Enhanced BERT Adaptation: Ensembling LoRA Models for Improved Fine-Tuning

Stanford CS224N Default Project

Denis Tolkunov
Stanford University
tolkunov@stanford.edu

Abstract

Transfer learning has become an indispensable technique for addressing a wide range of downstream NLP tasks using a single generic pre-trained model. Given the specialized nature of most downstream tasks, obtaining quality training data is often challenging, making it difficult to train large models from scratch. One effective approach is Parameter-Efficient Fine-Tuning (PEFT). PEFT enhances the performance of large AI models while optimizing resources such as time, energy, and computational power. It achieves this by fine-tuning a small subset of key parameters while maintaining most of the pretrained model's structure. In this project, we thoroughly explore the Low-Rank Adaptation (LoRA) approach and propose two ensembling methods: Omni-LoRA and DiCor-LoRA. By combining different LoRA models that capture various aspects of the data, we effectively address heterogeneous tasks. DiCor-LoRA achieves state-of-the-art performance, securing an overall score of 0.798 on the test dataset ranking us 4^{th} on the leaderboard.

1 Key Information to include

- Mentor: Yuan Gao
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

Human beings effortlessly plan, understand, and complete a variety of tasks through linguistic communication, making language systems a robust and fascinating problem space. Initial computational attempts to replicate human language abilities for multitask settings relied on rule-based systems crafted by domain experts. However, these systems lacked linguistic robustness, were costly to maintain, and failed to generalize across different tasks due to inadequate linguistic representation (Jurafsky and Martin (2008)). The advent of Transformer-based models, such as the Bidirectional Encoder Representations from Transformers (BERT), has revolutionized natural language processing (NLP) by creating powerful word embeddings that capture complex linguistic relationships (Devlin et al. (2019)). The current paradigm of fine-tuning these foundation models for specific tasks significantly improves task accuracy but often at the cost of degrading performance on other tasks. This issue, known as catastrophic forgetting (McCloskey and Cohen (1989)), highlights the challenge of maintaining high performance across multiple tasks when using separate model weights. In this project, we address the NLP challenge of multi-task fine-tuning by leveraging BERT to classify sentence sentiment, predict paraphrase pairs, and detect semantic textual similarity while sharing a single set of BERT embeddings. Our approach aims to create a robust multi-task model that mitigates catastrophic forgetting by fine-tuning BERT on specific datasets and employing advanced techniques to generate sentence embeddings that encapsulate semantic essence. This strategy enhances BERT's

capability to excel in various NLP tasks, demonstrating that a single, fine-tuned model can effectively handle multiple tasks with high performance.

3 Related Work

Parameter-Efficient Fine-Tuning (PEFT) methods are designed to mitigate the high costs associated with fine-tuning large-scale models by training only a small subset of parameters necessary for adapting to downstream tasks. These methods can be categorized into three main types. The first category, Adapter-based methods, introduces additional trainable modules into the original frozen backbone. For instance, (Houlsby et al. (2019)) propose adding fully-connected networks after attention and FFN layers in Transformer, (He et al. (2021)) suggest integrating these modules in parallel to enhance performance. The second category, Prompt-based methods, adds extra soft tokens or soft prompts to the initial input, focusing on fine-tuning these trainable vectors (Lester et al. (2021)). These approaches often face challenges due to their sensitivity to initialization, impacting their overall effectiveness. Both adapter-based and prompt-based methods typically result in increased inference latency compared to the baseline model.

The third category includes LoRA (Low-Rank Adaptation) methods, which are particularly notable for not adding any extra inference burden. LoRA (Hu et al. (2022)) and its variants apply low-rank matrices to approximate weight changes during fine-tuning and can merge with pre-trained weights before inference. Low-rank adaptation leverages the principle that the adaptation process is intrinsically low-dimensional (Aghajanyan et al. (2021)), allowing significant model changes to be represented with relatively few parameters. However, reducing the rank can introduce challenges related to generalization errors for specific tasks, compared to full-parameter fine-tuning. Recent research by Ren et al. (2024) has proposed mini-ensemble low-rank adapters, which utilize fewer trainable parameters while maintaining a higher rank and yielding promising results by having each adapter learn different dimensions of a hidden state.

