**Comparative Analysis of Efficient Adapter-Based Fine-Tuning of State-of-the-Art Transformer Models**

|  |  |  |
| --- | --- | --- |
| Mohammad Ali Sheikh  msheikh36@gatech.edu | Muhammad Aleem  maleem3@gatech.edu  Kajol Ramesh Singh ksingh362@gatech.edu | Saad M. Siddiqui ssiddiqui60@gatech.edu |

CS-7643 Deep Learning - Final Project

**Abstract**

*In this work, we investigate the efficacy of various adapter architectures on supervised binary classification tasks from the SuperGLUE benchmark as well as a supervised multi-class news category classification task from Kaggle. Specifically, we compare classification performance and time complexity of three transformer models, namely DistilBERT, ELECTRA, and BART, using conventional fine-tuning as well as nine state-of-the-art (SoTA) adapter architectures. Our analysis reveals performance differences across adapter architectures, highlighting their ability to achieve comparable or better performance relative to fine-tuning at a fraction of the training time. Similar results are observed on the new classification task, further supporting our findings and demonstrating adapters as efficient and flexible alternatives to fine-tuning. This study provides valuable insights and guidelines for selecting and implementing adapters in diverse natural language processing (NLP) applications.*

# **Introduction**

## **Problem and Existing Approaches**

Transformer-based models (LLMs) represent the SoTA in terms of NLP applications, having consistently demonstrated unparalleled performance across a wide range of tasks. These models are characterized by deep neural network architectures with large number of learnable parameters, often approaching the order of hundreds of millions, which are optimized through a combination of supervised, self-supervised, and semi-supervised pre-training on vast corpora.

NLP researchers and practitioners are interested in leveraging the capabilities of transformer models for domain-specific tasks, which invariably require updating or tuning learnable parameters. A popular method for doing so is fine-tuning, which involves updating either all learnable parameters or a subset of learnable parameters with methods such as gradient descent using loss from a smaller, task-specific labeled dataset. Another popular method is domain-adaptation, which involves aligning model performance with a target domain through adversarial training, domain-invariant representations, and other techniques which involve a mix of source and target domain data.

Despite their efficacy, both methods are often expensive in terms of time, often taking prohibitively long owing to the large number of learnable parameters that need to be updated. There is an also an additional risk of overfitting and issues with gradient flow, as the absence of appropriate regularization techniques and architectural constructs for gradient flow augmentation can cause problems with generalization when tuning a large number of parameters using small, labeled corpora.

## **Adapters**

Adapters are lightweight task-specific modules, typically consisting of projection layers or small neural networks, that can be inserted between layers in transformer models. During training, only adapter parameters are updated while the rest of the model parameters remain unchanged. This makes adapters a parameter-efficient alternative to finetuning and domain adaptation: adapters allow the retention of generalizable knowledge learned from large corpora as encoded by original model weights while still allowing the parameters of the adapter to learn task-specific information. Contrary to fine-tuning a classification head in conventional transfer learning, the insertion of adapter modules is not limited to later layers in the network.

Successful integration of adapters in transformer-based models can significantly reduce computational resources and time needed for modifying models for a specific domain or task, even with limited labeled data. This parameter efficiency means that adapting models to new tasks will become faster, cheaper, and more accessible, particularly for researchers and practitioners with limited resources. It will enable more widespread use of powerful language models in various applications, fostering innovation and advancing the field of NLP by making state-of-the-art tools available to a broader audience.

## **Objectives**

This project aimed to empirically validate and quantify the classification and time complexity benefits of adapters as parameter-efficient alternatives to fine-tuning transformer models for supervised classification. Additionally, the project also aimed to identify which adapter designs performed best across a range of transformer models and tasks to understand their advantages over traditional methods.

## **Datasets**

We have used two main datasets in this project.

1. **SuperGLUE**: A collection of diverse and challenging NLP tasks designed to evaluate the performance of language models on understanding and reasoning. It consists of tasks such as question answering, natural language inference, and co-reference resolution, providing a comprehensive assessment of model capabilities.
2. **News Category Classification**: A Kaggle dataset consisting of 210,294 records of news articles from the Huffington Post collected between 2012 and 2022 with predictors such as headlines, authors, and publication date mapped to 42 highly imbalanced categories such as Politics, Wellness, Entertainment, to name a few.

In the SuperGLUE benchmark, we specifically focus on the following binary classification tasks, partly due to their simplicity and partly due to time and resource constraints.

