN*), support vector machine with Gaussian kernel (*SVM*), random forest (*RF*), and XGBoost (*XG*). We also compared with an optimal deterministic classifier (*ODC*) that outputs the most dominant class. For regression, we use polynomial regression (*PR*), kernel ridge regression (*KR*) with radial basis function kernel, k-nearest neighbors (*KNN*), a three-layer ReLU neural network (*MLP*) with 50 hidden neurons at each layer, Gradient Boosting Trees (*GBT*), random forest (*RF*), and Gaussian process (*GP*).

4 Basic Findings of LIFT

We study LIFT on standard non-language classification and regression tasks. Table 1 summarizes our main findings on LIFT in terms of classification/regression performance, robustness of LIFT against corruption/outlier/attack, as well as the effect of context-aware learning and pretext task on LIFT.

4.1 How well does LIFT perform on standard ML tasks?

Classification Table 2 compares classification accuracies between algorithms on a wide range of tasks. We observe that LIFT can perform comparably well across most datasets compared to the best performing baseline algorithms (XGBoost, Random Forest, and MLP). In most cases, LIFT/GPT ranks highly in the top three best-performing methods. We find that LIFT can learn non-linear relationships between features and the responses: LIFT achieves 80.17% accuracy on the circle dataset while logistic regression failed to perform better than the ODC (50%). As the difficulty of tasks varies, which can be estimated by the average performance of baselines, LIFT also suffers from the performance degradation. LIFT can perform relatively well even when the number of features is as large as hundreds, though the limited number of tokens as inputs to LMs restricts the number of features LIFT can input. However, when the number of classes is large (say 100s), both LIFT/GPT-J and LIFT/GPT-3 have lower accuracies than many baselines, even they manage to be better than ODC. For instance, the accuracy gap between LIFT/GPT-J and the best algorithm, RBF-SVM, on

Type	Dataset (ID)	#F/#C	ODC	LogReg	KNN	DT	MLP	RBF-SVM	RF	XG	LIFT/GPT-J	LIFT/GPT-3
Synthetic	circles (3)	2 / 2	50.00	48.58±1.94	81.25±0.20	77.42±0.24	82.00±0.54	**83.08±0.59**	82.42±1.33	81.42±0.31	74.92±0.62	81.17±0.42
	two circles(6)	2 / 2	50.00	49.83±4.18	81.83±0.62	75.50±0.20	68.42±3.86	80.00±0.54	76.08±0.59	79.25±0.35	75.92±1.65	**99.00±0.00**
	blobs(2)	2 / 4	25.00	96.75±0.00	95.50±0.20	96.08±0.82	96.58±0.42	96.75±0.00	**97.17±0.24**	96.17±0.12	96.17±0.59	96.67±0.24
	moons(4)	2 / 4	50.00	88.58±0.12	**100.00±0.00**	99.25±0.41	98.75±1.08	**100.00±0.00**	99.75±0.00	99.83±0.12	99.58±0.42	**100.00±0.00**
	9Clusters(1)	2 / 9	11.25	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	**100.00±0.00**	**100.00±0.00**	99.75±0.00	**100.00±0.00**
Tabular (OpenML)	1511	9 / 2	68.18	87.12±0.54	**88.64±0.00**	85.98±0.53	86.36±1.86	86.36±0.00	85.23±0.00	85.23±0.00	85.23±1.61	84.85±1.42
	882	16 / 2	50.00	58.33±11.79	66.67±6.81	**77.78±3.93**	66.67±0.00	58.33±6.81	**77.78±3.93**	63.89±7.86	41.67±11.79	50.00±0.00
	44	58 / 2	60.59	93.27±0.00	90.77±0.00	90.7±0.14	**94.35±0.00**	93.70±0.00	95.01±0.00	**95.87±0.00**	94.03±0.54	94.90±0.36
	1479	101 / 2	49.79	77.78±0.00	56.38±0.00	56.38±0.89	50.21±0.00	68.72±0.00	51.44±0.00	59.26±0.00	**100.00±0.2**	**99.73±0.19**
	48	6 / 3	35.48	45.16±4.56	60.22±4.02	65.59±5.49	54.84±2.63	53.76±6.63	**67.74±7.90**	66.67±8.05	61.29±4.56	55.91±4.02
	23	10 / 3	42.71	49.49±0.83	50.85±1.91	56.72±0.32	**57.29±0.73**	56.50±0.97	53.45±1.05	52.43±0.42	49.83±0.28	**57.74±0.89**
	187	14 / 3	38.89	**100.00±0.00**	96.29±1.31	93.52±2.62	98.15±2.62	**100.00±0.00**	**100.00±0.00**	97.22±0.00	90.74±3.46	91.67±0.00
	54	19 / 4	25.88	80.39±1.00	69.61±0.74	63.92±2.37	79.21±0.28	**81.18±0.48**	75.88±1.27	73.14±0.28	64.31±2.37	70.20±2.73
	40496	8 / 10	11.00	68.67±0.94	63.67±6.13	66.33±2.87	**72.00±0.82**	68.00±0.82	64.33±0.94	66.00±0.82	67.33±4.25	69.33±2.05
	28	65 / 10	10.14	96.53±0.22	96.92±0.16	89.8±1.09	97.36±0.27	97.95±0.00	97.69±0.14	97.48±0.17	98.22±0.11	**98.99±0.30**
	12	217 / 10	10.00	97.67±0.12	97.67±0.31	87.67±1.05	96.5±0.35	**98.83±0.24**	97.75±0.35	96.75±0.00	94.17±1.75	93.08±0.24
	1491	65 / 100	0.94	81.35±0.15	77.60±0.97	43.86±1.21	77.71±1.91	**81.98±0.30**	77.71±1.98	70.21±0.29	50.23±1.33	53.96±1.15
	1493	65 / 100	0.94	81.67±0.97	80.62±0.76	46.88±1.93	76.88±2.44	**83.44±0.89**	73.12±0.76	70.73±1.41	50.32±2.18	55.73±3.19
Images	MNIST	784 / 10	11.35	91.46±0.00	96.78±0.00	86.97±0.11	97.41±0.00	97.91±0.00	95.03±0.00	97.66±0.00	96.88±2.65	**98.15±0.67**
	P-MNIST		11.35	92.60±0.00	96.79±0.00	87.38±0.11	97.49±0.00	**97.84±0.00**	94.71±0.00	97.68±0.00	95.80± 0.07	95.85±0.00
	P-FMNIST		10.00	84.35±0.00	84.72±0.00	79.25±0.12	88.43±0.00	88.04±0.00	84.51±0.00	**89.83±0.00**	87.20±5.66	88.02±1.46

