CURLoRA: Stable LLM Continual Fine-Tuning and Catastrophic Forgetting Mitigation

Muhammad Fawi

Abstract

This paper introduces CURLoRA, a novel approach to fine-tuning large language models (LLMs) that leverages CUR matrix decomposition in the context of Low-Rank Adaptation (LoRA). Our method addresses two critical challenges in LLM fine-tuning: mitigating catastrophic forgetting during continual learning and reducing the number of trainable parameters. We propose a unique modification to the CUR decomposition process, utilizing inverted probabilities for column and row selection which acts as an implicit regularization, and initializing the U matrix as a zero matrix, and only fine-tuning it. We demonstrate through experiments on multiple datasets that CURLoRA outperforms standard LoRA in mitigating catastrophic forgetting. It maintains model stability and performance across tasks while significantly reducing the number of trainable parameters. Our results show that CURLoRA achieves superior accuracy and perplexity scores compared to LoRA upon continual fine-tuning, particularly in scenarios with limited data.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing, demonstrating remarkable capabilities across a wide range of tasks [1]. However, fine-tuning these large models for specific tasks requires a lot of computational resources making it challenging to adapt these models efficiently, especially when working with limited datasets and in resource-constrained environments. [2]. Parameter-Efficient Fine-Tuning (PEFT) Methods have gained a lot of attention because they make fine-tuning large models accessible and possible. [3]

Low-Rank Adaptation (LoRA) [4] has emerged as an efficient PEFT method, enabling fine-tuning large language models on custom tasks while decreasing the number of trainable parameters hence requiring less resources. LoRA works by decomposing pre-trained weight matrices into low-rank matrices and fine-tune these ones instead of the original matrix. Although LoRA has proven to be very excellent and promising, it still faces challenges with catastrophic forgetting. Catastrophic forgetting in LLMs is a critical issue where the model loses previously acquired knowledge when fine-tuned on new tasks [5]. It occurs due to the overwriting of previously learned (pre-trained) weights during the fine-tuning process. In LoRA, this often happens as the adapted output can significantly deviate from the original:

$$y = xW + xW_{adapted} = x(W + AB) \tag{1}$$

where $W \in \mathbb{R}^{m\times n}$ is the original weight matrix, and AB is the low-rank update from multiplying $A \in \mathbb{R}^{m\times r}$ by $B \in \mathbb{R}^{r\times n}$ where $r < n$.

This work introduces CURLoRA, a novel approach that applies low-rank adaptation (LoRA) to pre-trained weight matrices using CUR matrix decomposition [6] instead of random initiation of the low-rank A or B matrices. We propose a unique modification to the CUR decomposition process and demonstrate its effectiveness in mitigating catastrophic forgetting while also reducing the number of trainable parameters. While LoRA successfully reduces computational costs by decomposing weight updates into low-rank matrices, it still suffers from catastrophic forgetting. CURLoRA leverages CUR decomposition with inverted probabilities and initiating U matrix as zero to further mitigate this issue.

2 Related Work

2.1 Catastrophic Forgetting

Catastrophic forgetting is a big challenge in machine learning, particularly in the context of continual learning [5]. Various approaches have been proposed to address this issue:

- **Elastic Weight Consolidation (EWC)** [7] uses Fisher information to measure the importance of parameters and selectively slow down learning on important parameters.
- **Progressive Neural Networks** [8] propose to freeze the network trained on previous tasks and add lateral connections to new columns for new tasks.
- **Memory-based approaches like Experience Replay** [9] store and replay examples from previous tasks during training on new tasks.

2.2 Efficient Fine-tuning of Large Language Models

As LLMs have grown in size, efficient fine-tuning methods have become crucial:

- **Adapter layers** [10] introduce small trainable modules between layers of a pre-trained model.
- **Low-Rank Adaptation (LoRA)** [4] decomposes weight updates into low-rank matrices, significantly reducing the number of trainable parameters.
- **Prefix-tuning** [11] prepends trainable continuous prompts to the input, allowing for task-specific adaptations.

