Personalized Benchmarking with the Ludwig Benchmarking Toolkit

Avanika Narayan, Piero Molino, Karan Goel, Willie Neiswanger, Christopher Ré
Department of Computer Science
Stanford University
{avanika, pmolino, kgoel, neiswanger, chrismre}@cs.stanford.edu,

Abstract

The rapid proliferation of machine learning models across domains and deployment settings has given rise to various communities (e.g. industry practitioners) which seek to benchmark models across tasks and objectives of personal value. Unfortunately, these users cannot use standard benchmark results to perform such value-driven comparisons, as traditional benchmarks evaluate models on a single objective (e.g. average accuracy) and don't facilitate a standardized training framework that controls for confounding variables (e.g. computational budget), making fair comparisons difficult. To address these challenges, we introduce the open-source Ludwig Benchmarking Toolkit (LBT), a personalized benchmarking toolkit for running end-to-end benchmark studies (from hyperparameter optimization to evaluation) across an easily extensible set of tasks, deep learning models, datasets and evaluation metrics. LBT provides a configurable interface for customizing evaluation and controlling training, a standardized training framework for eliminating confounding variables, and support for multi-objective evaluation. We demonstrate how LBT can be used to create personalized benchmark-studies with a large-scale comparative analysis for text classification across 7 models and 9 datasets. We explore the trade-offs between inference latency and performance, relationships between dataset attributes and performance, and the effects of pretraining on convergence and robustness, showing how LBT can be used to satisfy various benchmarking objectives.

Code Repository: https://github.com/HazyResearch/ludwig-benchmarking-toolkit

1 Introduction

Benchmarking has emerged as an important practice to measure progress in machine learning. Typically, benchmarking is done through leaderboards, where a participant's objective is to maximize a performance criterion on a challenging task or dataset. Prominent examples of these benchmarks include GLUE [1], SuperGLUE [2], ImageNet [3] and SQuAD [4].

As deep learning models become increasingly proficient at maximizing performance criteria like average accuracy, attention has shifted towards the need for more personalized, thorough, and thoughtful benchmarking that emphasizes a community or individual's needs [5]. In this work, we focus in particular on what we term *value-driven communities*—communities whose utility is aligned with optimizing and understanding model evaluation objectives beyond average performance. Examples of such communities include researchers interested in understanding the effects of model pretraining on robustness, and industry practitioners interested in the tradeoffs between inference latency and performance. However, standard benchmarking practices carry several limitations that makes personalizing benchmarks difficult for these communities.

First, the shift to personalized benchmarks changes the nature of benchmark design, turning users into benchmark designers. The major burden on benchmark designers so far has been in formulating a challenging task, collecting and preparing data, and selecting an appropriate performance criterion to capture progress on the task. Personalization transforms this burden from managing dataset collection

Preprint. Under review.

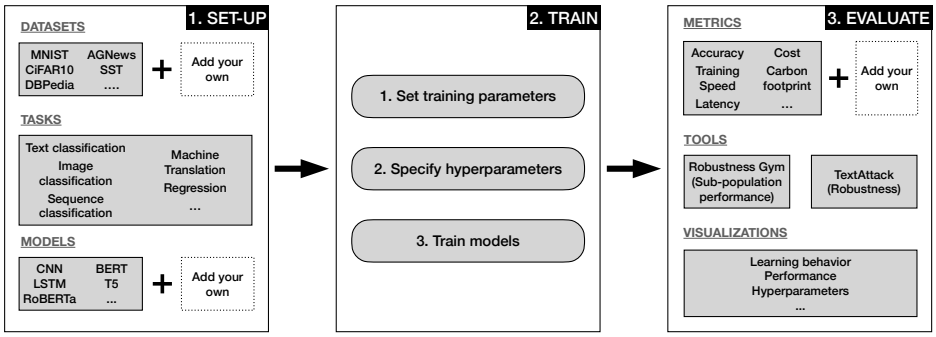


Figure 1: **Ludwig Benchmarking Toolkit.** LBT supports multi-objective evaluation, provides a standardized training framework, and includes an extensible set of datasets, models and metrics.

and curation to also managing careful training and evaluation. This shift requires new tools that permit finer-grained control over benchmarking studies, where users can customize training and evaluation based on their needs while automating as much of this burden as possible.

Second, a general goal for benchmarking is to help researchers and practitioners make apples-to-apples model comparisons and draw accurate conclusions about why some models perform better or worse. The mechanism adopted by leaderboard benchmarks limits the ability to precisely answer such questions because submitted models vary substantially: in the data, compute resources, preprocessing, training protocol, and implementations [6]. These confounding factors make it difficult to draw any conclusion about what parts of an implementation were ultimately responsible for its performance, especially since these factors may matter even more than architecture differences [7, 8, 9, 10, 11].

Lastly, existing benchmarks provide relatively little utility to an individual who wants to compare a collection of models on multiple objectives such as robustness, training speed, inference latency, size, or other properties—all aspects that are of interest to the value-driven communities of researchers and practitioners that we focus on [5, 12]. Instead, leaderboards excel at catalyzing progress in the larger research community e.g. moving from 70% to 90% accuracy on GLUE in 2 years [13].

Taken together, these challenges highlight the need for personalized benchmarking tools that complement existing leaderboard-style benchmarks, and allow researchers and practitioners in value-driven communities to (i) configure benchmark training and evaluation, (ii) fairly compare models by controlling for confounding variables, and (iii) perform multi-objective evaluation.

