Adaptive Text-to-Command Translation for Robot Navigation: Fine-Tuning T5-Small for Natural Language-Based Control

Adrien Ramsamy, Akhil Javvadi, Sai Jagadeesh Muralikrishnan,
Suhas Nagaraj, Varun Lakshmanan, Vijay Dev Reddy Chevireddi
University of Maryland, College Park
{aramsamy, ajavvadi, jagkrish, suhas99, varunl11, creddy}@umd.edu

Abstract

Natural Language Processing for robotic navigation offers significant insights but remains underexplored. This work focuses on enabling robots to interpret natural language commands and translate them into actionable navigation plans. We fine-tune a T5-Small model using task-specific synthetic datasets to address this challenge and implement Low-Rank Adaptation (LoRA) to optimize computational efficiency by reducing trainable parameters to 0.36%. The approach was evaluated on a diverse synthetic test dataset comprising varied task complexities. This work demonstrates the feasibility of transformer-based models for real-world robotic control and highlights LoRA as a resource-efficient fine-tuning solution. These findings establish a strong foundation for scalable, natural language-driven robotic systems.

1 Introduction

Improving human-robot interaction in dynamic environments requires enabling robots to understand and carry out natural language commands. This feature enables robots to comprehend and convert commands such as "Move to the red battery and then proceed to the blue battery" into organized navigation plans. However, due to the varied and adaptable nature of human language inputs, conventional navigation systems that depend on rule-based or template-driven approaches find it difficult to generalize. As a result, models that can handle natural language variability robustly without relying on visual signals or pre-structured templates are required.

Although Vision-and-Language Navigation (VLN) techniques (4) combine language commands with visual information, they are not intended for situations in which the robot must function only with natural language inputs. However, transformer-based models, such as T5 (1), have demonstrated impressive performance in tasks that are solely text-to-text, including question-answering, summarization, and machine translation. These models are interesting options for text-based robotic navigation because of their exceptional ability to handle linguistic variety. However, not much research has been done on their use in robotic control, especially when it comes to real-world navigation problems.

The lightweight T5-Small model is refined in our attempt to convert natural language navigation directions into sequential, structured navigation plans. In order to replicate real-world navigation tasks requiring sequences of waypoints (such as batteries represented by changing colors), we create a synthetic dataset containing over 24,581 navigation instructions of various difficulties.

We evaluate the fine-tuned T5-Small model and compare its performance to a Low-Rank Adaptation (LoRA) (5) based fine-tuning approach. LoRA significantly reduces the number of trainable parameters to only 0.36% of the full model, making it computationally efficient for resource-constrained systems. On the test set, the task-specific fine-tuned model achieves a perfect sequence accuracy of 100% and 100% position accuracy, while the LoRA-based model achieves a near-perfect sequence accuracy of 98.51% and position accuracy of 98.88%.

To validate the real-world applicability of these models, we integrate them into the ROS2 Humble framework and deploy them in a TurtleBot3 simulation using Gazebo. The robot successfully interprets high-level natural language commands and autonomously navigates through multiple waypoints, demonstrating the feasibility and practicality of the proposed approach.

Our key contributions are as follows:

1. Fine-Tuning T5-Small for Robotic Navigation: We fine-tune the T5-Small model to

translate natural language instructions into structured navigation plans.

2. Performance Comparison with LoRA: We compare full fine-tuning with LoRA-based fine-tuning, showing that LoRA achieves near-perfect accuracy with significantly reduced computational requirements.

3. Real-World Validation: We validate the performance of both models through integration with ROS2 and demonstrate successful autonomous navigation in a simulated TurtleBot3 environment.

Our work shows the potential of lightweight fine-tuning techniques like LoRA for effective deployment on resource-constrained systems and emphasizes the viability of employing transformer-based models for natural language-driven robotic navigation.

2 Dataset

2.1 Data Preparation

We created a task-specific extensive dataset of 24,581 navigation instructions to train and evaluate the T5-Small model to meet the particular requirements of our project. In order to simulate navigation scenarios including sequences of one to five battery colors (red, green, blue, orange, and purple), a skeleton template of 251 sentences with placeholders was used at the start of the procedure. Each sentence in the template is applied with permutations to generate that particular sentence with every possible color order, producing a wide range of comprehensive and varied combinations of instructions.

The dataset was split into:

- **Training Set:** 19,801 navigation instructions used for fine-tuning the T5-Small model.
- **Validation Set:** 4,781 instructions used to evaluate model performance during training.