In this project, we implement and evaluate three approaches: a single LoRA model with hyperparameters specifically tuned for each task, and two novel ensemble models, Omni-LoRA and DiCor-LoRA. These innovative models combine LoRA configurations with varying ranks and scaling parameters to effectively capture diverse aspects of the data, thereby aiming to improve overall model performance.

4 Approach

4.1 Baselines

We implemented two baselines to evaluate our approach. The first baseline keeps all original BERT parameters frozen, except for the task-specific classification and regression heads, aiming to assess the pre-trained BERT model's capability on our downstream tasks with minimal adjustments. The second baseline involves three separately fine-tuned BERT models, each dedicated to one task, to estimate BERT's potential upper-bound performance. For all tasks, we used a linear layer with a dropout probability of 0.3 as the head. Specifically, the STS task employed MSE loss, the sentiment task used softmax activation with cross-entropy loss, and the paraphrase task applied sigmoid activation with binary cross-entropy loss. The first baseline's linear layer was trained with a learning rate of 10^{-3}, while the second baseline (full fine-tuning) used a learning rate of 10^{-5}, with both trained for 10 epochs (see Table 1). Throughout the project, we utilized the Adam optimizer with decoupled weight decay regularization (Loshchilov and Hutter (2017)).

Semantic Textual Similarity and Paraphrase Detection require evaluating two input sequences together, posing a challenge for BERT, which is typically designed for single sequences. To address this, we use the cross-encoding method from (Devlin et al. (2019)). This approach merges the input sequences with a [SEP] token before processing them through BERT, allowing the model to consider both sentences simultaneously. A classification or regression layer is then applied to the combined output. This method captures stronger inter-sentence nuances through its attention mechanism, making it effective for sentence-pair tasks (Figure 2).

4.2 Low-Rank Adaptation

The LoRA method (Hu et al. (2022)) operates on the principle that during fine-tuning, the changes to the model's weights are confined to a low-dimensional space, despite the model possessing a large number of parameters overall. This concept is supported by observations Aghajanyan et al. (2021) that highly optimized models typically inhabit a low-intrinsic dimensional manifold. More concretely, if the pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, and ΔW its accumulated gradient update during fine-tuning, then the update is restricted to being a matrix of lower rank

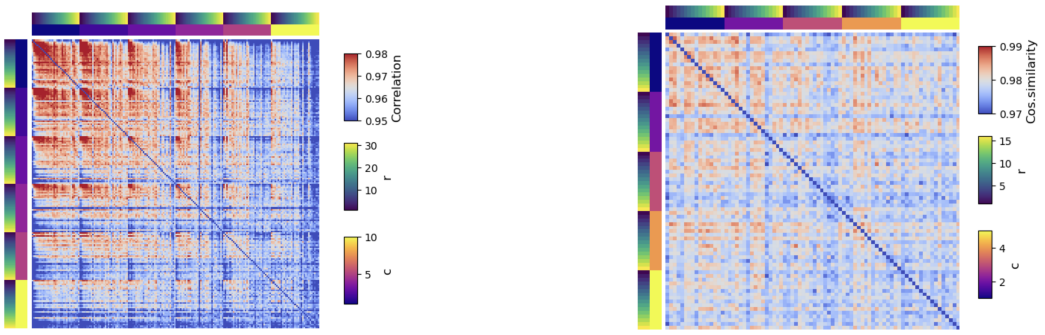
$$W_0 + \Delta W = W_0 + BA, \tag{1}$$

where $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$. Here, the rank r is generally much smaller than the rank of the full pre-trained weight matrix W_0. At the start of training, the matrices A and B are initialized such that $A_{ij} \sim N(0, \sigma^2)$ and $B_{ij} = 0$, ensuring $\Delta W = 0$ at the beginning. Throughout the training process, W_0 remains unchanged while A and B include tunable parameters. ΔW is scaled by $c = \frac{\alpha}{r}$, where α is a constant in r. In essence, the scaling factor c controls the contribution of the low-rank adaptation to the overall model. Specifically, it scales the output of the low-rank layers before they are added to the original model's weights. This allows fine-tuning the influence of the adaptation, ensuring that the model balances between preserving original parameters and incorporating the changes introduced by the low-rank adaptation.

We implemented a LoRA model and applied it to the Query and Value components of the transformer. Additionally, we attempted to apply LoRA to the Key component, but this did not result in an improvement in model accuracy.