Table 2: Classification accuracies (%) of evaluated algorithms. We evaluate on 2D synthetic data, tabular datasets in OpenML [61] , and image datasets, varying number of features (#F) and number of classes (#C). The second column contains dataset name for synthetic/image datasets, and OpenML ID for tabular datasets. Overall, LIFT performs comparably well across all tasks. LIFT can be adapted well to non-linear datasets (circles, two circles), beyond the capacity of logistic regression. On OpenML data, LIFT achieves competitive performances to the best methods, such as XGBoost or RBF-SVM. However, LIFT performs worse given larger number of classes (100s). On vision data, LIFT performs relatively well, comparable to the best algorithms.

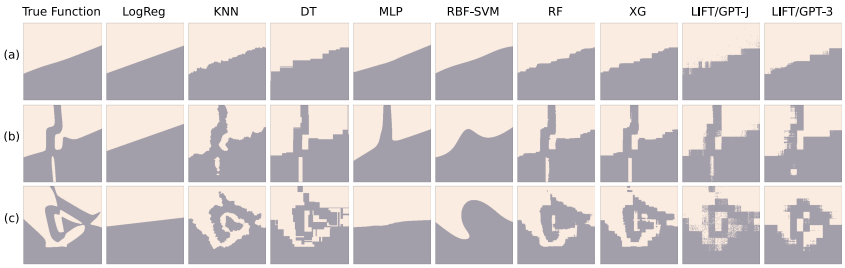

Figure 3: Visualizing the decision boundary of algorithms on three synthetic datasets shown at the first column of each row. These datasets are generated by training a neural network: we take three snapshots of the decision boundary of a trained neural network and define a dataset having label as the prediction of the neural network. From the top to the bottom, we choose latter snapshots, which gives us the dataset that is more difficult to classify. We find that GPTs finetuned by LIFT adapts well on different boundaries.

100-class dataset (OpenML ID=1491) is nearly 30%. We note that most baseline algorithms require feature scaling and normalization for good performance, while LIFT can directly use raw values.

Regression As shown in Fig. 2, LIFT works well for regression tasks. See the Appendix for the comparison of *function approximation* errors. For the low-dimensional cases, LIFT achieves comparable performance to baselines but fails to beat the strongest baselines such as GP. This is as expected since GPT models measure the error by comparing tokens instead of measuring how close the prediction values are to true values. We conjecture that the regression performance can be further improved by level encoding, *that is*, representing the numerical values as binary values. In this way, the distance between the prediction and the true values can be captured by the loss function of GPT. However, as we mentioned in Sec. 3, developing approaches to further strengthen the performance of LIFT is out of the scope of our work. We leave this as one of the interesting future directions.

We also investigate the *interpolation and extrapolation* of LIFT, the results of which are in Appendix. All methods fail to extrapolate and interpolate well for all functions. It turns out that LIFT is not having good interpolation performance except in the linear regression case. An interesting observation we have made is that LIFT tends to output seen values (from training data) for extrapolation.