Table 1: Summary of SuperGLUE Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Description** | **Training**  **Set Size** | **Validation**  **Set Size** |
| Boolean Questions | Answering yes/no questions | 9427 | 3270 |
| Commitment Bank | Textual entailment task | 250 | 56 |
| Recognizing Textual  Entailment | Binary entailment classification task | 2500 | 278 |

To supplement these tasks, we explored model performance on the Corpus of Linguistic Acceptability (CoLA), Winograd NLI (WNLI), and Microsoft Research Paraphrase Corpus (MRPC) tasks from the GLUE benchmark. **In the interest of brevity, however, we have chosen to focus the rest of the paper exclusively on the tasks in Table 1 and News Classification**.

Exploring the performance of adapters on the latter task was motivated by a choice to explore if model performance trends on a binary classification academic benchmark such as SuperGLUE also generalized to a complex, multi-class classification task using real-world data.

# **Approach**

At a high level, our approach to the problem involved running experiments to benchmark model performance in terms of validation set accuracy and F1 scores (where applicable) as well as training time with and without one each of 9 possible adapters across three SuperGLUE tasks and the News Classification task with three SoTA transformer models, namely DistilBERT, ELECTRA, and BART.

## **Addressing the Problem**

To address our problem, we used HuggingFace’s Transformers library along with the Adapters library. The Adapters repository provided code for running simple sequential bottleneck adapters on GLUE tasks, but we needed to extend this code to fit our requirements. Specifically, we added support for SuperGLUE, incorporating tasks from Table 1. We also extended the repository to include various adapter architectures such as sequential bottleneck, stacked sequential bottleneck Mix and Match, iA3, LoRA, Prefix Tuning, Compacter++, Prompt Tuning, and UniPELT. A full description of each adapter, its architecture, and potential use cases based on literature review is provided in the Appendix.

Additionally, we modified the code to return epoch-level estimates of training and validation loss as well as task-specific model evaluation metrics such as accuracy, F1 score, and Matthew’s Correlation. We also modified the code base to handle traditional fine-tuning on the same tasks for comparison. Our approach therefore resulted in a single pipeline that could be used to benchmark model performance in terms of evaluation metrics and time complexity with and without adapters across all three transformer models of interest.

## **Novelty**

Firstly, our approach is novel in its comprehensive evaluation of a wide range of adapter architectures on state-of-the-art models and complex benchmarks. By including a practical news classification task, we aimed to provide real-world insights into the effectiveness of adapters, highlighting their potential for efficient and flexible adaptation of pre-trained language models to new tasks.

Additionally, by considering epoch-level learning curves in conjunction with end-of-training point estimates of model performance with and without adapters, the comprehensive nature of our analysis provides clear insights into the efficiency and effectiveness of adapters, demonstrating their potential as a viable alternative to full model fine-tuning in NLP tasks. By evaluating a wide range of adapter architectures, we contributed new knowledge to the field, offering practical guidelines for selecting and implementing adapters in various NLP applications.

## **Hypotheses**

We believed our approach would be successful because adapters offer a more efficient and flexible way to adapt pre-trained models to new tasks, requiring fewer resources than full fine-tuning. By keeping our machine learning hyperparameters constant, we ensured a fair comparison between fine-tuning and various adapter architectures. We ran each configuration for 10 epochs, maintaining consistency in learning rate, batch size, and other relevant hyperparameters to isolate the effect of the adapter architecture itself. This approach allowed us to leverage the strengths of different adapter designs and rigorously evaluate their performance against traditional fine-tuning. The advantage of this methodology is that it highlights how different adapter architectures can provide significant resource savings without compromising performance.

We anticipated that some tasks would take a long time to complete due to the large datasets and the inherent complexity of the tasks. To mitigate this, we utilized GPUs to speed up the training process and carefully planned our training schedules ahead of time to ensure efficient resource use. Additionally, we expected to face challenges with preprocessing the data for each specific task, as different tasks in the SuperGLUE benchmark have unique requirements. We adjusted our code accordingly to handle these preprocessing differences, ensuring that each task was processed correctly for both the adapters and fine-tuning methods.