2.3 CUR Matrix Decomposition

CUR decomposition has been applied in various domains for its interpretability and efficiency:

- In data analysis, CUR has been used for feature selection and dimensionality reduction [6].
- In scientific computing, CUR has been applied to accelerate large-scale matrix computations [12].
- In machine learning, CUR has been explored for model compression and interpretation [13].

However, to the best of our knowledge, CUR decomposition has not been previously applied to the problem of fine-tuning large language models or addressing catastrophic forgetting in this context.

3 Background on CUR Decomposition

CUR decomposition is a matrix factorization technique that approximates a matrix A as the product of three matrices: C, U, and R. Unlike Singular Value Decomposition (SVD), CUR decomposition uses actual columns and rows from the original matrix, making it more interpretable.[6].

Given a matrix $A \in \mathbb{R}^{m \times n}$, CUR decomposition approximates A as:

$$A \approx CUR \tag{2}$$

where:

- $C \in \mathbb{R}^{m \times c}$ consists of c columns of A
- $R \in \mathbb{R}^{r \times n}$ consists of r rows of A
- $U \in \mathbb{R}^{c \times r}$ is a small matrix that ensures CUR is close to A

The columns and rows are typically chosen based on their statistical leverage scores.[12] Leverage scores indicate the importance of columns and rows in representing the original matrix. High leverage scores identify influential columns and rows, while low scores identify less critical ones.

4 This Work

In this section, we present CURLoRA, our novel approach to fine-tuning large language models that leverages a modified CUR matrix decomposition to mitigate catastrophic forgetting. We provide a detailed mathematical formulation of the approach, analyze it theoretically, and explain how it addresses the challenge of catastrophic forgetting upon continual learning.

4.1 CURLoRA

The core idea is to decompose the pre-trained weight matrices using a modified CUR approach and then fine-tune only the U matrix. This approach constrains the parameter space of possible adaptations keeping the fine-tuned parameters as small as possible to keep $\|W_{\text{adapted}} - W\|_F$ close to the original weight matrix frobenius norm ($\|W\|_F$) i.e. $W + W_{\text{adapted}}$ is so close to W to avoid the deviation of the adapted output.

4.2 Mathematical Formulation

Given a weight matrix $W \in \mathbb{R}^{m \times n}$, we first compute the probability of each column:

$$p_j = \frac{\|W_{:j}\|_2^2}{\|W\|_F^2} \tag{3}$$

where $W_{:j}$ is the j-th column of W, while $\|\cdot\|_2^2$ denotes the square of the L2 norm of the column and $\|\cdot\|_F^2$ denotes the square of the Frobenius norm of W. This will give us the probability of each column. For instance, if W has three columns with norms 2, 3, and 5, the probabilities are 4/38, 9/38, and 25/38 respectively.

We then invert these probabilities:

$$\tilde{p}_j = \frac{1/p_j}{\sum_{i=1}^{n} 1/p_i} \tag{4}$$

where $\tilde{p}_j$ is the inverted probability of the j-th column of W. The same steps are followed for rows. Inverted probabilities are used to sample columns and rows with lower leverage scores, which implicitly regularize the model and limit the magnitude of fine-tuning adjustments.

Then, we sample r columns and rows, where $r < n$, according to these inverted probabilities to construct C and R, which will always be fixed, with columns and rows with lower original probabilities. This trick plays a major role in the approach as it serves two purposes:

- It acts as a form of regularization, preventing the model from overfitting or moving too much towards the task and limiting the adaptation of the U matrix stopping it from growing so big in magnitude.
- It preserves the model's original behavior by focusing adaptations on less influential parts of the weight matrix. In addition, since C and R contain actual columns and rows from the original matrix, they contribute to the stability of the fine-tuning process.