4.2 Can we predict the inductive biases of language models?

We investigate the inductive biases of models via their decision boundaries. In doing so, we train a simple neural network to construct a set of decision boundaries with different levels of complexity, shown in the first column of Fig. 3 and observe how LIFT behaves for different datasets. Fig. 3 visualizes decision boundaries learned by different algorithms. Both GPT-J and GPT-3 models finetuned with LIFT are able to adapt to different boundaries. Interestingly, their boundary shapes are axis-parallel, similar to those of tree-based classifiers. We conjecture that this phenomenon may come from the encoding and tokenization schemes of GPT for numeric values. In particular, GPT may decide the output token by sequentially considering each set of digits in the numbers. We also observe the fragmentation behavior of decision boundaries of GPT, similar to the behavior

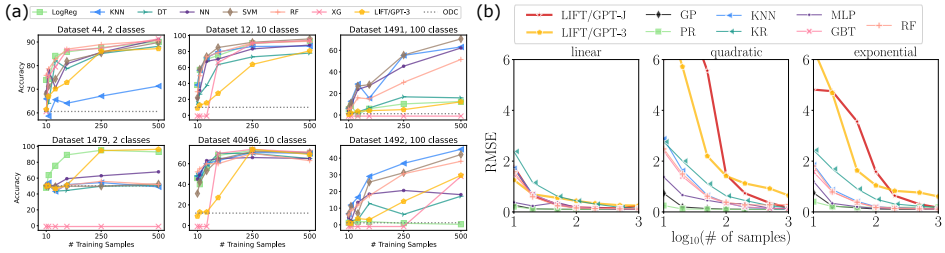

Figure 4: Sample complexity of algorithms on classification and regression tasks. *(a) For classification,* LIFT/GPT-3 achieves comparably good sample complexity among algorithms. When the number of classes increases, LIFT needs more samples to start adapting to the task. *(a) For regression (function approximation),* LIFT/GPT needs slightly larger sample complexity to achieve similar performance. One can notice that LIFT/GPT achieves very competitive or even better performance when around 1000s of samples are given.

of convolution neural networks [72]. See Appendix for more detailed measurements that quantify the similarity between decision boundaries. Our results indicate that finetuning GPT using LIFT provably extracts useful features from labeled datasets to adapt to different decision boundaries, which is partly consistent with recent findings on inductive biases of language models [73].

4.3 How many samples does LIFT need?

We investigate if LIFT is sample efficient. Fig. 4(a) and Fig. 4(b) show results on classification and regression tasks, respectively. We find that the GPT model can be quickly finetuned to new tasks when LIFT is applied. For *classification*, when the number of classes increases, LIFT does need more samples for adaptation. This is probably because GPT requires more samples to learn the more complex input and output spaces of the data. For *regression*, we find that $n = 1000$ samples are sufficient for LIFT to have small RMSE, similar to other baselines. There exist some functions (quadratic, cosine, piece-wise) where LIFT has lower sample complexity than popular baselines.

Finetuning vs. in-context learning Compared to in-context learning, finetuning is a more appropriate solution when the unseen task requires large training data due to the limited context length of LMs. However, it is unclear if this holds given smaller numbers of samples. We investigate this question by comparing LIFT with in-context learning on various data (see Table 10 in the Appendix). In most cases, LIFT/GPT outperforms in-context learning using the same subsets of training data.

4.4 How robust is LIFT?

We investigate to which extent LIFT is robust to practical settings of training data: label corruption, noisy features, outlier samples, adversarial examples, and class imbalance.

Robustness to label corruption We randomly corrupt a percentage of labels in the training data by assigning a new label under two different regimes. For *random errors,* we assign a new label in the label space with equal probability. For *systematic errors,* we assign all errors in a specific class to a singular other class as described in [74]. See Fig. 5(c) for results. We find that LIFT/GPT-3 follows the general trend of the baselines, not particularly outperforming or underperforming in terms of robustness, while LIFT/GPT-J is more sensitive to the corruption, which we also observe on KNN. We suspect that the current finetuning scheme of GPT-J is more sensitive than the black-box finetuning implementation of GPT-3. Nevertheless, we attribute the robust behavior of LIFT/GPT-3 to the ensemble effect of the underlying language models. See the Appendix for detailed results and visualization of the learned decision boundaries.