One of the challenges we anticipated – and indeed did encounter – was that some adapter architectures were not well-suited for certain tasks. For example, our initial attempt using a simple sequential bottleneck adapter on the BoolQ task ran successfully but did not surpass the performance of traditional fine-tuning. This outcome was expected, as the sequential bottleneck is a basic adapter architecture. To address this, we explored more advanced adapter architectures such as stacked sequential bottleneck Mix and Match, iA3, and LoRA, which showed improved performance. Additionally, we faced issues with integrating all adapter types into the existing codebase, which required significant modifications to ensure compatibility and correct metric reporting. Through iterative testing and code adjustments, we managed to overcome these obstacles and achieve a comprehensive evaluation of the different adapter architectures.

# **Experiments**

## **Model Selection**

We used three base models: DistilBERT, BART, and Electra base. These models were chosen for their relatively lightweight architecture and the fact that they have not been extensively tested with adapters in existing literature. DistilBERT, with approximately 66 million parameters, is a distilled version of BERT, making it more efficient while retaining a substantial portion of BERT’s performance capabilities. BART, with around 140 million parameters, is a transformer model that combines bidirectional and autoregressive transformers, suitable for a wide range of NLP tasks including classification. ELECTRA, consisting of about 110 million parameters, is known for its efficiency and performance in pre-training tasks. These models are well-suited for NLP-based binary classification tasks due to their architecture, which effectively captures contextual information necessary for making accurate predictions.

## **Adapter Methods**

We experimented with various adapter methods to understand how they modify the architecture of each model and their impact on performance. The adapters used were seq bn (Sequential Bottleneck), prefix tuning, MAM (Mix and Match) adapter, compacter plusplus (Compacter++), UniPELT (Unified Prompt and Embedding Learning Techniques), LoRA (LowRank Adaptation), stacked seq bn (stacked sequential bottleneck with tanh and ReLU activations), IA3 (Improved Adapter Architecture with Activation and Attention) and prompt tuning. These adapters were selected based on their innovative approaches to modifying the model architecture, aiming to enhance performance with minimal additional parameters.

We deliberately experimented with a wide range of adapters because of the diversity of architectural modifications and preference biases they exhibited. For instance, adapters like Sequence Bottleneck and its stacked variant introduce bottleneck layers that reduce computational overhead while capturing essential information. As such, these adapters were expected to be more generic in their preference biases and utility across SuperGLUE tasks. In contrast, prefix tuning and prompt tuning focus on optimizing input prompts rather than model weights, and were expected not to perform well as generic, out-of-the-box adapters for classification tasks. MAM and Compacter++ utilize advanced parameter sharing and low-rank decomposition techniques to improve efficiency. UniPELT integrates prompt tuning with embedding learning, while LoRA and IA3 introduce low-rank approximations and improved activation functions, respectively, to boost performance.

## **Metrics**

To compare the performance of fine-tuning and adapter training, we used two primary metrics: time to train, evaluation dataset accuracy and training and validation loss by epoch. These metrics were chosen to provide a clear comparison of the computational efficiency and predictive performance of each method. Time to train is crucial for understanding the resource requirements of each approach, while evaluation accuracy directly reflects the effectiveness of the models and adapters in making accurate predictions. Additionally, training and validation loss by epoch helps monitor the learning process and convergence behavior, providing insights into the stability and efficiency of each training method.

## **Hyperparameters**

We used default hyperparameters for all models to ensure a fair comparison. This decision was made to eliminate the variability introduced by hyperparameter tuning and to focus solely on the performance differences attributable to the fine-tuning and adapter methods. By maintaining consistent hyperparameters, we ensured that any observed differences in performance were due to the inherent characteristics of the models and adapters themselves, rather than external tuning factors.

## **News Classification**

All three models were first fine-tuned on the News Classification dataset. Owing to time and resource constraints, we chose to focus on only one of the three models, specifically ELECTRA, for adapter tuning experimentation with the Mix-and-Match and Sequential Bottleneck (ReLU) adapters, partly because these adapters had demonstrated good performance on concurrent SuperGLUE experiments and partly because their architecture seemed to be intuitively compatible with the requirements of the News Classification problem. The data was converted to the Hugging Face Dataset format, and a tokenization function was applied to the text headlines, truncating and padding them to a maximum length of 128 tokens. This prepared the data for efficient training with transformer models. We also implemented custom preprocessing and metrics specific to the news classification task and analyzed the accuracy, F1-score, and loss curves for each configuration, in addition to measuring training time.