In contrast to random initialization using Kaiming-uniform or Gaussian for weight A and zeros for weight B [4], which ensures starting from the base model, AB is zero, while doesn't limit the increase in AB, or SVD-based initialization [14], which may introduce higher initial values and greater variability, the U matrix is initialized as a zero matrix to ensure the fine-tuning process starts from the base configuration, preventing large initial deviations and contributing to model stability during the process:

$$U_{\text{init}} = 0 \tag{5}$$

During fine-tuning, we update only the U matrix, keeping C and R fixed as they play a crucial role in ensuring the stability of the process by limiting the increase of U:

$$W_{\text{adapted}} = CUR \tag{6}$$

4.3 Theoretical Analysis of Catastrophic Forgetting Mitigation

To understand how CURLoRA helps mitigate catastrophic forgetting, we analyze its properties mathematically:

4.3.1 Parameter Space Constraint

In CURLoRA, we decompose the original weight matrix W as:

$$W \approx CUR \tag{7}$$

During fine-tuning, we're optimizing:

$$W_{\text{adapted}} = C(U + \Delta U)R \tag{8}$$

where ΔU represents the changes made to U during fine-tuning. By constraining the updates to the subspace defined by C and R, CURLoRA limits drastic changes, thereby preserving the model's original knowledge.

4.3.2 Implicit Regularization

By initializing U as a zero matrix, and C and R with columns and rows of low weight values, the ones with lower probabilities, we provide an implicit regularization where C and R will always limit the unnecessary increase of U. This can be seen as adding a regularization term to the loss function quantified by the norm of the matrix U that is aimed to be kept small:

$$L_{\text{CURLoRA}}(\theta) = L_{\text{task}}(\theta) + \|U\|_F \tag{9}$$

where $\|U\|_F$ is the Frobenius norm of the U matrix that is being fine-tuned. This implicit regularization term encourages the model to keep the changes small. For instance, if U is initially zero, this term will push the fine-tuning process to make only necessary adjustments, preventing overfitting and excessive reliance on the fine-tuned parameters.

4.3.3 Reduced Interference

During fine-tuning, W is fixed, so the variable gradient flows through W_{adapted}, which is itself updated through U as C and R are fixed. Considering the gradients of the loss L with respect to the parameters, we can express the gradient of the loss with respect to W_{adapted} as follows:

$$\frac{\partial L}{\partial W_{\text{adapted}}} = C\left(\frac{\partial L}{\partial U}\right)R \tag{10}$$

This means that the gradient of the loss with respect to W_{adapted} is dependent on the gradients with respect to U scaled by the fixed matrices C and R. By projecting the gradients onto the subspace defined by C and R, the updates to W_{adapted} are constrained. This means that changes during fine-tuning are less likely to interfere with the model's ability to perform the original task, potentially reducing interference with directions important for the original task.

4.3.4 Reduced Degree of Freedom

If $W \in \mathbb{R}^{m \times n}$ and we use a rank-k adaptation, then:

- Full fine-tuning has mn degrees of freedom
- LoRA has $k(m + n)$ degrees of freedom
- CURLoRA has only k^2 degrees of freedom

This significant reduction in degrees of freedom inherently limits how far the model can stray from its original configuration.

4.3.5 Stability Analysis

We can analyze the stability of the adapted and fine-tuned weights and how its change is bounded using the fact that the change that happens to original W is W_{adapted}:

$$\Delta W = W_{\text{fine-tuned}} - W = W + W_{\text{adapted}} - W = W_{\text{adapted}} \tag{11}$$

To quantify this change, we can use the Frobenius norm, $||W_{\text{adapted}}||_F$. By utilizing the submultiplicativity property of the Frobenius norm, we can say that the growth of W_{adapted} is controlled through the norms of C, U, and R:

$$||W_{\text{adapted}}||_F = ||CUR||_F \leq ||C||_F ||U||_F ||R||_F \tag{12}$$

This equation ensures that the Frobenius norm of the adapted weight matrix W_{adapted} has an upper bound. Since C and R are fixed and U starts at zero, the fine-tuning process focuses on minimizing W_{adapted}. As a result, the adaptation remains stable and the model preserves its original knowledge while allowing for necessary adjustments.