Robustness to outliers We investigate the robustness of LIFT to outliers in regression tasks whose y is not consistent in terms of fitting $(\boldsymbol{X}, \mathbf{y})$. We inject four outliers to the training data of 200 samples. Fig. 5 compares the RAE values of LIFT and baselines with and without outliers when there are 2% outliers in the training set. Due to the space limit, we defer the full version of in Sec. C. We observe that LIFT with both GPT-3 and GPT-J are the most robust among all eight methods. To be more specific, the performance of LIFT is almost not affected, while all the baselines suffer from a huge performance drop. To further study how robust LIFT is to outliers, we evaluate the performance of LIFT with datasets including 1%, 2%, 5%, 10%, and 20% outliers, respectively. We include median-3NN and median-5NN that serve as the optimal robust baselines [75]. Fig. 24 shows that LIFT with GPT-3 is almost as robust as median-3NN and median-5NN, while LIFT with GPT-J is more vulnerable when the percentage of outliers increases.

Robustness to adversarial samples We test how LIFT is robust against adversarial attacks. Since we don't have full access to the GPT-3 model, we tried the transfer attack. In other words, for a given

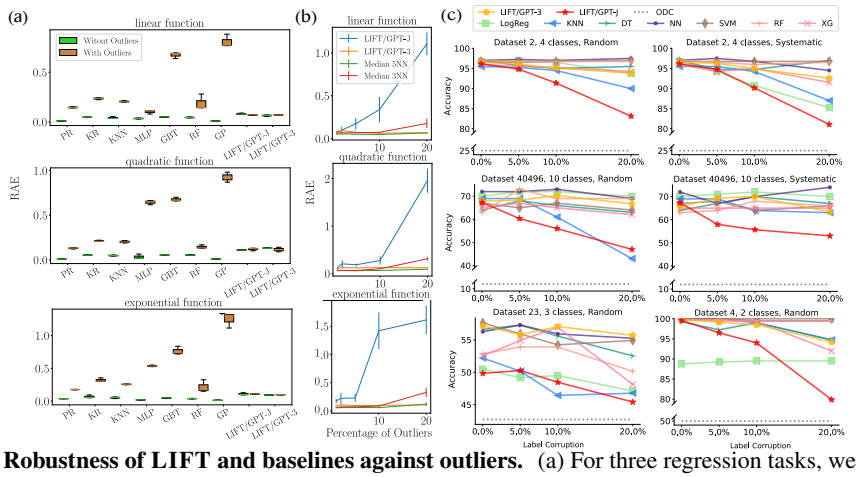

Figure 5: Robustness of LIFT and baselines against outliers. (a) For three regression tasks, we show how the regression error changes as we inject 2% outliers in the dataset. Both GPT-3 and GPT-J, finetuned by LIFT are highly robust against outliers, compared with baselines. (b) Robustness of LIFT and robust baselines against different fraction of outliers. We show the regression error of LIFT when the portion of outliers increases up to 20%. One can observe that LIFT/GPT-3 is comparable to a strong baseline (median NN), while LIFT/GPT-J fails when the portion increases. See Appendix for the comparison on all tested regression tasks. (c) Comparison of LIFT and baselines trained on corrupted synthetic data (ID 2 & 4) and real OpenML (ID 23 & 40496) datasets. We simulate *random errors* and *systematic errors*. While LIFT/GPT-3 shows robustness across the data, LIFT/GPT-J is more sensitive to the corrupted labels, similar to KNN.

real data (x, y), we generated adversarial example $(x + \delta, y)$ for a neural network and tested whether GPT finetuned by LIFT correctly classifies the adversarial examples. See Appendix for the result.

Robustness to class-imbalance We also investigate LIFT on class-imbalanced binary classification (OpenML datasets having IDs 1444, 1467, and 1511). Though evaluated datasets all have high class-imbalance values (ratio of the majority-class-size and the minority-class-size), we find that LIFT can perform well, achieving high F1, precision, and recall scores across the tasks. For instance, on the OpenML-1511 dataset (class-imbalance value is nearly 8), ODC gets 0 scores for both precision and recall while LIFT/GPT-J achieves the best recall (82.61 ± 7.10) and F1 scores (84.43 ± 1.43) along with XGBoost enjoying the highest precision. See Appendix for the detailed result.

4.5 How do different language models affect LIFT?

Tasks	Basic Classification					Classification w/ Feat. Names			In-context Classifction					
DID (#Prompts)	311	1493	1491	48	54	48	23	54	48 (50)	13 (35)	40496 (32)	1511 (28)	54 (42)	893 (13)
GPT-J	93.97±1.00	50.32±2.18	20.23±1.33	61.29±4.56	64.31 ±2.37	58.06±13.94	48.36±0.97	69.02±3.67	34.33±1.47	56.90±19.51	10.00±0.82	56.06±17.14	48.89±3.14	25.49±0.55
Ada	95.39±0.67	55.73±3.19	53.96±1.15	48.39±4.56	70.20±2.73	43.01±1.52	57.48±1.14	72.16±1.20	39.78±10.97	68.97±0.00	15.33±3.86	52.65±12.11	24.31±1.47	55.56±8.32
Babbage	96.81±0.07	62.19±1.80	67.50±3.87	48.39±2.63	72.06±3.82	47.31 ±8.05	58.53±1.05	69.22±1.11	47.31±3.04	71.26±0.81	11.00±0.00	53.79±12.07	24.32±0.56	53.33±5.44
Curie	95.21±0.06	62.50±0.97	61.88±1.48	50.54±8.47	74.27±0.73	49.46±6.08	55.42±0.84	70.66±2.28	32.26±0.00	70.69±0.00	20.67±4.78	67.80±0.53	26.±2.22	53.33±0.00
Davinci	96.81±0.41	57.19±0.70	58.13±2.50	56.99±3.04	71.47±0.88	49.46±9.97	56.31±0.04	68.16±1.69	49.46±4.02	67.82±4.06	20.67±6.60	68.94±0.54	26.28±2.22	55.55±3.14