We take a first step in this direction and introduce the **Ludwig Benchmarking Toolkit (LBT)**, a personalized benchmarking toolkit for creating and running *configurable, standardized, and multi-objective* benchmarking studies with ease. To create a benchmark suite in LBT, users specify a task, a set of models to compare and datasets for evaluation, configure training and hyperparameter search spaces for model training, and evaluate and compare the trained models, as depicted in Figure 1. In particular, LBT has the following properties:

Configurable. To support configurability, LBT provides out-of-the-box support for training cutting-edge deep learning models that span classification, regression, and generation tasks across multiple modalities. LBT enables users to control training conditions by providing a simple configuration file interface for specifying training parameters and the hyperparameter search space. To support personalization, LBT makes it simple to extend the toolkit, and gives users explicit mechanisms for introducing custom models, datasets, and metrics.This is particularly useful for benchmarking models in application-specific scenarios.

Standardized. To enable fair comparisons between models trained using the toolkit, LBT implements a standardized training framework that ensures every model can be trained using the same dataset splits, preprocessing, training loop, and hyperparameters. During configuration, users choose which variables to hold constant across models (e.g., training time, the optimizer, preprocessing techniques, and the hyperparameter tuning budget), controlling for any potential confounders.

Multi-objective. To provide greater support for varied evaluation metrics, LBT expands the set of evaluation criteria beyond standard performance-based evaluation (e.g., average accuracy, F1 score) to include fiscal cost, size, training speed, inference latency, and carbon footprint [14]. Furthermore, to enable developers to compare models on the basis of robustness and critical subpopulation

performance (evaluation factors relevant for application deployment), LBT includes integrations with two popular open-source evaluation toolkits, TextAttack [15] and Robustness Gym [16].

We validate that value-driven communities can use LBT to conduct personalized benchmarking studies by performing a large-scale, multi-objective comparative analysis of 7 deep learning models across a diverse set of 9 text classification datasets. We explore hypotheses that are of interest to researchers and practitioners, including on the tradeoffs between inference latency and performance, relationships between dataset attributes and performance, and the effects of pretraining on convergence and robustness, all while controlling for important confounding factors. Our results show that DistilBERT has the best inference efficiency and performance trade-off, BERT is the least robust to adversarial attacks, and that pretrained models do not always converge faster than models trained from scratch.

2 Related Work

There have been several impactful works contributing to the landscape of model benchmarking. We provide a brief overview of these efforts and discuss how they relate to our work.

Critiques of leadboard-style benchmarks. Recently, leaderboard-style benchmarks have been critiqued extensively. Ethayarajh and Jurafsky [5] argue that existing leaderboards are poor proxies for the natural language processing (NLP) community and advocate that they report additional metrics of practical concern (e.g. model size) to enable users to build personal leaderboards. Furthermore, Rogers [6] critiques the lack of standardization in entries submitted to leaderboards suggesting that inequity in compute and data used at training time makes fairly comparing models on the basis of these reported results difficult. The aforementioned critiques are key motivations for our work.

Flexible leaderboards. Earlier this year, Liu et al. [17] introduced ExplainaBoard, an interactive leaderboard and evaluation software for interpreting 300 NLP models. Like LBT, Explainaboard provides tooling for fine-grained analysis and seeks to make the evaluation process more interpretable. However, it does not provide a standardized training and implementation framework that addresses the challenge of confounds when making model comparisons. Another flexible leaderboard is DynaBench [18], a platform for dynamic data collection and benchmarking for NLP tasks that addresses the problem of static datasets in benchmarks. DynaBench dynamically crowdsources adversarial datasets to evaluate model robustness. While LBT focuses on the model implementation and evaluation challenges of benchmarking, Dynabench's focus is on data curation. Most recently, Facebook introduced Dynaboard [19], an interface for evaluating models across a holistic set of evaluation criteria including accuracy, compute, memory, robustness, and fairness. Similar to LBT, Dynaboard enables multi-objective evaluation. However, Dynaboard's focuses less on configuring personalized benchmark studies, as users cannot introduce their own evaluation criteria or datasets.

Benchmarking deep learning systems. Performance oriented benchmarks like DAWNBench and MLPerf [20] evaluate end-to-end deep learning systems, reporting many efficiency metrics such as training cost and time, and inference latency and cost. They demonstrate that fair model comparisons are achievable with standardized training protocols, and our work is motivated by these insights.

Benchmarking tools. To our knowledge, there is a limited set of toolkits for configuring and running personalized benchmarking studies. ShinyLearner [21] is one such solution that provides an interface for benchmarking classification algorithms. However, ShinyLearner only supports classification tasks, a small number of deep learning architectures (e.g. does not support any pretrained language models) and only reports performance-based metrics.

3 The Ludwig Benchmarking Toolkit (LBT)

In Section 3.1 we describe the communities that LBT is intended to serve. In Section 3.2 we provide an overview of LBT and an example of how it is used. Lastly, in Section 3.3, we provide a more detailed discussion of the properties and features of LBT, including how LBT addresses the needs of the communities described in Section 3.1.

3.1 Benchmarking for Value-Driven Communities

We start by describing the users that LBT primarily targets. In particular, LBT best supports the needs of communities that satisfy the following characteristics:

1. **Value driven.** The community is aligned around objectives for which average accuracy alone is not a good proxy (e.g. training speed). Users' goals are primarily to compare models using evaluations that align with their objectives.
2. **Prefer automation.** Users value the ability to control and configure their benchmarks, but do not want or do not know how to implement a full experimental framework from scratch.
3. **Require standardization.** Users place strong emphasis on conducting clear, standardized analyses where the training and hyperparameter optimization processes are carefully controlled, in order to advance understanding and draw accurate conclusions.