# Results and Discussion

Figure 1 - box and whisker plot of metrics per model Figure 2 - Loss curves Table 1 - performance of adapters per task Table 2 - performance of base models

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Adapter** | **Average Validation Set Accuracy** | | | **Average Training Time (seconds)** | | |
| **DistilBERT** | **BART** | **ELECTRA** | **DistilBERT** | **BART** | **ELECTRA** |
|  |
| **no\_adapter** | 0.699 | 0.763 | 0.750 | 405.097 | 993.363 | 804.641 |  |
| **compacter\_plusplus** | 0.642 | 0.651 | 0.642 | 237.038 | 553.714 | 474.077 |  |
| **ia3** | 0.644 | 0.655 | 0.648 | 238.930 | 565.351 | 483.088 |  |
| **lora** | 0.647 | 0.668 | 0.705 | 234.217 | 570.378 | 468.389 |  |
| **mam\_adapter** | 0.649 | 0.734 | 0.734 | 301.291 | 703.610 | 583.443 |  |
| **prefix\_tuning** | 0.637 | 0.646 | 0.640 | 243.644 | 566.234 | 481.239 |  |
| **prompt\_tuning** | 0.623 | 0.646 | 0.577 | 257.721 | 315.335 | 506.995 |  |
| **seq\_bn** | 0.670 | 0.691 | 0.750 | 248.974 | 574.488 | 497.056 |  |
| **seq\_bn\_16\_2\_relu** | 0.670 | 0.740 | 0.775 | 262.672 | 605.005 | 529.235 |  |
| **seq\_bn\_16\_2\_tanh** | 0.677 | 0.739 | 0.782 | 262.099 | 605.389 | 529.138 |  |
| **unipelt** | 0.672 | 0.705 | 0.736 | 331.829 | 824.208 | 634.885 |  |

A group of graphs showing different types of data

Description automatically generated with medium confidence



A screenshot of a graph

Description automatically generated

## Adapters and Performance Efficiency

Our experiments reveal that adapters consistently lead to performance comparable to traditional fine-tuning while requiring significantly less training time. This is evident from Figure 1, which presents a box and whisker plot illustrating the distribution of time complexity with and without adapters. The plot demonstrates that the training times for models with adapters (represented in blue) are generally lower than those without adapters (represented in red), while their evaluation accuracies remain on par.

For instance, in the GLUE tasks, the training time for DistilBERT with adapters is significantly less than without adapters, and the evaluation accuracy remains comparable. Similarly, for Electra on SuperGLUE tasks, the adapterenabled models show a substantial reduction in training time without a loss in evaluation accuracy. However, some anomalies are present, such as BART on GLUE tasks, where the adapter-based model exhibits slightly longer training times in some configurations. These anomalies may arise from specific implementation details or variations in hyperparameter settings, but overall, the trend supports the efficiency of adapters.

## Sequential Bottleneck Adapters on GLUE andSuperGLUE

The sequential bottleneck adapters with tanh and ReLU activations (seq bn tanh and seq bn relu) show a tendency to overfit simpler tasks in the GLUE benchmark but excel in the more complex SuperGLUE tasks. Specifically, on SuperGLUE, the seq bn tanh achieves an evaluation accuracy of 0.733, and the seq bn relu achieves 0.728, making them the top performers. However, their average accuracy on the GLUE tasks is only 0.573 and 0.583, respectively, as shown in Table 1. This discrepancy can be attributed to the complex architecture of these adapters, which includes two sequential bottleneck layers. The architecture allows the model to capture more intricate patterns, beneficial for complex tasks but prone to overfitting simpler ones. Figure 2 further supports this, showing a significant gap between training and validation loss curves for these adapters on GLUE tasks.

## LoRA and Sequential Bottleneck Adapters onGLUE and SuperGLUE

Conversely, LoRA and seq bn adapters exhibit strong performance on simpler GLUE tasks but underperform on the more complex SuperGLUE tasks. Their tendency to underfit on SuperGLUE is evident from the loss curves in Figure 2, which show no significant gap between training and validation loss. This suggests that these adapters may not capture the complex relationships required for SuperGLUE tasks. However, their simpler architecture, which imposes fewer constraints on the model, makes them more suitable for the straightforward nature of GLUE tasks.