Table 3: Comparison of LIFT on basic classification tasks, classification tasks with feature names, and in-context classification (note that columns 48(50) etc. indicate DID(#Prompts)). In general, larger LIFT/GPT-3 models (babbage, curie, davinci) perform better than LIFT/GPT-3 ada and LIFT/GPT-J. However, performance on different datasets is not always consistent with model size. Also observed that model size has large effects on in-context classification vs. other tasks.

Together with GPT-J and GPT-3 (Ada version), we now evaluate tasks on larger pretrained LMs to verify whether LIFT benefits more from largerd LMs. We consider three bigger versions of GPT-3, namely Baggage, Curie, and Davinci (in the ascending order of the number of parameters). We compare all models on several regression tasks in Table 4 and classification tasks in Table 3. In general, we find that larger LMs bring some improvements when combined with LIFT.

Function	GPT-J	Ada	Babbage	Curie	Davinci
Linear	0.08±0.01	0.06±0.01	0.06±0.00	0.06±0.01	0.06±0.00
Quadratic	0.11±0.00	0.13±0.00	0.11±0.02	0.10±0.01	0.09±0.00
Exponential	0.11±0.02	0.09±0.00	0.09±0.01	0.08±0.00	0.08±0.00
Cosine	0.38±0.08	0.44±0.10	0.41±0.06	0.38±0.01	0.38±0.05
L1-norm	0.10±0.00	0.09±0.01	0.10±0.01	0.08±0.01	0.09±0.01
Piecewise	0.15±0.01	0.17±0.05	0.15±0.02	0.15±0.01	0.14±0.01

Table 4: Comparison of LIFT on different LMs across regression tasks. The scores are regression errors. In general, Davinci model performs the best, but the gaps to other models are not always significant.

However, these improvements are not quite significant compared to smaller LMs. This may be because the downstream task is small with respect to all evaluated LMs, and finetuning is necessary for LMs to adapt to these new modalities. Furthermore, we observe the consistent improvements of in-context classification with larger models across classification tasks. We conjecture that larger LMs probably contain more embedded knowledge, useful for classification when training data is small. See the Appendix for the detailed result.

5 Evaluation of LIFT-specific learning properties

In this section, we study the behavior of LIFT in a more fine-grained manner.

5.1 Is LIFT doing Bayesian prediction?

We explore how LIFT performs for different levels of noise. We apply LIFT/GPT per each synthetic regression dataset of 1000 training samples. We then perform inferences with varying temperatures 20 times to calculate the standard deviation σ for each prediction. Also, we increase the noise level from 0 to 2 on feature value x with $\sigma(x) = (x + 10)/10$ to show how the noise level influences the model's prediction behavior. Fig. 6 shows our results on six synthetic datasets. We observe that the standard deviation of LIFT's prediction aligns well with that of the observations across six different tasks. These results imply that LIFT/GPT may perform Bayesian prediction.

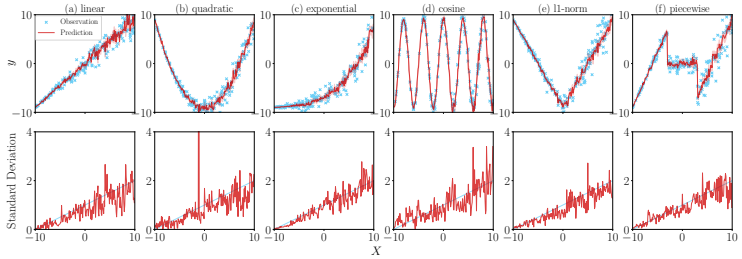

Figure 6: Investigating Bayesian prediction on six regression tasks. Compared to the observations (top), standard deviations of predictions from LIFT/GPT aligns well, across all datasets. This implies the similarity of LIFT/GPT to Bayesian predictions.