Taken together, these characteristics allow us to more precisely target communities that remain underserved by traditional benchmarks [5]. To ground this discussion, we provide examples of three communities that satisfy the characteristics we outlined:

- **ML researchers interested in performing comparative meta-analyses.** These users are researchers with extensive experience in training and evaluating ML models. Their goal is to compare models across various objectives (e.g., learning dynamics, bias, fairness, robustness, efficiency) and tease apart the effects of preprocessing, hyperparameters, and modeling choices (e.g., pretraining, model architectures) on performance. They would benefit from a standardized pipeline for training and evaluation (for fair comparison and accurate analyses), access to robust training metadata and evaluation metrics, and tooling to perform fine-grained evaluation. Since these users are experts, they require explicit mechanisms for customization (e.g., custom datasets, models and metrics).
- **Industry practitioners interested in deployment-readiness.** These users are engineers with low to medium experience training and evaluating ML models. Their goal is to find the best model for their task of interest as quickly as possible, taking into account deployment-specific criteria such as inference latency and training speed. They would benefit from a simple user-interface that removes the need to write any deep learning code, provides extensive reporting of metrics, and provides out-of-the-box support for common ML tasks and architectures. They value the ability to add application specific datasets and evaluation criterea to their benchmark study.
- **Subject-matter experts interested in task-specific performance.** These users are domain experts (e.g. radiologists) with limited experience training and evaluating deep learning models. Their goal is to find the best model for their task of interest based on performance on domain-specific data (e.g. ECG data for arrhythmia classification) and specific error metrics. Similar to the previous example, they would prefer a low-code, simple user interface and necessitate configurability to introduce domain-specific datasets and metrics.

In the following sections, we describe LBT's *configurable, standardized, and multi-objective* toolkit, and show how it can enable value-driven communities to create personalized benchmark studies.

3.2 Toolkit Overview and Usage

First, we provide a brief overview of LBT and describe how it is used. LBT enables users to run end-to-end model benchmarking studies and is composed of four main components: off-the-shelf task, model, and dataset support, model training, evaluation, and a shared research database. Users can choose from a large number of tasks, train models with a pipeline that provides standardization (e.g., of preprocessing, training loops, and hyperparameter searches), evaluate results across objectives of interest, and publish benchmark outcomes to a shared Elasticsearch [22] database. All components in this toolkit are configurable from a set of simple files.

Running benchmarking experiments in LBT is an easy five-step process. In each experiment, users populate three configuration files: one for their task, model, and hyperparameter space. We describe these configuration files and the five-step process next.

1. Define the experiment: Users can choose from any of the supported tasks, models, and datasets already available in LBT or easily add their own. Each supported task has an associated task config file that specifies the end-to-end model structure corresponding to the task.

Example: Consider a text classification experiment, comparing the performance of an RNN and the ELECTRA model. We will run the experiment across 2 datasets: Social Bias Frames and Hate Speech and Offensive Language. Figure 2 (center) presents a sample task config file.

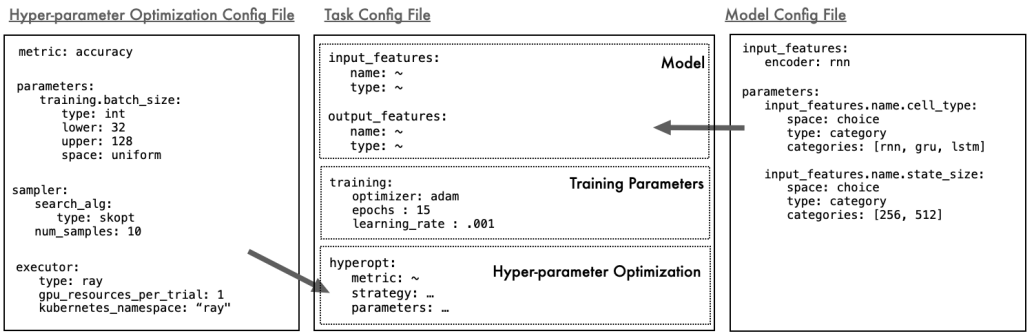


Figure 2: **Sample LBT configuration files.** Setting-up an experiment in LBT requires populating 3 configuration files that define task, model / training parameters and hyperparameter optimization.

2. Specify the training parameters and hyperparameter search: Users specify values for the hyperparameters that should be constant across all training runs, as well as the hyperparameters they would like to optimize.

Example: We specify model-specific parameters and their search space in the model config files (Figure 2: right). To control for the optimizer, learning rate, and training epochs, we set their values to "adam", "0.0001", and "15" in the task configuration file. We specify our optimization metric (validation accuracy), parameters we would like to optimize (batch size) and our search algorithm (skopt) in the hyperparameter config file (Figure 2: left).

3. Run the optimization experiment: Running an experiment is a simple one-line command. When an experiment is run, a configuration file for each task, model, and dataset combination is saved (See Figure A.4 for an example). The saved file records the model architecture, training variables, and hyperparameter settings and can be used to reproduce an experiment with a one-line command: `python experiment_driver.py -reproduce <path to experiment config file>`

Example: We choose to run our experiment on a GCP cluster across four machines. We specify our compute environment by passing in a flag at runtime and by specifying the name of our Kubernetes cluster in the hyperparameter configuration file (Figure 2: left).