Empirical results (see Section 7) demonstrate that the Frobenius norm of W_{adapted} remains bounded across multiple tasts, validating the theoretical stability analysis.

4.4 Theoretical Analysis of Output Shift

To understand why CURLoRA is expected to perform better than standard LoRA in terms of catastrophic forgetting, we can analyze the shift in the output during fine-tuning.

For a given input x, the original output is $y = xW$. After fine-tuning:

For LoRA: $y_{\text{adapted}} = x(W + AB)$

For CURLoRA: $y_{\text{adapted}} = x(W + CUR)$

We can quantify the shift using the Frobenius norm of the difference:

$$||y - y_{\text{adapted}}||_F = ||xW - x(W + W_{\text{adapted}})||_F = ||xW - xW - xW_{\text{adapted}}||_F = ||xW_{\text{adapted}}||_F \tag{13}$$

For LoRA: $||x(AB)||_F$

For CURLoRA: $||x(CUR)||_F$

This equation measures the shift in the model's output after fine-tuning. y is the original output, and y_{adapted} is the output after fine-tuning. After fine-tuning for a different task, the adapted output y_{adapted} might shift. We use the Frobenius norm to quantify this shift. If the shift is small, it means that the model's predictions haven't changed much, indicating that the model has retained its original knowledge. As shown, the shift depends on W_{adapted} i.e. to make sure the shift isn't so big, we need to keep W_{adapted} as small (in magnitude or size) as possible.

CURLoRA's main aim is to minimize W_{adapted} while ensuring that the difference $||W - W_{\text{adapted}}||_F$ remains close to $||W||_F$. By focusing on minimizing W_{adapted}, CURLoRA effectively controls the shift in the output, thereby preserving the model's original behavior and mitigating catastrophic forgetting.

Theoretically, CURLoRA should result in a smaller shift because:

1. The C and R matrices are directly sampled from W, maintaining some structure of the original matrix.
2. The C and R matrices are sampled from columns and rows with lower values.
3. Only U is trained, which is constrained by C and R.
4. The initialization of U as a zero matrix.

This constrained adaptation in CURLoRA is expected to lead to better preservation of the model's original knowledge, thereby reducing catastrophic forgetting.

4.5 Memory Efficiency

CURLoRA offers significant memory savings compared to full fine-tuning and even LoRA. For a weight matrix $W \in \mathbb{R}^{m \times n}$, the number of trainable parameters for each method, considering rank r where $r < n$, is:

- Full fine-tuning: mn
- LoRA (rank r): $mr + nr$
- CURLoRA (rank r): r^2

The memory savings can be substantial, especially for large matrices. In our Mistral experiment, with rank 16, the trainable parameters were:

- Full fine-tuning: 7,248,023,552 parameters
- LoRA: 9,437,184 parameters
- CURLoRA: 24,576 parameters

This reduction in trainable parameters not only saves memory but also potentially leads to faster training and inference times.

In conclusion, CURLoRA provides multiple mathematical mechanisms that can help mitigate catastrophic forgetting:

- It constrains the parameter space of possible adaptations.
- It provides implicit regularization towards the original weights.
- It preserves important directions from the original weight matrix.
- It reduces the degrees of freedom in adaptation, limiting potential deviation.
- It allows for direct control and analysis of weight stability through the U matrix.

These properties suggest that CURLoRA can indeed help in reducing catastrophic forgetting while still allowing for meaningful and good adaptation to new tasks. The effectiveness of these theoretical mechanisms are validated through our experiments on various tasks and datasets, as detailed in the following sections.