5.2 Does LIFT benefit from incorporating feature names?

Unlike standard machine learning algorithms, LIFT can be provided with the context information by including feature names in the prompts. For instance, consider a data sample from Iris dataset as "$\texttt{x} = (5.1, 3.5, 1.4, 0.2), \texttt{y} =$ `Iris-setosa`" where the feature names are "`sepal length, sepal width, petal length, petal width`". We can incorporating the context information when converting this sample into a sentence, as "`An iris plant with sepal length` 5.1 `cm, sepal width` 3.5 `cm, petal length` 1.4 `cm, petal width` 0.2 `cm. This plant belongs to Iris-setosa.`" Intuitively, including context information may improve the sample complexity as LIFT can utilize the prior knowledge for better predictions. We empirically verify this intuition, showing the result in Table. 5 and Appendix.

For classification, we observe that using feature names indeed improves the accuracies of LIFT/GPT on both two GPT models. We note that the accuracies on data 23 show no difference between using or not using feature names, while the differences are more significant on two other datasets. For regression, on the other hand, we observe that finetuning with column names does not necessarily help in reducing sample complexity. However, interestingly, for the Medical Insurance dataset, whose task is to predict the cost of medical insurance, LIFT with column names helps improve the sample complexity. We conjecture that the common knowledge embedded in feature names of the Medical Insurance dataset is much stronger in correlation to the responses than in other evaluated datasets.

DID	ODC	LIFT/GPT-J		LIFT/GPT-3	
		W/ Names	W/O Name	W/ Names	W/O Name
23	42.71	48.36±0.97	**49.83±0.28**	57.48±1.14	**57.74±0.89**
48	35.48	**66.05±5.84**	61.29±4.56	**67.74±3.02**	55.91±4.02
54	25.88	**69.02±3.67**	64.31±2.37	**72.16±2.00**	70.20±2.73

Table 5: Classification results for with or without feature names on OpenML datasets with IDs 23, 48 and 54.

6 Improving LIFT with existing techniques

Previous sections illustrate the effectiveness of standard LIFT in learning non-language tasks. In this section, we provide two techniques for improving LIFT.

6.1 Two-stage intermediate finetuning for LIFT with synthetic pretext tasks

In Sec. 4.3, we observe that LMs need a sufficient number of samples to start adapting to the non-language tasks. We suspect that LMs need early samples to learn the task description: input space, label space, and the sentence template [45]. Therefore, we suggest utilizing synthetic data instead of real training data to describe the task for LMs, thus reducing the

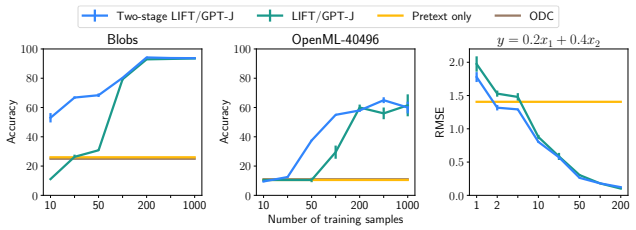

Figure 7: Two-stage finetuning. We first apply LIFT on synthetic pretext data, then continue applying LIFT on real datasets. The inter-finetuning (blue) outperforms finetuning (green) when the number of training samples is small.

sample complexity. Motivated by recent works in pretraining for few-shot learning [76], we propose a two-stage intermediate finetuning approach. Given a task, we generate a simple mixture of Gaussian synthetic data with the same number of features and classes as the real dataset. We first apply LIFT on the LM with this synthetic dataset for a few epochs, then continue the procedure with the real training dataset.

Fig. 7 shows our comparison between two-stage LIFT (inter-finetuning) with the original LIFT (finetuning). We find that intermediate finetuning consistently improves the performances of LIFT on both classification and regression tasks when the number of training samples is small.

6.2 Data augmentation

In various classification tasks, data augmentation [78] has been considered a simple tool for improving the generalization performance. We focus on mixup [77], a popular data augmentation that makes virtual samples by mixing the features/labels of real samples. We test whether mixup helps in the LIFT framework. Specifically, we finetune GPTs using the prompt of mixup samples and compare the accuracy of GPTs finetuned with original real samples. Table 6 shows our results for various OpenML datasets. We observe that mixup does not help LIFT on both GPT-3 and GPT-J, while mixup guarantees sufficient gain for logistic regression.

Setting / OpenML ID		467	683	736	855	859	874	911	1005
LogReg	no mixup	85.0	76.7	72.4	63.3	95.7	97.3	67.4	60.0
	mixup	90.0	81.1	76.5	66.0	98.3	100.0	70.1	64.9
GPT-J	**no mixup**	90.9	66.7	60.9	83.0	93.3	30.0	88.0	53.5
	mixup	45.5	25.0	65.2	34.0	93.3	10.0	40.0	25.58
GPT-3	**no mixup**	90.9	63.9	65.2	81.0	100.0	76.7	86.0	73.6
	mixup	78.8	47.2	59.4	68.67	95.56	63.33	82.0	65.12

Table 6: Classification accuracy (%) on various OpenML datasets, with/without mixup [77]. For both GPT-J and GPT-3, mixup is not helping LIFT, while improving the accuracy in logistic regression training.