4. Evaluate the results: Users can perform an in-depth meta-analysis using the set of performance-based evaluation metrics recorded by Ludwig during model training, along with the additional metrics logged by LBT. On top of analyses performed using the reported metrics, users can use the TextAttack and Robustness Gym APIs to better understand fine-grained aspects of model performance.

Example: We want to gain a better understanding of bias in our offensive language detection classifiers. We test if these models classify text with African American Vernacular English (AAVE) as offensive more frequently than without [23]. We define a Robustness Gym slice for samples containing words unique to AAVE, and compare model performance on this subpopulation to identify bias.

5. Publish the experiment: All experiments run using LBT can be uploaded to a shared Elasticsearch research database. Due to the flexible nature of Elasticsearch, users can update experiments with any additional metrics and analyses over the duration of their study.

Example: To publish results to the database we add one command-line flag at runtime: `python experiment_driver.py ... -esc elasticsearch_config.yaml`

3.3 Toolkit Design and Features

Next, we describe the key design choices and features of LBT which enable value-driven communities to create personalized benchmarks.

Configurable

Out-of-the-box support for tasks, datasets, and models. LBT integrates directly with the popular *Ludwig Deep Learning Toolbox (Ludwig)* [24], enabling LBT to use the existing models, datasets, and hyperparameter tuning methods available in Ludwig. Thus, LBT can support several tasks out-of-the-box like multi-class and multi-label classification, regression, sequence tagging, and sequence generation over a diverse set of input data types such as tabular, image, text, audio, and time series.

Simple user-interface for configuration. Configuring a benchmark study in LBT is as simple as specifying a task (e.g. image classification), choosing a set of models to compare from the ones available (or implementing a new one), selecting datasets for evaluation and declaring training parameters. This is achieved by populating declarative configuration files for the benchmark task, training parameters, model-specific parameters, and hyperparameter search space (see Figure 2).

Extensible to new tasks, datasets, models and evaluations. LBT provides explicit mechanisms for users to personalize and extend the toolkit to their needs. Figure A.1 illustrates how to register a new dataset and custom evaluation metric. Adding new models and tasks is simple, and requires implementing a new instance of an encoder or decoder in Ludwig (functions mapping from input data to hidden representation and from hidden representation to predictions respectively).

Standardized

Standardized model training. To ensure that models trained in LBT can be compared fairly, LBT includes a standardized framework for training and hyperparameter optimization. Using this framework, models can be trained using the same dataset splits, preprocessing techniques, training loop, and hyperparameter search space if necessary. LBT harnesses the extensive hyperparameter tuning support in Ludwig to provide automated hyperparameter optimization when training benchmark models. LBT supports running distributed, multi-node experiments both locally or on remote clusters such as Google Cloud Provider (GCP), Amazon Web Services (AWS), and SLURM.

Shared research database. To support communities in sharing, replicating, and extending experiments, we provide access to a shared research database that stores the results, reported metrics, and metadata of experiments run in LBT. Experiments are uploaded to the database along with their configuration files. Users can search the database to view and download experiments run by other users, and reproduce them using the experiment's configuration file.

Multi-Objective LBT exposes three flavors of evaluation support: metrics, tools, and visualizations.

Metrics for multi-objective evaluation. With respect to metrics, LBT expands the scope of traditionally reported evaluation metrics (e.g. average accuracy) to include cost, efficiency, training speed, inference latency, model size, and more. Table A.1 details the additional metrics supported in LBT.

Integrations for fine-grained evaluation. To further support custom evaluations, LBT enables users to compare models based on robustness. In this work, we define robustness as critical subpopulation performance [16] and sensitivity to adversaries and input perturbations [15, 25], acknowledging that this is not a universal definition of robustness as other dimensions of robustness exist (e.g. robustness to online distributional shift). Nonetheless, LBT integrates with two open-source evaluation tools for measuring robustness: Robustness Gym (RG) [16] and TextAttack [15]. LBT's API for RG lets users inspect model performance on a set of pre-built subpopulations (e.g., sentence length, image color etc.), as well as add more subpopulations for their data and use cases (see Figure A.2 for an example). The TextAttack integration helps LBT users evaluate model robustness to input perturbations (see Figure A.3 for sample API usage.

Visualizations. Finally, LBT provides an API to generate visualizations for learning behavior, model performance, and hyperparameter optimization, using statistics generated during model training.

4 Case Study: Large-Scale Text Classification Analysis

Next, we demonstrate how users with diverse benchmarking objectives can configure personalized benchmarks and conduct deep, comparative meta-analyses using LBT. The goal of this case study is twofold. First, we seek to show that when using LBT we can replicate previously reported experimental results accurately. Second, we want to demonstrate how LBT can address the unmet needs of value-driven communities in running configurable, standardized, and multi-objective benchmark studies. As such, the goal is not to show novel insights into models but rather to demonstrate the practicality and usability of the toolkit. With these goals in mind, we use LBT to conduct a large-scale text classification benchmark study that spans 4 tasks, 9 datasets, and 7 models with a total of 1260 trained models, all evaluated across a variety of metrics. To ground our benchmarking, we use the metrics and tools supported by LBT to study a few relationships in particular: the tradeoff between efficiency and performance, effects of dataset attributes on performance, and the effects of pretraining on performance. We provide experimental details in Section 4.1 and describe our hypotheses and findings in Section 4.2.

Table 1: **Overall Performance**. The table reports the accuracy of the top performing models for each dataset and model pair.