4.3 LoRA Ensembles

To the best of our knowledge, the method of ensembling LoRA models as we have implemented it is an **original contribution**. The effectiveness of ensemble methods can be understood through the concepts of variance, bias, and error correlation among the base models. The main idea is to combine multiple models to achieve more accurate and robust predictions than those made by individual models. This requires the models in the ensemble to be diverse and have low correlation. When multiple models are combined, the overall error can be reduced if the errors of these models are uncorrelated. This happens because individual errors can offset each other. However, if the models are highly correlated, they tend to make the same errors, reducing the benefits of ensembling.



(a) STS (b) SST-5

Figure 1: Visualization of model diversity: (a) Heatmap showing the pairwise Pearson correlation coefficients between 186 LoRA models trained on STS data, calculated using logits on the STS development set. (b) Heatmap depicting the cosine similarity between predictions from 80 models trained on SST-5 data, calculated on the SST-5 development dataset. The rank r and scaling parameter c of each model are color-coded in the side bars. The diversity of the models increases with the rank r and the scaling parameter c in both cases.

The prediction of the ensemble can be expressed as: $\hat{y}_{ens} = \frac{1}{N}\sum_{i=1}^{N} \hat{y}_i$, where $\hat{y}_i$ represents the prediction from the i-th model, N is the number of models in the ensemble. The variance of the ensemble prediction is generally given by $Var(\hat{y}_{ens}) = \frac{1}{N^2}\left(\sum_{i=1}^{N} Var(\hat{y}_i) + \sum_{i\neq j} Cov(\hat{y}_i, \hat{y}_j)\right)$.

Assuming equal variance for each LoRA model's prediction, $Var(\hat{y}_i) = \sigma^2$ and a common correlation ρ among the model predictions, the variance of the ensemble prediction simplifies to:

$$Var(\hat{y}_{ens}) = \frac{1}{N}\sigma^2 + \frac{N-1}{N}\rho\sigma^2. \tag{2}$$

The goal is to maximize diversity of the models, i.e. to minimize ρ. The variance of ensemble predictions can be mitigated by averaging the outputs of multiple models. However, the effectiveness of this ensembling technique is diminished when there is a high correlation between the individual model predictions. Our methodology for constructing the LoRA ensemble involves a two-step process: initially, we select the top-performing models based on prediction accuracy. Subsequently, we enhance diversity by choosing a subset of models that exhibit minimal correlation with each other.

The variance reduction concept in ensemble models applies similarly to majority voting ensembles, particularly when considering correlated models. In a majority voting ensemble, each model votes for a class label, and the class with the majority votes is chosen as the final prediction. When models are correlated, the effective error rate p_{eff} can be approximated as $p_{eff} = \rho p + (1-\rho)p^2$, where p is the individual model error rate, and ρ is the average correlation between model's errors. The probability that the ensemble makes an incorrect prediction is then $P_{ens} = \sum_{k=\lceil \frac{N}{2} \rceil}^{N} \binom{N}{k} p_{eff}^k (1-p_{eff})^{N-k}$. From this formula we see that by reducing correlation ρ and increasing the number of models the ensemble achieves lower error rates, enhancing overall prediction accuracy (Hastie et al. (2004)).

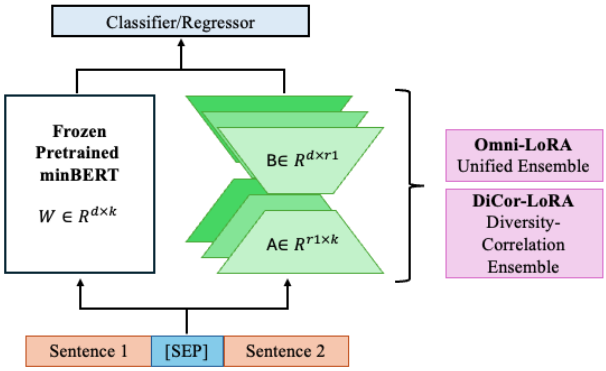


Figure 2: Ensembling LoRA models involves combining models with varying ranks r and scaling parameters $c = \alpha/r$ to capture different aspects of the data. In Omni-LoRA, we aggregate all generated models (all available models) into one ensemble. In DiCor-LoRA, we optimize the ensemble by selecting a small subset of models based on maximizing diversity and correlation with the true predictions. We employ a cross-encoding strategy for semantic textual similarity and paraphrase detection tasks.