7 Discussion and Conclusion

Limitations and open questions Despite promising performances on various tasks and settings, we observe some limitations of LIFT to basic learning tasks. LIFT/GPT do not perform well if the features have high dimensions (for regression), or when the output space is large (for classification). Also, the context length of LIFT is restricted to the context length of LMs, which can limit the feature dimension of the datasets. Besides, our works open some interesting questions for future works. First, based on the similarity of decision boundaries, do LMs and LIFT/GPT share similar behaviors to ensemble methods or decision tree? Secondly, are LMs universal models that can adapt well to any modalities and domains? Lastly, can LIFT adapts better for regression task using more sophisticated encoding schemes for numeric features?

In this work, we study the Language-Interfaced FineTuning (LIFT) framework for solving non-language downstream tasks without changing models' architecture or loss function. LIFT first converts labeled samples into sentences and then finetune pretrained LMs on the sentence dataset using the standard finetuning method and loss function. Via an extensive empirical study on a suite of non-language classification and regression tasks, we show that LIFT/GPT performs relatively well on low-dimensional classification and regression tasks. We find LIFT/GPT robust against outliers and label corruption and share behaviors of Bayesian prediction. We further improve LIFT with the use of in-context feature names and two-stage intermediate finetuning methods. While LIFT still suffers from the high dimensionality of features in regression tasks and the large numbers of classified categories, the findings of LIFT open a few interesting directions for future research investigations. Nevertheless, our work is arguably one of the first to thoroughly study the efficacy of LIFT with pretrained language models on standard regression and classification tasks, paving the way for enabling "no-code machine learning with language models".

Social impacts Future research should also investigate potential fairness issues of applying LIFT. Based on large language models, LIFT might have embedded bias targeting certain social groups. Especially when feature names are included in the training prompts, it is possible that the models will be more sensitive to social biases and thus might make unfair and even harmful predictions.

References

[1] David Forsyth and Jean Ponce. Computer vision: A modern approach. Prentice hall, 2011.

[2] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

[3] KR1442 Chowdhary. Natural language processing. Fundamentals of artificial intelligence, pages 603–649, 2020.

[4] 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. On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258, 2021.

[5] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. Where does alphago go: From church-turing thesis to alphago thesis and beyond. IEEE/CAA Journal of Automatica Sinica, 3(2):113–120, 2016.

[6] Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar: An evolutionary computation perspective. In Proceedings of the genetic and evolutionary computation conference companion, pages 314–315, 2019.

[7] Shih-Chung B Lo, Heang-Ping Chan, Jyh-Shyan Lin, Huai Li, Matthew T Freedman, and Seong K Mun. Artificial convolution neural network for medical image pattern recognition. Neural networks, 8(7-8):1201–1214, 1995.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017.

[10] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[13] David So, Quoc Le, and Chen Liang. The evolved transformer. In International Conference on Machine Learning, pages 5877–5886. PMLR, 2019.

[14] Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. arXiv preprint arXiv:1706.05137, 2017.

[15] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. Journal of Big data, 3(1):1–40, 2016.

[16] Dhanesh Ramachandram and Graham W Taylor. Deep multimodal learning: A survey on recent advances and trends. IEEE signal processing magazine, 34(6):96–108, 2017.

[17] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning, pages 160–167, 2008.

[18] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. arXiv preprint arXiv:2103.05247, 2021.

[19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[20] Belinda Z Li, Jane Yu, Madian Khabsa, Luke Zettlemoyer, Alon Halevy, and Jacob Andreas. Quantifying adaptability in pre-trained language models with 500 tasks. arXiv preprint arXiv:2112.03204, 2021.

[21] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. arXiv preprint arXiv:1704.01444, 2017.

[22] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In International Conference on Machine Learning, pages 12697–12706. PMLR, 2021.

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

[24] Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A Smith. Story cloze task: Uw nlp system. In Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics, pages 52–55, 2017.

[25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.

[26] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In International Conference on Machine Learning, pages 2790–2799. PMLR, 2019.

[27] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. Advances in neural information processing systems, 30, 2017.

[28] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Guihong Cao, Daxin Jiang, Ming Zhou, et al. K-adapter: Infusing knowledge into pre-trained models with adapters. arXiv preprint arXiv:2002.01808, 2020.

[29] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. arXiv preprint arXiv:2104.08786, 2021.

[30] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heinz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. arXiv preprint arXiv:2111.01243, 2021.

[31] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[32] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[33] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942, 2019.

[34] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.

[35] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. arXiv preprint arXiv:2204.02311, 2022.