## Performance of Prompt Tuning and Prefix Tuning adapters

Prompt tuning and prefix tuning adapters perform poorly across our tasks, as they are highly task-specific. According to Table 1, these adapters consistently rank at the bottom. Prefix Tuning and Prompt Tuning adapters are generally less effective for classification tasks due to their architectural design, which is tailored more towards tasks requiring rich contextual understanding and generation rather than straightforward classification. Prefix Tuning involves prepending trainable prefix tokens to the input sequence in the transformer model, primarily modifying the keys and values in the multi-head attention mechanism. This setup enhances the model’s ability to handle sequence-tosequence tasks such as text generation or translation. However, for classification tasks, the additional complexity introduced by prefix tokens does not significantly benefit the process. Classification tasks typically require the model to make decisions based on a fixed representation of the input, and Prefix Tuning’s focus on the attention mechanism does not substantially enhance the core feed-forward layers crucial for capturing discriminative features necessary for classification. Similarly, Prompt Tuning adapts the model by adding a fixed prompt to the input, a learned embedding that influences the model’s processing of the data. This design is particularly effective for generative tasks like question answering or text completion, where the model needs to predict or generate text based on the given context. However, in classification tasks, Prompt Tuning’s reliance on prompt-based input modification is less effective. Classification tasks require a clear-cut decision boundary and efficient transformation of input features through dense layers, which prompt-based modifications do not enhance directly. Therefore, Prompt Tuning and Prefix Tuning both perform bad on GLUE and SuperGLUE.

## Effectiveness of Mix and Match Adapter

The Mix and Match (mam adapter) proves to be a robust adapter, performing well across both GLUE and SuperGLUE tasks. It ranks highest on GLUE and among the top three on SuperGLUE, as indicated in Table 1. This suggests that mam adapter strikes an effective balance between simplicity and complexity. Its architecture, which combines Prefix Tuning and parallel bottleneck adapters, allows it to dynamically adapt to different task requirements, making it well-suited for the diverse demands of classification tasks. Prefix Tuning involves prepending trainable prefix tokens to the input sequence, enhancing the model’s ability to capture relevant information from the input sequence by modifying the keys and values in the multi-head attention mechanism. This component is crucial for identifying discriminative features in classification tasks. Parallel Bottleneck Adapters, on the other hand, are lightweight modules inserted in parallel within the feed-forward blocks of the transformer architecture. These adapters consist of downprojection, non-linearity, and up-projection layers, ensuring that the model can handle complex transformations efficiently. This design is essential for extracting and processing relevant features from the input data.

## Comparison Across Models

When comparing adapter-based training across Distil-

BERT, BART, and Electra base models, as shown in Table 2, we see distinct performance trends. For GLUE tasks, the Electra base model achieves the highest evaluation accuracy of 0.715514 with a training runtime of 676.499733 seconds. BART follows with an accuracy of 0.647231 but with a significantly higher runtime of 850.637433 seconds. DistilBERT has the lowest accuracy of 0.533019 but the shortest runtime of 346.250467 seconds. For SuperGLUE tasks, BART leads with an accuracy of 0.763387 and a runtime of 992.844100 seconds. Electra base is close behind with an accuracy of 0.750473 and a runtime of 804.256600 seconds, while DistilBERT achieves an accuracy of 0.699274 with a runtime of 404.445200 seconds. Given these results, Electra base emerges as the best model overall due to its balance of high accuracy and reasonable training time, attributed to its efficient architecture designed for both pre-training and fine-tuning tasks.

## Comparison and Analysis for News CategoryDataset

We selected the electra-base-discriminator model based on its performance in our initial experiments on the GLUE and SuperGLUE benchmarks. The model was evaluated with and without adapters to determine the effectiveness of adapter-based fine-tuning compared to traditional methods. The selected adapters were MAM and seq bn 16 2 relu based on previous experiments. We reused some parts of the base code, and made necessary modifications to adapt it for the news classification dataset and run experiments with some base configuration.

The fine-tuned model (without adapters) achieved the highest evaluation accuracy (0.6365) and F1 score (0.6246), indicating that full fine-tuning can still be the most effective method for achieving high performance on the news classification task and also had the lowest evaluation loss (1.3247), which aligns with its higher accuracy and F1 score. The MAM adapter showed a slight drop in performance with an evaluation accuracy of 0.6135 and an F1 score of 0.5962. It also had slightly higher eval loss. The Sequential Bottleneck adapter (seq bn 16 2 relu) had the lowest evaluation accuracy (0.5667), F1 score (0.5319) among the three configuration and higher eval loss (1.5746). The evaluation runtime was lowest for the Sequential Bottleneck adapter (83.7581s), followed by the MAM adapter (87.0635s), and the highest for the fine-tuned model (89.2809s). This shows that while adapters can reduce evaluation time slightly, the difference is not substantial. Training time was significantly reduced with adapters. The fine-tuned model took 2:33:41, whereas the MAM adapter reduced the time to 2:02:46, and the Sequential Bottleneck adapter further reduced it to 1:55:44. This demonstrates one of the primary advantages of using adapters: reduced training time.