During hyperparameter tuning, we generated 186 LoRA models for the STS task with varying ranks and scaling parameters, 80 models for the SST task, and 4 models for the paraphrase task. The paraphrase task has the fewest models due to the computationally expensive training process on the Quora dataset. We calculated the pairwise correlation between the predictions of the STS models and the cosine similarities between the predictions of the SST-5 and Quora models (Figure 1). We observed a degree of diversity among the models, which increases with rank and scaling parameters, as expected, since the complexity of the models also increases.

We hypothesize that these LoRA models capture different aspects of the data. Motivated by these observations, we ensembled the models using logits averaging for the STS task and majority voting for the SST and paraphrase tasks. In cases of ties in the majority vote ensemble, we favored the model with the highest rank. We refer to this ensemble model as Omni-LoRA. This approach resulted in improved accuracy and more robust predictions.

However, the full ensemble contains a significant number of correlated models, which is suboptimal (Figure 1). To address this, we implemented the DiCor-LoRA ensemble model, which includes a small number (4-11) of diverse models, each with high predictive accuracy. This approach aims to enhance the ensemble's overall performance by reducing redundancy and improving model diversity.

The ensemble models are selected through a two-step process: first, we identify the models with the highest predictive accuracy. In the second step, we select models based on their lowest correlation for the STS task or lowest cosine similarity for SST-5 and Quora predictions.

5 Experiments

5.1 Data

SST-5: Stanford Sentiment Treebank consists of single sentences from movie reviews each with the label of negative (0), somewhat negative (1), neutral (2), somewhat positive (3), or positive (4). We have 8,544 train, 1,101 development, and 2,210 test examples.

Quora: The Quora dataset consists of question pairs with labels indicating whether instances are paraphrases of one another. We have 141,506 train, 20,215 development, and 40,431 test examples.

STS: The SemEval STS Benchmark dataset consists of sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). We have 6,040 train, 863 development, and 1,725 test examples.

5.2 Evaluation method

We evaluate model performance using accuracy for sentiment analysis and paraphrase detection, while Pearson correlation is used to assess performance on the semantic textual similarity task.

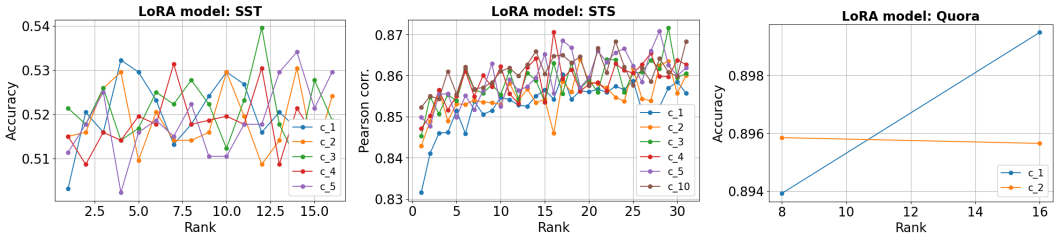
5.3 Experimental details

We trained all the models using a learning rate of 10^{-4}. After conducting several small experiments with different learning rate schedules, we observed no improvement in model accuracy, leading us to use a fixed learning rate throughout the project.

Prior to conducting a large-scale grid search for parameter optimization, we performed preliminary experiments to determine the optimal number of training epochs by observing when improvements in development accuracy ceased. We found the optimal number of epochs to be 10 for the STS task, 20 for the Quora task, and 20 for the SST task, with a batch size of 8 used consistently.

LoRA adaptation was applied to the Query and Value components of the transformer. We also attempted to apply LoRA to the Key component, but this did not enhance model accuracy and only increased the number of training parameters.

Our custom implementation of the AdamW optimizer (Adam with decoupled weight decay regularization) performed comparably to the PyTorch implementation. However, since the PyTorch version ran slightly faster, we opted to use it.



(a) SST-5 (b) STS (c) Quora

Figure 3: Development set accuracy of LoRA models trained with different ranks r and scaling parameters c. (a) The SST-5 model was trained for 20 epochs. (b) The STS model was trained for 10 epochs. (c) The paraphrase detection model was trained for 20 epochs. A learning rate of 10^{-4} and a batch size of 8 were used for all models.