	Dataset								
Model	HS	AG	SST5	MGB	IR	GE	YR	DBP	SBF
RNN	0.875	0.910	0.476	0.879	0.769	0.458	0.954	0.986	0.653
Stacked Parallel CNN	0.883	0.911	0.468	0.883	0.753	0.448	0.948	0.986	0.640
DistilBERT-base	0.915	0.934	0.528	0.888	0.758	0.549	0.965	0.991	0.675
BERT-base	0.919	0.943	0.530	0.892	0.801	0.546	0.969	0.992	0.687
ELECTRA-base	0.911	0.932	0.540	0.896	0.747	0.542	0.969	0.990	0.663
T5-small	0.912	0.935	0.541	0.894	0.769	0.535	0.968	0.991	0.680
RoBERTa-base	0.918	0.940	0.551	0.898	0.780	0.541	0.973	0.991	0.687

4.1 Experimental Setup

We conduct benchmarking on text classification tasks due to the abundance of available datasets, task-suitable models and published results [26, 27, 28].

Datasets. We chose the following nine classification datasets: Hate Speech and Offensive Language (HS) [29], AGNews (AG) [30], DBPedia (DBP) [30], Yelp Review Polarity (YR) [30], SST5 (SST5) [31], MD Gender Bias (MGB) [32], Irony Classification (IR) [33], GoEmotions (GE) [34] and Social Bias Frames (SBF) [35]. The datasets were chosen based on their diversity in average sentence length (ranging from 5 to 132), dataset size (ranging from 1364 to 56000), number of classes (ranging from 2 to 27), and language type (e.g. formal vs. informal language). The datasets cover four common text classification tasks: sentiment analysis, emotion classification, topic classification, and hate and offensive speech detection, and span both binary and multi-way classification.

Models. We analyze five pretrained language models and two text encoders trained from scratch. The pretrained models are BERT-base [36], DistilBERT-base [37], Electra-base [38], RoBERTa-base [39], T5-small [40] and were chosen due to variance in size and pre-training strategies. The text encoders are a stack of bidirectional RNN layers (with the cell type chosen to be RNN, Long Short-Term Memory layer (LSTM) [41], or Gated Recurrent Unit (GRU) [42]) and a stacked implementation of the CNN for sentence classification [43] (Stacked Parallel CNN or SP-CNN).

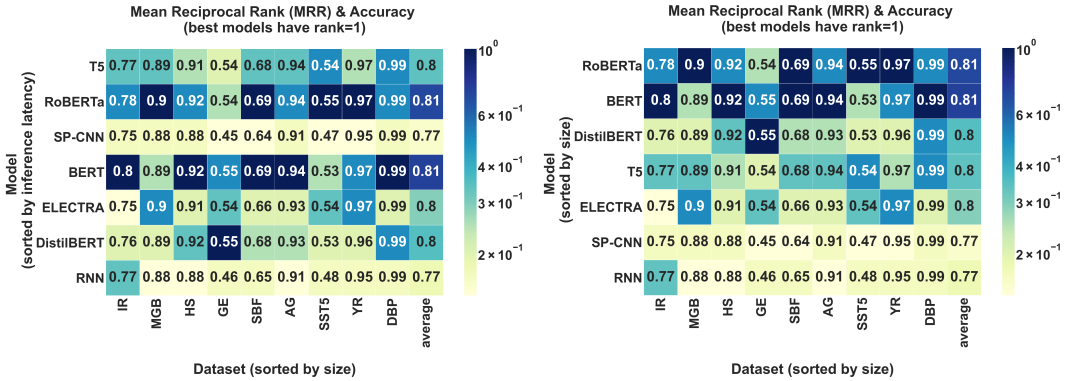
Hyperparameters. Across all experiments we use the Adam optimizer [44] and the Scikit Optimize (skopt) hyperparameter search algorithm [45], sampling 20 hyperparameter settings per dataset and model pair (e.g., BERT and SST5). We optimize over learning rate, model hidden dimension size, and model-specific parameters such as cell type for the RNN model and size and number of stacked layers for the SP-CNN. All experiments used Tesla T4 GPUs on GCP.

4.2 Results and Analysis

Average Accuracy Analysis. First, we compare models on the basis of average accuracy to demonstrate that standard, accuracy-based benchmark comparisons are possible in LBT. Table 1 shows the performance of the best hyperparameter configuration for each model-dataset pair. We verify that these results are aligned with previously reported experimental results [27, 28]. Based on average accuracy alone, RoBERTa-base and BERT-base have the best performance across all nine datasets. We note that on some datasets, the accuracy gap between the best and worst model is as large as 0.09 (GE, SST5) while only 0.02 on others (MGB, DBP).

Value-driven Analysis. Next, we aim to validate the efficacy of LBT in enabling value-driven communities to create personalized benchmark studies. We do this by structuring our analysis into three themes: efficiency and performance tradeoffs, effects of dataset attributes on performance, and effects of pretraining on performance. We use LBT to test multiple hypotheses related to these themes. The proposed hypotheses and associated analysis demonstrate how users with various objectives can effectively use LBT to achieve their benchmarking objectives.