5 Methodology

5.1 CURLoRA Implementation

Our CURLoRA implementation consists of the following steps:

1. **Decomposition**: For each weight matrix W in the layers we want to apply CURLoRA to, we perform the following:
   - Compute column probabilities: $p_j = \frac{\|W_{:j}\|_2^2}{\|W\|_F^2}$
   - Invert probabilities: $\tilde{p}_j = \frac{1/p_j}{\sum_{i=1}^{n} 1/p_i}$
   - Sample columns and rows according to $\tilde{p}_j$ to construct C and R
   - Initialize U as a zero matrix
2. **Fine-tuning**:
   - **Objective**:
     - The primary objective of the experiment is to evaluate catastrophic forgetting during continual learning, rather than to optimize accuracy for each individual task.
   - **Model Specific Adjustments**:
     - For GPT-2 and Mistral, the model's "lm_head" is replaced with a task-specific output layer. During training, only the U matrix is continually updated, while C and R remain fixed.
     - Replacing the "lm_head" ensures that each task has its own task-specific output layer that remains untouched when the model is being fine-tuned on a different task, contributing to the mitigation of task knowledge degradation.
   - **Continual Learning Strategy**:
     - Once a weight matrix is decomposed, C and R are fixed permanently. The U matrix is continually updated for each new task to facilitate continual learning.
   - **Application of CURLoRA**:
     - CURLoRA is applied to the attention layers (Query, Key, Value). [15]
3. **Inference**: Use the adapted weight matrix $W_{\text{adapted}} = CUR$ for forward passes along with the original W matrix i.e. $x(W + CUR)$.

6 Experiment Setup

6.1 Datasets

We used the following datasets for our experiments:

- **GLUE-MRPC**: Microsoft Research Paraphrase Corpus for paraphrase detection [16]
- **GLUE-SST-2**: Stanford Sentiment Treebank for binary sentiment classification [17]
  These datasets are part of the General Language Understanding Evaluation (GLUE) benchmark [18], which includes a diverse set of tasks for evaluating natural language understanding systems.
- **Sentiment140**: A large-scale sentiment analysis dataset [19]
- **WikiText-2**: A dataset that we use to measure language model perplexity [20]

The datasets were selected for their diverse task requirements and common use in benchmarking.

6.2 Model and Hyperparameters

We used **Mistral 7B** (v0.3) [21] and **GPT-2 Large** [22] as our base models. For both LoRA and CURLoRA, we used the following hyperparameters:

- Ranks: [8, 16, 24]
- Alpha: 1
- Optimizer: AdamW
- Learning rate: 2.5e-4
- Scheduler: Cosine with 500 warmup steps
- Training epochs: 3
- Batch size:
  - Mistral: 8
  - GPT-2: 32
- Max length:
  - Mistral: 512
  - GPT-2: 256

6.2.1 Notes on hyperparemeters and architecture

- **Robustness and Regularization**:
  - CURLoRA's performance was evaluated across different ranks, demonstrating robustness to moderate changes. Optimal results can be achieved by fine-tuning other hyperparameters, such as the learning rate. Dropout was not utilized, as the objective was to observe the implicit regularization effects of CURLoRA without the influence of explicit regularization.
- **Data Constraints**:
  - For Mistral, each fine-tuning task was limited to 1000 records to simulate scenarios with limited data and resources for large models.
  - For GPT-2, the SST-2 fine-tuning task was limited to 5000 records due to resource constraints.
  - For the sentiment analysis task, the Sentiment140 test dataset was used for training, while the train dataset was used for evaluation. This choice was made because the test dataset has three labels, whereas the train dataset has only two. This allowed for fine-tuning the models on a multi-class task rather than a binary one.
- **Task Specific Adjustments**:
  - For the sentiment analysis task with GPT-2, due to the small size of the dataset used for fine-tuning, the number of epochs was adjusted to 5, and the learning rate scheduler was not used.