The grid search for LoRA model ranks and scaling parameters revealed the following optimal settings for achieving the best accuracy (or Pearson correlation) on the development datasets: LoRA SST: $r = 12$, $c = 3$, LoRA Quora: $r = 16$, $c = 1$; LoRA STS: $r = 29$, $c = 3$.

The DiCor-LoRA SST ensemble model consists of 11 models selected for their maximal prediction accuracy and diversity, as measured by cosine similarity, with the following rank and scaling parameters (r, c): (12, 3), (14, 5), (4, 1), (7, 4), (12, 4), (5, 1), (13, 5), (8, 3), (15, 3), (3, 5), (9, 1).

The DiCor-LoRA Quora ensemble model includes 4 models optimized for prediction accuracy and diversity, measured by cosine similarity, with these parameters (r, c): (8, 1), (8, 2), (16, 1), (16, 2).

The DiCor-LoRA STS ensemble model comprises 10 models chosen for their high prediction accuracy and diversity, as indicated by cosine similarity, with the following parameters (r, c): (29, 3), (28, 5), (16, 4), (17, 5), (23, 10), (31, 10), (18, 5), (21, 10), (24, 5), (21, 5).

5.4 Results

The performance of the models on the development set is presented in Table 1. The LoRA (best) model shows significantly higher accuracy compared to the Frozen BERT baseline. However, since the hyperparameters were tuned on the same set, there is a likelihood of overfitting to the development dataset. This is particularly evident in the SST task, where the accuracy's dependence on the r and c parameters is highly irregular (Figure 3.a). This is further confirmed when comparing the LoRA (best) SST accuracy to the more robust Omni-LoRA model, where accuracy decreases from 0.54 to 0.53, while accuracies for the Quora and STS tasks show slight increases.

The DiCor-LoRA model achieves significant accuracy improvements across all three tasks compared to both the Omni-LoRA and LoRA (best) models. This supports the notion that a small ensemble of accurate and diverse models outperforms a large ensemble of models.

Table 1: Development accuracy results for various models. The results for Finetuned BERT correspond to three distinct BERT models, each fine-tuned for specific tasks. LoRA (best) refers to a single model with optimized hyperparameters, potentially overfitting to the development data. Omni-LoRA represents a brute force ensemble combining 4-186 different LoRA models. DiCor-LoRA is an ensemble of selectively chosen 4-11 models, selected based on maximizing diversity and correlation.

		SST-5 (dev.)	Quora (dev.)	STS (dev.)	Overall score
Baseline	Frozen BERT	0.395	0.697	0.526	0.618
	Finetuned BERT	0.543	0.883	0.884	0.789
PEFT	LoRA (best)	0.540	0.898	0.872	0.791
	Omni-LoRA	0.530	0.905	0.874	0.791
	DiCor-LoRA	**0.544**	**0.905**	**0.885**	**0.797**

The performance of the DiCor-LoRA model on the test dataset is shown in Table 2. The results are comparable to those observed on the development set, demonstrating the robustness of our ensembling approach across different tasks.

Table 2: Model performance on the test set.

	SST-5 (test)	Quora (test)	STS (test)	Overall score
DiCor-LoRA	0.550	0.904	0.882	**0.798**

6 Analysis

The behavior of the development accuracy as a function of the model rank (Figure 3) provides valuable insights into the data characteristics related to each task. For instance, the SST accuracy plateaus after rank 5, whereas the STS accuracy plateaus around ranks 15-20, indicating that the