[36] Alex Tamkin, Trisha Singh, Davide Giovanardi, and Noah D. Goodman. Investigating transferability in pretrained language models. In FINDINGS, 2020.

[37] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. arXiv preprint arXiv:1605.09782, 2016.

[38] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International conference on machine learning, pages 1597–1607. PMLR, 2020.

[39] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

[40] Ting Chen, Mario Lucic, Neil Houlsby, and Sylvain Gelly. On self modulation for generative adversarial networks. arXiv preprint arXiv:1810.01365, 2018.

[41] Jason Phang, Thibault Févry, and Samuel R Bowman. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. arXiv preprint arXiv:1811.01088, 2018.

[42] Raphael Rubino and Eiichiro Sumita. Intermediate self-supervised learning for machine translation quality estimation. In Proceedings of the 28th International Conference on Computational Linguistics, pages 4355–4360, 2020.

[43] Nikita Moghe, Mark Steedman, and Alexandra Birch. Cross-lingual intermediate fine-tuning improves dialogue state tracking. arXiv preprint arXiv:2109.13620, 2021.

[44] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? arXiv preprint arXiv:2101.06804, 2021.

[45] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? arXiv preprint arXiv:2202.12837, 2022.

[46] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In International Conference on Machine Learning, pages 1691–1703. PMLR, 2020.

[47] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Kathryn Tunyasuvunakool, Olaf Ronneberger, Russ Bates, Augustin Žídek, Alex Bridgland, et al. High accuracy protein structure prediction using deep learning. Fourteenth Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book), 22:24, 2020.

[48] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. Advances in neural information processing systems, 34, 2021.

[49] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. arXiv preprint arXiv:2110.01889, 2021.

[50] Sercan O Arık and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In AAAI, volume 35, pages 6679–6687, 2021.

[51] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678, 2020.

[52] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67, 2020.

[53] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In International Conference on Machine Learning, pages 8748–8763. PMLR, 2021.

[54] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. arXiv preprint arXiv:2102.01293, 2021.

[55] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of transformers with simple arithmetic tasks. arXiv preprint arXiv:2102.13019, 2021.

[56] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In International Conference on Learning Representations, 2019.

[57] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 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, November 2019. Association for Computational Linguistics.

[58] Tin Kam Ho. Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition, volume 1, pages 278–282. IEEE, 1995.

[59] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. Xgboost: extreme gradient boosting. R package version 0.4-2, 1(4):1–4, 2015.

[60] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax. May 2021.

[61] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. SIGKDD Explorations, 15(2):49–60, 2013.

[62] Yann LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

[63] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.

[64] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211, 2013.

[65] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon Shaolei Du, Ken-Ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In International Conference on Learning Representations, 2021.

[66] Medical insurance dataset (kaggle). https://www.kaggle.com/datasets/mirichoi0218/insurance.

[67] Pınar Tüfekci. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. International Journal of Electrical Power & Energy Systems, 60:126–140, 2014.

[68] John R Quinlan et al. Learning with continuous classes. In 5th Australian joint conference on artificial intelligence, volume 92, pages 343–348. World Scientific, 1992.

[69] Paulo Cortez and Alice Maria Gonçalves Silva. Using data mining to predict secondary school student performance. 2008.

[70] Shai Shalev-Shwartz and Shai Ben-David. Understanding machine learning: From theory to algorithms. Cambridge university press, 2014.

[71] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. October 2021.

[72] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, and Tom Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective. arXiv preprint arXiv:2203.08124, 2022.

[73] Charles Lovering, Rohan Jha, Tal Linzen, and Ellie Pavlick. Predicting inductive biases of pre-trained models. In International Conference on Learning Representations, 2020.

[74] Yanyao Shen and Sujay Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In International Conference on Machine Learning, pages 5739–5748. PMLR, 2019.

[75] Peter J Huber. Robust statistics. In International encyclopedia of statistical science, pages 1248–1251. Springer, 2011.

[76] Mingda Chen, Jingfei Du, Ramakanth Pasunuru, Todor Mihaylov, Srini Iyer, Veselin Stoyanov, and Zornitsa Kozareva. Improving in-context few-shot learning via self-supervised training. arXiv preprint arXiv:2205.01703, 2022.

[77] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412, 2017.

[78] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of big data, 6(1):1–48, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The contribution and scope of this paper has been summarized in the abstract and the last part of Introduction.
   (b) Did you describe the limitations of your work? [Yes] We wrote the limitations in Sec.7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] This has been discussed in Sec.7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Our paper conforms to the ethics guideline.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code, data, and instructions are provided in the shared anonymous GitHub repository.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All the training details are discussed in the paper and they can be found at the anonymous GitHub repository.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All the experiments are run multiple times and at the tables mean and standard deviation of those runs are presented.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] This has been described in Sec.3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Our work uses public datasets/models, and has been cited properly, in the paper and GitHub repo.
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]