6.3 Evaluation Metrics

We used the following metrics for evaluation:

- Accuracy: For classification tasks (MRPC, SST-2, Sentiment140)
- Perplexity: For language modeling capability (WikiText-2)

6.4 Experimental Procedure

Our experimental procedure was as follows:

1. Measure initial perplexity of the base model on WikiText-2 concatenating the whole dataset into a single string.
2. Fine-tune on MRPC and evaluate.
3. Fine-tune on SST-2 and evaluate, then re-evaluate on MRPC.
4. Fine-tune on Sentiment140 and evaluate, then re-evaluate on MRPC and SST-2.
5. Re-calculate perplexity on WikiText-2.

This procedure was carried out for both LoRA and CURLoRA independently.

7 Results and Discussion

Tables 1 and 2 present the results of our experiments comparing LoRA and CURLoRA across multiple tasks and evaluation metrics.

Table 1: Mistral Experimental Results: LoRA vs CURLoRA

Metric	LoRA-8	CURLoRA-8	LoRA-16	CURLoRA-16	LoRA-24	CURLoRA-24
Initial WikiText-2 Perplexity	5.44	5.44	5.44	5.44	5.44	5.44
MRPC Accuracy (After MRPC)	0.68	0.66	0.65	0.66	**0.67**	0.66
SST-2 Accuracy (After SST-2)	0.51	**0.86**	0.51	0.86	0.49	0.86
MRPC Accuracy (After SST-2)	**0.68**	0.66	0.32	0.66	0.68	0.66
Sentiment140 Accuracy	**1.00**	0.94	1.00	0.94	1.00	0.94
MRPC Accuracy (After Sentiment140)	0.32	**0.66**	0.32	0.66	0.32	0.66
SST-2 Accuracy (After Sentiment140)	0.49	**0.86**	0.49	0.86	0.49	0.86
Final WikiText-2 Perplexity	53896.68	**5.44**	65055.02	**5.44**	17049.72	**5.44**

Table 2: GPT-2 Large Experimental Results: LoRA vs CURLoRA

Metric	LoRA-8	CURLoRA-8	LoRA-16	CURLoRA-16	LoRA-24	CURLoRA-24
Initial WikiText-2 Perplexity	28.25	28.25	28.25	28.25	28.25	28.25
MRPC Accuracy (After MRPC)	0.79	0.70	0.81	0.70	**0.83**	0.70
SST-2 Accuracy (After SST-2)	**0.94**	0.76	0.93	0.79	0.92	0.86
MRPC Accuracy (After SST-2)	0.76	0.70	**0.78**	0.70	**0.78**	0.70
Sentiment140 Accuracy	0.92	**0.99**	0.86	0.99	0.93	0.93
MRPC Accuracy (After Sentiment140)	0.49	0.70	**0.73**	0.70	0.49	0.70
SST-2 Accuracy (After Sentiment140)	**0.90**	0.76	0.90	0.79	0.88	0.87
Final WikiText-2 Perplexity	42.96	**28.25**	43.62	**28.08**	44.32	**28.25**

7.1 Performance Analysis

7.1.1 Task-Specific Performance

CURLoRA consistently performed well on different tasks, showing high accuracy even after fine-tuning on subsequent tasks. This suggests that CURLoRA is more effective at preserving task-specific knowledge.

7.1.2 Catastrophic Forgetting and Stability

The stability of CURLoRA's performance across tasks is particularly noteworthy. While (Mistra) LoRA-16's accuracy, for example, on MRPC dropped from 0.6495 to 0.32 after fine-tuning on other tasks, CURLoRA-16 (Mistral) maintained its accuracy at 0.66. This demonstrates CURLoRA's superior ability to mitigate catastrophic forgetting.

7.1.3 General Language Modeling Capability

The final perplexity scores on WikiText-2 provide strong evidence for CURLoRA's effectiveness in preserving general language modeling capabilities. While all LoRA's perplexity, in both Mistral and GPT2, increased dramatically, all CURLoRA models maintained the original perplexity, indicating no degradation in general language understanding.