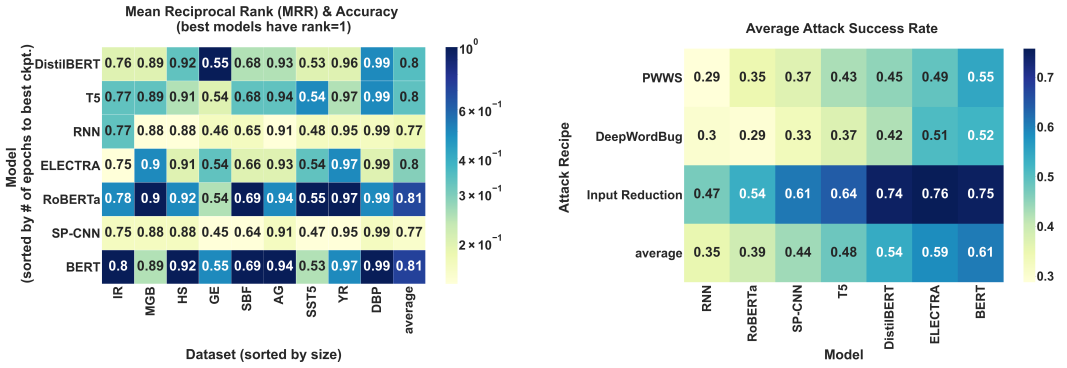
1. Inference Latency and Performance Tradeoffs: For an engineer looking to deploy a text-classification model in production, comparing models based on their size and inference efficiency is of significant interest as low latency is critical to delivering real-time, inference-based services. To demonstrate that LBT supports such a benchmark comparison, we investigated whether there is



(a) Model size and inference latency are not directly correlated.

(b) Larger models perform better than smaller ones, regardless of dataset size.

Figure 3: **Mean Reciprocal Rank & Accuracy.** In (a) and (b) numbers are accuracy scores and the colors represent the MRR of a model for each dataset (darker indicates better performance).



(a) DistilBERT and T5 take the longest to converge.

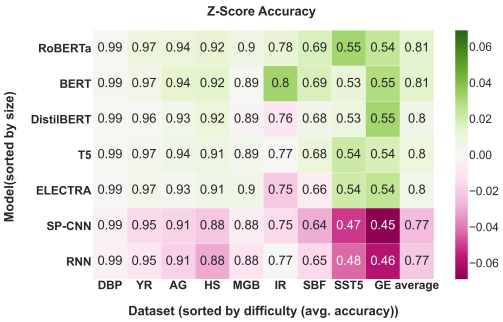
(b) BERT and ELECTRA are the least robust to adversarial attacks.

Figure 4: **Effects of Pretraining**. In (a), the numbers are accuracy scores and the colors represent the MRR of a model for each dataset (darker indicates better performance). In (b), the numbers are the average attack success rate of an attack strategy across all datasets.

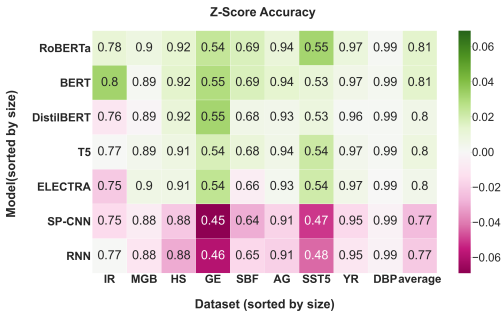
a trade-off between performance and latency and if better-performing models, which are typically larger, have slower inference speeds. In Figure 3a, we see that BERT, which obtains the best performance on the largest number of datasets, has lower latency than RoBERTa and T5-small, suggesting that inference efficiency and performance are not directly related. Our results also indicate that DistilBERT has a very convincing tradeoff between inference speed and performance.

2. Dataset Attributes and Performance: For practitioners trying to find the best model for their datasets, it is useful to better understand how model performance differs as a function of dataset attributes such as number of samples or average sentence length. We show how LBT can be used to understand these relationships by testing the following hypotheses. Based on existing works in the literature [46], we hypothesized that simpler models would perform better on smaller datasets as they overfit less. Figure 3b indicates that larger models outperform the smaller, simpler models across all datasets.

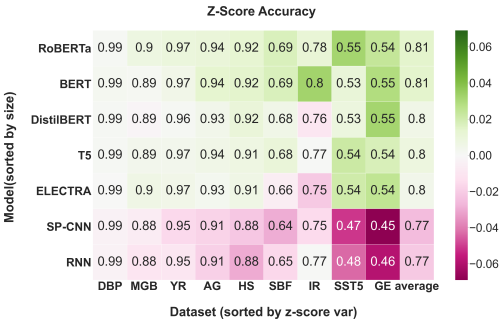
Furthermore, we hypothesized that evaluating on smaller datasets would result in the most variance in model performance. However, Figure 5b shows that the datasets with the highest variance are the midsize ones. This is contrary to common belief that pretrained models have a greater advantage over models trained from scratch in data-constrained settings [47]. Figure 5a illustrates that the datasets with the highest variance in performance (where pretraining provides the biggest advantages) are those that are the most difficult (based on average accuracy). Lastly, we hypothesized that performance is positively correlated with sentence length. Figure 5d suggests that there is a positive correlation between average sentence length and performance, confirming our hypothesis.



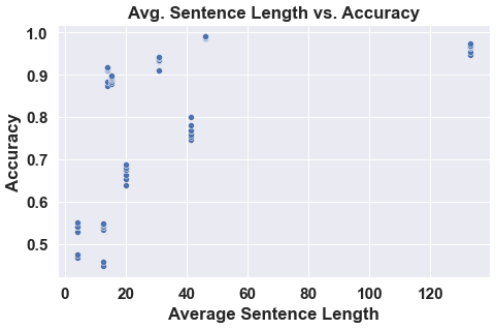
(a) Datasets which are hardest to learn have the greatest variance in performance.