From learning curves in Figure for the electra-basediscriminator model with different configurations (no adapter, MAM adapter, and seq bn 16 2 relu adapter), here are the insights:

* Without Adapters: The model learns effectively but shows signs of overfitting, as indicated by the gap between training and validation losses.
* With MAM Adapter: The model shows smooth convergence and less overfitting, suggesting that the MAM adapter provides beneficial regularization.
* With seq bn 16 2 relu Adapter: The model learns effectively but still shows signs of overfitting, although to a lesser extent compared to the model without adapters.

This suggests that the use of adapters, particularly the MAM adapter, helped reduce overfitting and provided a more stable convergence.

# Conclusion

The results from our experiments demonstrate that adapter-based training provides a highly effective and efficient alternative to traditional fine-tuning for NLP classification tasks. Adapters generally achieve comparable

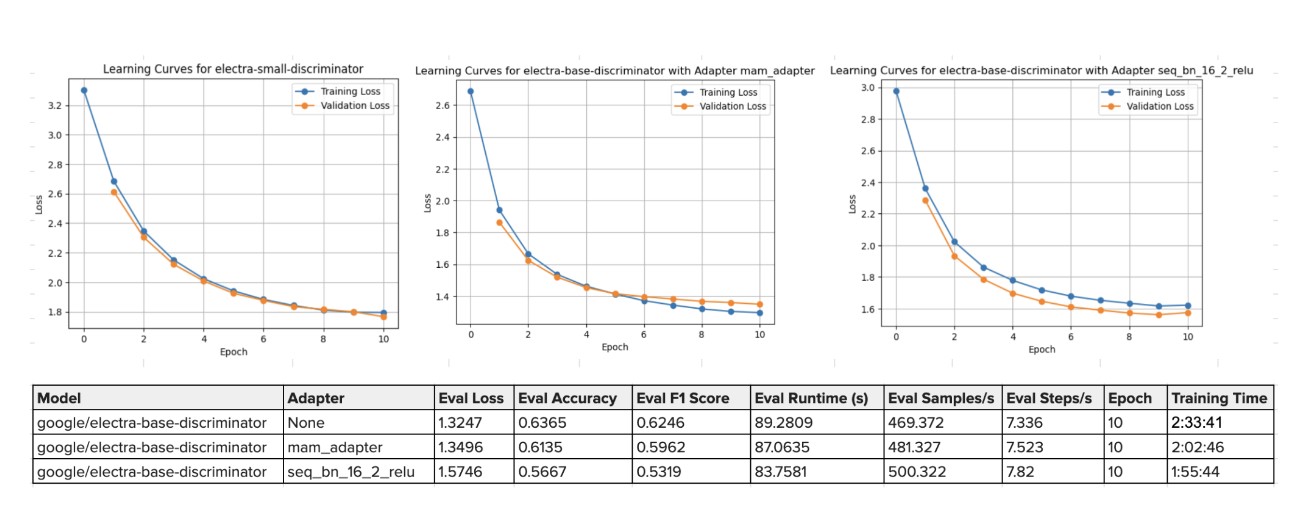


Figure 1. Comparative Results of electra-base-discriminator on News Category Dataset with and without Adapters

performance to full fine-tuning while significantly reducing training time, as evidenced by Figure 1. Specific adapters, such as Mix-and-Match (MAM), exhibit robust performance across both GLUE and SuperGLUE tasks, balancing the complexity and efficiency of their architecture. Sequential bottleneck adapters with tanh and relu activations show a tendency to overfit on simpler GLUE tasks but perform well on more complex SuperGLUE tasks, suggesting that their stacked architecture is better suited for handling more intricate datasets. Conversely, adapters like LoRA and sequential bottleneck adapters tend to underfit on complex tasks, indicating a need for further refinement in their design for such applications. Furthermore, adapters like Prompt Tuning and Prefix Tuning, while powerful for generative tasks, do not align well with the requirements of classification due to their architectural focus on contextual understanding and sequence generation rather than feature extraction and decision-making.