7.2 Theoretical Insights

The experimental results align with our theoretical analysis:

- **Parameter Space Constraint**: The stability of CURLoRA's performance across tasks supports our hypothesis that constraining adaptations to the subspace spanned by C and R helps preserve original knowledge.
- **Implicit Regularization**: The maintained perplexity on WikiText-2 suggests that CURLoRA's implicit regularization effectively prevents overfitting to specific tasks.
- **Reduced Interference**: The consistent performance across tasks indicates that CURLoRA successfully reduces interference between task-specific adaptations.

7.3 Limitations and Future Work

While CURLoRA shows promising results, there are several areas for future research:

- **Scalability**: While CURLoRA shows promising results, its scalability to larger models needs further investigation. Further studies are needed to assess CURLoRA's performance on larger models and more diverse tasks like instruction tuning and datasets.
- **Computational Complexity**: Conducting detailed analysis of time and space complexity compared to full fine-tuning and LoRA.
- **Implicit Regularization Limitation**: Implicit regularization via zero initialization of U has to be further studied especially in highly dynamic environments where more flexible adaptations are needed.
- **Optimal Rank and Alpha Selection**: Investigating methods for automatically selecting the optimal rank and alpha for CURLoRA could further improve performance.
- **Combination with Other Techniques**: Exploring the integration of CURLoRA with other continual learning techniques could yield even better results.
- **Quantization Support**: Exploring the implementation of CURLoRA on quantized model which may lead to QCURLoRA

8 Conclusion

This paper introduced CURLoRA, a novel approach to fine-tuning large language models that leverages CUR matrix decomposition to mitigate catastrophic forgetting and improve computational efficiency. Through theoretical analysis and empirical experiments, we demonstrated that CURLoRA outperforms standard LoRA in maintaining model stability and performance across tasks while significantly reducing the number of trainable parameters.

Key contributions of this work include:

- A novel modification to CUR decomposition using inverted probabilities for column and row selection and initiating U matrix as zeros. Sampling columns and rows based on inverted probabilities distinguishes CURLoRA from traditional CUR, offering better stability and performance.
- Theoretical analysis of how CURLoRA addresses catastrophic forgetting.
- Empirical evidence of CURLoRA's effectiveness across multiple tasks and evaluation metrics with multiple models.

Our results suggest that CURLoRA is a promising approach for efficient and stable fine-tuning of large language models, particularly in scenarios with limited fine-tuning data. CURLoRA's approach to mitigating catastrophic forgetting has broad implications for continual learning in NLP and beyond. Future research could explore its integration with other adaptation techniques to enhance model robustness

References

[1] 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.

[2] Xiang Lisa Li and Percy Liang. Efficient few-shot learning without prompts. arXiv preprint arXiv:2111.10952, 2021.

[3] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023.

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In International Conference on Learning Representations, 2021.

[5] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of learning and motivation, 24:109–165, 1989.

[6] Michael W Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences, 106(3):697–702, 2009.

[7] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13):3521–3526, 2017.

[8] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pages 8154–8162, 2016.

[9] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. Advances in Neural Information Processing Systems, 32, 2019.

[10] 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.

[11] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190, 2021.

[12] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Relative-error cur matrix decompositions. SIAM Journal on Matrix Analysis and Applications, 30(2):844–881, 2008.

[13] Nishant Yadav, Nicholas Monath, Manzil Zaheer, and Andrew McCallum. Efficient k-nn search with cross-encoders using adaptive multi-round cur decomposition, 2023.

[14] Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. Lora-xs: Low-rank adaptation with extremely small number of parameters. arXiv preprint arXiv:2405.17604, 2024.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[16] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In Proceedings of the Third International Workshop on Paraphrasing (IWP2005), 2005.

[17] 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.

[18] 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. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pages 353–355, 2018.

[19] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. In CS224N project report, Stanford, volume 1, page 2009, 2009.

[20] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843, 2016.

[21] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. arXiv preprint arXiv:2310.06825, 2023.

[22] 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.