(b) Variance in performance is greatest for midsize datasets, not small datasets.



(c) Variance in performance across models is greatest for GE and SST5.



(d) Model perfomance is positively correlated with sentence length.

Figure 5: **Dataset Attributes and Performance**. In (a), (b), and (c), the numbers are accuracy scores and the colors represent z-score.

3. Effects of Pretraining: For a researcher trying to better understand the effects of pretraining, exploring the impact of pretraining on robustness and model convergence is an interesting research direction [48, 49]. We demonstrate how the features of LBT make exploring the aforementioned relationships feasible. Inspired by prior work on the impact of pretraining on robustness and model convergence [48], we hypothesized that (i) pretrained models are more robust to adversarial attacks and that (ii) pretrained models should converge faster than models trained from scratch [49].

To test (i), we used the LBT TextAttack integration to compare the robustness of models to three different types of attack. The three attacks we used were DeepWordBug [50] (character insertion, swap, deletion, and substitution), PWWS [51] (synonym swap), and Input Reduction [52] (word deletion). As shown in Figure 4b, we see that the average successful attack rate is high for fine-tuned BERT models which suggests that these models are less robust to input perturbations. In contrast, RoBERTa and the RNN model have surprisingly low attack success rates. These results disprove our hypothesis and warrant further analysis.

To test (ii), we used the number of epochs elapsed until the best checkpoint as a proxy metric for model convergence. Figure 4a shows that some pretrained models do indeed converge fast (BERT and RoBERTa), while others (T5 and DistilBERT) are actually the slowest to converge.

5 Limitations and Conclusion

Limitations. We begin by acknowledging the limitations of LBT. First, the standardized training framework used to run experiments in LBT results in a trade-off between making fair comparisons on a limited model input space and making inaccurate comparisons on an unconstrained model input space. Second, while dataset curation is a key challenge in constructing benchmark studies, LBT doesn't currently provide tooling for curating robust and comprehensive evaluation datasets. Finally, we identify LBT's dependence on Ludwig for task, model, and training support as a potential drawback. Currently, Ludwig focuses on supervised models and does not support tasks such as question answering or summarization. However, Ludwig is a growing platform supported by an active developer community, so expanded support for new tasks is likely. Moreover, while Ludwig is

designed to be extensible to new models, doing so could be time consuming (take on the order of a few hours) and might serve as a potential bottleneck for users who want to spin up a benchmark study on a more expedient timeline.

Conclusion. In this work, we present LBT: an extensible toolkit for creating personalized model benchmark studies across a wide range of machine learning tasks, deep learning models, and datasets. We demonstrate how LBT helps value-driven communities more appropriately benchmark models by (i) providing configurable interface for creating custom benchmarks studies, (ii) implementing a standardized training framework that helps users study the tradeoffs and effects of the variables they care about by controlling for confounding variables, and (iii) providing access to a diverse set of evaluation metrics useful for multi-objective evaluation.

6 Ethical Considerations

We acknowledge that there are several ethical considerations pertaining to our work. Firstly, as a benchmarking toolkit, we make use of a variety of open-source datasets. In some cases, we have limited knowledge as to how the datasets were curated and if the data was collected in an ethical manner [53, 54]. Moreover, these datasets can contain several harmful biases (e.g., gender, race) that can be further propagated by models trained on their contents [53]. Another concern is data poisoning, where datasets are tampered with the intent of biasing a downstream trained model [55]. Secondly, LBT makes use of several pretrained language models. An abundance of recent work has highlighted a variety of biases that exist in these models [56, 57]. That being said, because the toolkit is modular, users have the agency to replace any datasets and models in their benchmark studies that they believe might have ethical issues. Moreover, LBT also provides the community with the necessary tools to compare models and datasets based on bias. We hope that the community will use LBT for these objectives.

References

[1] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

[2] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems, 2020.

[3] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[4] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

[5] Kawin Ethayarajh and Dan Jurafsky. Utility is in the eye of the user: A critique of nlp leaderboards. *arXiv preprint arXiv:2009.13888*, 2020.

[6] Anna Rogers. How the transformers broke nlp leaderboards, Jun 2019. URL https://hackingsemantics.xyz/2019/leaderboards/.

[7] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models matter, so does training: An empirical study of cnns for optical flow estimation, 2018.

[8] Robin M. Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley – benchmarking deep learning optimizers, 2021.

[9] Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R. Collins, Jeff Schneider, Barnabas Poczos, and Eric P. Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly, 2020.

[10] Matthias Aßenmacher and Christian Heumann. On the comparability of pre-trained language models, 2020.

[11] Jishnu Mukhoti, Pontus Stenetorp, and Yarin Gal. On the importance of strong baselines in bayesian deep learning. *arXiv preprint arXiv:1811.09385*, 2018.

[12] Tal Linzen. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages

5210–5217, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.465. URL https://www.aclweb.org/anthology/2020.acl-main.465.

[13] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[14] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning, 2020.

[15] John X Morris, Eli Lifland, Jin Yong Yoo, and Yanjun Qi. Textattack: A framework for adversarial attacks in natural language processing. *arXiv preprint arXiv:2005.05909*, 2020.

[16] Karan Goel, Nazneen Rajani, Jesse Vig, Samson Tan, Jason Wu, Stephan Zheng, Caiming Xiong, Mohit Bansal, and Christopher Ré. Robustness gym: Unifying the nlp evaluation landscape. *arXiv preprint arXiv:2101.04840*, 2021.