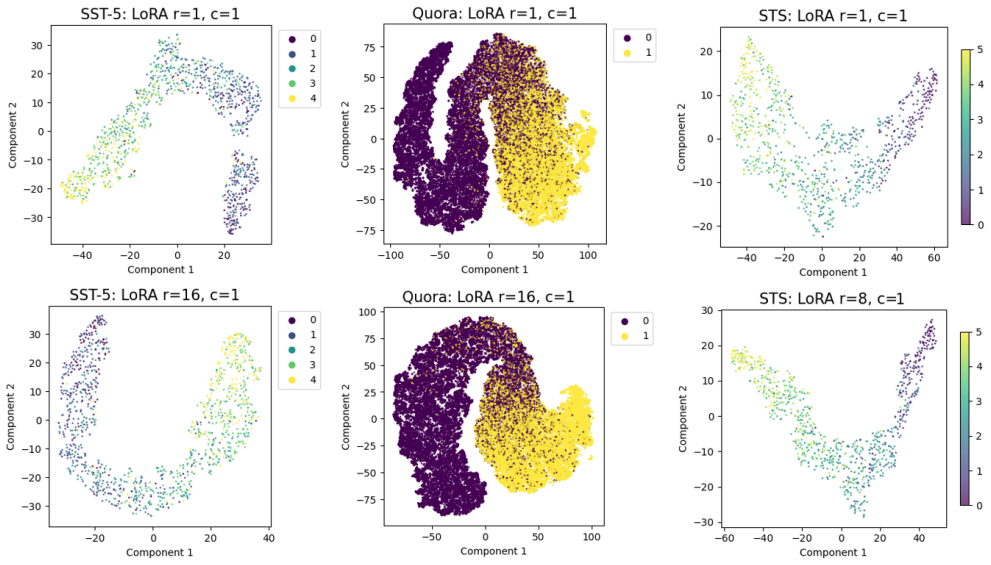


Figure 4: t-SNE visualizations of the LoRA embeddings for SST-5, Quora, and STS are presented to compare models with low and high ranks.

semantic textual similarity task requires greater model capacity compared to the sentiment analysis task. While we lack sufficient simulation data to draw definitive conclusions about the Quora dataset, it is evident that the optimal rank is no less than 16, and increasing the scaling factor does not enhance accuracy at this rank.

Using t-SNE, we visualized the LoRA embeddings for both lower and higher ranks (Figure 4). In general, increasing the rank of the model tends to produce smoother embeddings. This suggests that higher-rank models learn better representations of the data, effectively capturing the underlying structure. Notably, the embeddings for the SST model with rank 16 appear much smoother compared to those with rank 1. This is intriguing because, for the STS model, higher ranks show a more erratic behavior in accuracy (see Figure 3.a). This likely indicates the presence of noise in the STS dataset labels. Although the samples are consistently ordered with their labels across the embedding manifold, there is significant overlap between neighboring categories, which may lead to lower accuracy for the STS model compared to others. For the Quora models, different ranks seem to capture various aspects of the data, resulting in relatively smooth but distinctively shaped embedding manifolds.

7 Conclusion

In this project, we implemented three models: LoRA—a standard low-rank adaptation method—and two ensemble models. Omni-LoRA, a brute force ensemble, combines models with varying ranks and scalings. Our best-performing ensemble model, DiCor-LoRA, is based on a carefully selected set of models chosen for their accuracy and diversity. DiCor-LoRA achieved an overall score of 0.798 on the test set. We believe that ensembling is a promising approach to enhance the effectiveness of low-rank adaptation models. As future avenues for this research, we could investigate different ensembling methodologies such as weighted averaging, boosting, or Bayesian model averaging.

8 Ethics Statement

In our project focusing on sentiment analysis, paraphrase detection, and semantic textual similarity tasks, there are several ethical concerns and potential societal risks. One significant concern is algorithmic bias, particularly in sentiment analysis, where biased training data can lead to discriminatory outcomes, reinforcing societal stereotypes or prejudices. Additionally, in paraphrase detection and semantic textual similarity tasks, there is a risk of unintended disclosure of sensitive information,

especially if the models are trained on data containing personal or confidential content. These issues could exacerbate existing inequalities and privacy concerns, ultimately leading to mistrust in AI systems. To address these ethical challenges and societal risks, several practical strategies can be implemented. Firstly, conducting thorough bias assessments on training data and implementing techniques such as data augmentation, adversarial training, or fairness-aware learning algorithms can help mitigate algorithmic bias in sentiment analysis. For paraphrase detection and semantic textual similarity tasks, anonymization and aggregation of sensitive data during model training can reduce the risk of privacy breaches. Additionally, adopting transparency measures, such as providing explanations for model predictions and documenting data collection and processing methods, fosters trust and accountability in AI systems. Moreover, involving diverse stakeholders, including ethicists, domain experts, and impacted communities, in the design and evaluation process can help identify and address potential ethical concerns proactively.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. 2004. The elements of statistical learning: Data mining, inference, and prediction. *Math. Intell.*, 27:83–85.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *ArXiv*, abs/2110.04366.

N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Daniel Jurafsky and James Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. *ArXiv*, abs/2402.17263.