The T5-Small model was able to generalize well because of the variation in sentence forms and battery sequences that guarantee reliable performance across untested navigation instructions.

2.2 Test Dataset Preparation

To avoid data leaking, the test dataset was generated ***after*** finishing the training and validation stages. It contained additional instructions created using human input obtained from many sources, including friends and real-world examples, even though it was built on the same skeleton template concept. To make these instructions more like genuine, human-like commands, they were expanded and rephrased with contextual diversity.

Using 60 skeleton sentences, this independent test dataset provided a rigorous standard to assess the generalization abilities of both the LoRA-adapted and fine-tuned models in a variety of realistic scenarios.

Table 1: ***Sample of Navigation Instructions and Color Order***

Input Label	Battery Count	Output Label
Reach blue battery, which is preceded by red battery and followed by orange battery; this sequence is subsequently succeeded by purple battery and ultimately by green battery.	5	Red ; Blue ; Orange ; Purple ; Green
While green battery is mentioned first, you must start at purple battery, visit blue battery, proceed to red battery, and only then reach green battery.	4	Purple ; Blue ; Red ; Green
Your journey ends at orange battery, but it starts at purple battery and continues through blue battery before reaching the final point.	3	Purple ; Blue ; Orange
Your task is to visit red battery first and then move to blue battery.	2	Red ; Blue
Visit the orange battery to perform a quick inspection.	1	Orange

3 Methodology

3.1 Model Architecture

This project converts natural language instructions into structured navigation commands using the T5-small transformer model. A more compact and effective variant of the original T5 architecture, the T5-small model is made to manage text-to-text operations with fewer resources while maintaining high performance. The model has an encoder-

decoder structure, with several layers in both the encoder and the decoder. In particular, T5-small is lighter than bigger T5 versions due to its 6 layers in both the encoder and decoder.

Each layer in the encoder and decoder contains a hidden size of 512, which refers to the dimension of the embeddings and the intermediate representations within the model. This smaller hidden size reduces the computational complexity without significantly sacrificing performance. Additionally, the feed-forward networks in each layer are designed with a size of 2048 units that ensures that the model can process and transform information efficiently.

T5-small also uses 8 attention heads in its multi-head attention mechanism, which allows the model to focus on different parts of the input sequence simultaneously. These architectural choices make T5-small a computationally efficient variant, capable of handling tasks like translation, summarization and question answering, while being suitable for environments with limited computational resources.

3.2 Model Understanding

To understand t5-small's capabilities and performance for our task, we evaluated it on a relatively easy navigation prompt and a relatively more complex prompt from our training dataset. Then, we further evaluated it by adding our formatting requirement to each navigation prompt. Figure 1 shows the navigation prompts that we used.

```
--- First prompt ---
Input (with format request): Go to the green battery first, then go to the red battery, followed by the orange battery,
blue battery and purple battery.
Just give me the color order in this way: Go to [color1]; Go to [color2]

Input (without format request): Go to the green battery first, then go to the red battery, followed by the orange battery,
blue battery and purple battery.


--- Second prompt ---
Input (with format request): Before going to purple battery, start your task at green battery and make sure to visit blue battery along the way.
Then go to the red battery, and wrap up at the orange battery.
Just give me the color order in this way: Go to [color1]; Go to [color2]

Input (without format request): Before going to purple battery, start your task at green battery and make sure to visit blue battery along the way.
Then go to the red battery, and wrap up at the orange battery.
```

Figure 1: Navigation prompts extracted from the training dataset and with added formatting requirement used to understand t5-small's capabilities and performance.

3.3 Model Training

3.3.1 Few-shot Learning

We performed few-shot learning to further study and understand the model's capability to generalize and comprehend our task when fed a small amount of task-specific data. To provide the model with a reasonable understanding of our task and its complexity spectrum, we trained the model with five navigation prompts evenly distributed across our training dataset (Figure 2), ensuring coverage from simpler prompts at the beginning to more complex ones toward the end.

```
Input: Go to the green battery first, then go to the red battery, followed by the orange battery, purple battery and blue battery.
Output: green; red; orange; purple; blue

Input: Set out to the red battery, then proceed to the green battery. Continue to the blue battery next, then purple battery, and finally wrap up the task at orange battery.
Output: red; green; blue; purple; orange

Input: Before you end at green battery, commence at orange battery, stop by red battery, make your way to purple battery, yet be sure to visit blue battery beforehand.
Output: orange; red; blue; purple; green

Input: Advance to orange battery, following on the heels of purple battery and leading into blue battery; this series then progresses to red battery, ending with green battery.
Output: purple; orange; blue; red; green

Input: red battery is your final stop, but to get there, you must start at blue battery, visit purple battery next, and proceed to orange battery.
Output: blue; purple; orange; red
```

Figure 2: Navigation prompts extracted from the training dataset and used to perform few-shot learning.