[17] Pengfei Liu, Jinlan Fu, Yang Xiao, Weizhe Yuan, Shuaicheng Chang, Junqi Dai, Yixin Liu, Zihuiwen Ye, and Graham Neubig. Explainaboard: An explainable leaderboard for nlp. *arXiv preprint arXiv:2104.06387*, 2021.

[18] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding, 2020.

[19] Douwe Kiela, Zhiyi Ma, Tristan Thrush, Somya Jain, Ledell Wu, Robin Jia, and Adina Williams. Dynaboard: Moving beyond accuracy to holistic model evaluation in nlp. https://ai.facebook.com/blog/dynaboard-moving-beyond-accuracy-to-holistic-model-evaluation-in-nlp/, 2021.

[20] Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. *Training*, 100(101):102, 2017.

[21] Stephen R Piccolo, Terry J Lee, Erica Suh, and Kimball Hill. Shinylearner: A containerized benchmarking tool for machine-learning classification of tabular data. *GigaScience*, 9(4): giaa026, 2020.

[22] Elasticsearch: Restful, distributed search & analytics — elastic. URL https://www.elastic.co/.

[23] Mengzhou Xia, Anjalie Field, and Yulia Tsvetkov. Demoting racial bias in hate speech detection. In *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, pages 7–14, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.socialnlp-1.2. URL https://www.aclweb.org/anthology/2020.socialnlp-1.2.

[24] Piero Molino, Yaroslav Dudin, and Sai Sumanth Miryala. Ludwig: a type-based declarative deep learning toolbox. *arXiv preprint arXiv:1909.07930*, 2019.

[25] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

[26] Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, and Brown. Text classification algorithms: A survey. *Information*, 10(4):150, Apr 2019. ISSN 2078-2489. doi: 10.3390/info10040150. URL http://dx.doi.org/10.3390/info10040150.

[27] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

[28] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning based text classification: A comprehensive review, 2021.

[29] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, 2017.

[30] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2016.

[31] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 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, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

[32] Emily Dinan, Angela Fan, Ledell Wu, Jason Weston, Douwe Kiela, and Adina Williams. Multi-dimensional gender bias classification, 2020.

[33] Byron C. Wallace, Do Kook Choe, Laura Kertz, and Eugene Charniak. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–516, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-2084. URL https://www.aclweb.org/anthology/P14-2084.

[34] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. Goemotions: A dataset of fine-grained emotions, 2020.

[35] Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. Social bias frames: Reasoning about social and power implications of language, 2020.

[36] 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.

[37] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[38] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[39] 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.

[40] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

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

[42] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[43] Yoon Kim. Convolutional neural networks for sentence classification, 2014.

[44] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[45] Tim Head, MechCoder, Gilles Louppe, Iaroslav Shcherbatyi, fcharras, Zé Vinícius, cmmalone, Christopher Schröder, nel215, Nuno Campos, Todd Young, Stefano Cereda, Thomas Fan, rene rex, Kejia (KJ) Shi, Justus Schwabedal, carlosdanielcsantos, Hvass-Labs, Mikhail Pak, SoManyUsernamesTaken, Fred Callaway, Loïc Estève, Lilian Besson, Mehdi Cherti, Karlson Pfannschmidt, Fabian Linzberger, Christophe Cauet, Anna Gut, Andreas Mueller, and Alexander Fabisch. scikit-optimize/scikit-optimize: v0.5.2, mar 2018. URL https://doi.org/10.5281/zenodo.1207017.

[46] Aysu Ezen-Can. A comparison of lstm and bert for small corpus, 2020.

[47] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

[48] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721. PMLR, 2019.

[49] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4927, 2019.

[50] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018.

[51] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097, 2019.

[52] Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. *arXiv preprint arXiv:1804.07781*, 2018.

[53] Emily M Bender and Batya Friedman. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604, 2018.

[54] Jack Bandy and Nicholas Vincent. Addressing" documentation debt" in machine learning research: A retrospective datasheet for bookcorpus. *arXiv preprint arXiv:2105.05241*, 2021.

[55] Hongyan Chang, Ta Duy Nguyen, Sasi Kumar Murakonda, Ehsan Kazemi, and Reza Shokri. On adversarial bias and the robustness of fair machine learning. *arXiv preprint arXiv:2006.08669*, 2020.

[56] Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. Language (technology) is power: A critical survey of" bias" in nlp. *arXiv preprint arXiv:2005.14050*, 2020.

[57] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

[58] Giampaolo Rodola. Psutil package: a cross-platform library for retrieving information on running processes and system utilization. *Google Scholar*, 2016.

[59] Python module for getting the gpu status from nvida gpus. URL `https://pypi.org/project/GPUtil/`.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[61] Sercan O. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning, 2020.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.

[63] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021.

[64] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[65] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining, 2019.

[66] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[67] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[68] 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.

[69] Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. Flaubert: Unsupervised language model pre-training for french, 2020.

[70] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. Camembert: a tasty french language model. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*,

2020. doi: 10.18653/v1/2020.acl-main.645. URL http://dx.doi.org/10.18653/v1/2020.acl-main.645.

[71] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019.

[72] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale, 2020.

[73] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[74] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[75] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[76] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[77] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

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]
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
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 (e.g. for benchmarks)...
   (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] See Section A.4
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and A.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (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] See Section 4
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 citep the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See page 1 for link to LBT code repository
   (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]