Future experiments could expand on these findings by exploring larger base models with more parameters, such as BERT-large or GPT-3, to evaluate the scalability and effectiveness of adapters in more complex scenarios. Additionally, more extensive hyperparameter tuning could provide deeper insights into optimizing adapter performance across various tasks. However, scaling up to models with millions of parameters presents significant computational challenges. Solutions like NVIDIA TensorRT could be employed to optimize inference performance and reduce the computational load, making it feasible to run these large models efficiently.

Moreover, the difficulties associated with datasets, such as the low evidence for some tasks in the GLUE and SuperGLUE benchmarks, underscore the need for highquality, well-annotated datasets to achieve reliable performance evaluations. Future work could involve curating or identifying more robust datasets with higher evidence and clearer task definitions to mitigate these challenges. Additionally, leveraging advanced data augmentation techniques and synthetic data generation could help improve the robustness and generalizability of models trained with adapters.

The news classification task also revealed that adapterbased tuning might not always outperform traditional finetuning, highlighting the importance of task-specific considerations and suggest the need for further refinement and task-specific tuning. Our findings support the continued exploration of adapter-based methods, with a focus on optimizing configurations and addressing dataset-specific challenges to maximize their effectiveness

In conclusion, while adapter-based training offers significant advantages in terms of efficiency and performance, future research should focus on scaling these methods to larger models, optimizing hyperparameters, and addressing dataset-related challenges to fully realize their potential in NLP classification tasks. These efforts will not only enhance model performance but also contribute to the development of more efficient and scalable solutions for a wide range of NLP applications.

(10 points) How did you measure success? What experiments were used? What were the results, both quantitative and qualitative? Did you succeed? Did you fail? Why? Justify your reasons with arguments supported by evidence and data.

Important: This section should be rigorous and thorough. Present detailed information about decision you made, why you made them, and any evidence/experimentation to back them up. This is especially true if you leveraged existing architectures, pretrained models, and code (i.e. do not just show results of fine-tuning a pre-trained model without any analysis, claims/evidence, and conclusions, as that tends to not make a strong project).

# Other Sections

You are welcome to introduce additional sections or subsections, if required, to address the following questions in detail.

(5 points) Appropriate use of figures / tables / visualizations. Are the ideas presented with appropriate illustration? Are the results presented clearly; are the important differences illustrated?

(5 points) Overall clarity. Is the manuscript selfcontained? Can a peer who has also taken Deep Learning understand all of the points addressed above? Is sufficient detail provided?

(5 points) Finally, points will be distributed based on your understanding of how your project relates to Deep Learning. Here are some questions to think about:

What was the structure of your problem? How did the structure of your model reflect the structure of your problem?

What parts of your model had learned parameters (e.g., convolution layers) and what parts did not (e.g., postprocessing classifier probabilities into decisions)?

|  |  |  |
| --- | --- | --- |
| Student Name | Contributed Aspects | Details |
| Mohammad Ali Sheikh | Data Creation and Implementation | Scraped the dataset for this project and trained the CNN of the encoder. Implemented attention mechanism to improve results. |
| Muhammad Aleem | Writing base script | Implemented the base script to run base experiments on SuperGLUE tasks with the 5 adapters mentioned in the paper |
| Saad M. Siddiqui | Data Creation and Implementation | Scraped the dataset for this project and trained the CNN of the encoder. Implemented attention mechanism to improve results. |
| Kajol Ramesh Singh | New classification task - experiments and analysis | Made necessary modifications to adapt base code for the news category dataset, and run experiments for all three models, with fine tuning and with multiple adapters. |

Table 2. Contributions of team members.

What representations of input and output did the neural network expect? How was the data pre/post-processed? What was the loss function?

Did the model overfit? How well did the approach generalize?

What hyperparameters did the model have? How were they chosen? How did they affect performance? What optimizer was used?

What Deep Learning framework did you use?

What existing code or models did you start with and what did those starting points provide?

Briefly discuss potential future work that the research community could focus on to make improvements in the direction of your project’s topic.

# Work Division

The delegation of tasks among team members was crucial for the successful execution of the project. Muhammad Aleem initially found the base code for GLUE tasks using a single adapter and extended it to include SuperGLUE and other adapters. This foundational work set the stage for further development and integration of additional methods. [insert rest of the team work]. Kajol R Singh focused on the news classification task, reusing the existing base code and making necessary modifications to adapt it for the news classification dataset, and conducting experiments with all three models, including their adapter configurations.

8. Code

# References