3.3.2 Task-Specific Fine-Tuning

A unique, variable dataset created to mimic actual navigation situations for a robotic system was used to refine the T5-Small model. More than 24,581 navigation instructions of various levels of complexity were included in the dataset. These included multi-step navigation with conditional routing as well as basic approaches like switching between consecutive batteries.

To fine-tune the T5-Small model to meet task-specific criteria, **all of its 60,727,808 parameters have to be trained.** The model was modified to produce precise navigation directives based on abstract textual inputs by utilizing its previously learned capabilities. The method for fine-tuning is shown in Figure 3.

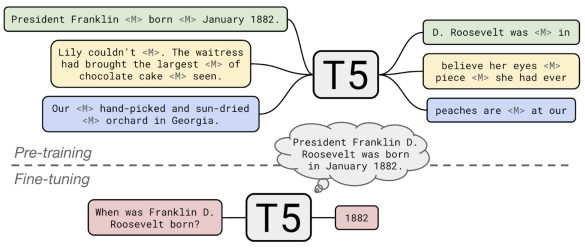

Figure 3: Fine Tuning of T5 small

Training Configuration Task-based Fine-tuning was conducted using Hugging Face's `Seq2SeqTrainer` framework. In training, we utilized the AdamW optimizer with a learning rate of 2×10^{-5}, a batch size of 16 for both training and validation, and cross-entropy loss as the objective function. The model was trained for 25 epochs to achieve effective task-specific adaptation.

3.3.3 LoRA-Based Fine-Tuning

The T5-Small model was modified for the task using Low-Rank Adaptation (LoRA), which dramatically decreased the number of trainable parameters without sacrificing performance. By adding

low-rank matrices to the transformer layers of the model, LoRA allows for parameter-efficient fine-tuning (PEFT) with minimal model modification.

To reduce the overfitting, our implementation employed a LoRA configuration with rank $r = 6$, a scaling factor of 32, and a dropout rate of 0.1. In comparison to comprehensive fine-tuning, this resulted in a considerable reduction in the trainable parameters to 0.36% of the total. **The total trainable parameters in the T5-small model are 60,727,808. We brought it down to 221,184 trainable parameters (0.36% of the total) using LoRA.**

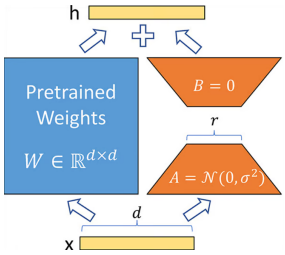

Figure 4: Low-Rank Adaptation (LoRA) methodology illustrating the incorporation of low-rank matrices into pre-trained weights (5).

The fine-tuning was carried out on the created dataset using the Hugging Face 'transformers' library. The training utilized an AdamW optimizer with a learning rate of 2×10^{-5} and a batch size of 16 for 25 epochs. A Seq2SeqTrainer was used for managing training and validation workflows, while the 'DataCollatorForSeq2Seq' handled padding and batching.

Key implementation details:

- The LoRA configuration and adapter layers were added using the PEFT library.
- The optimizer only updated parameters associated with the LoRA layers, keeping the rest of the model frozen.
- Evaluation metrics such as sequence and position accuracy were calculated using a custom function to validate model performance.

This approach utilized the model's pre-trained information to adjust to task-specific requirements while enabling effective fine-tuning on resource-constrained systems.

3.4 Simulation

The simulation shows how a well-tuned natural language processing model can be seamlessly integrated with a robotic navigation system. It also shows how the robot can understand natural language orders and navigate via pre-established waypoints in a realistic setting. The simulation makes use of complex mapping and navigation methods in addition to ROS2, Gazebo, and the TurtleBot3 software.

3.4.1 Environment Setup

The environment consists of a map generated using the Cartographer package, which provides the necessary framework for localization and navigation. The map is populated with five batteries, each assigned a unique color (red, green, blue, orange, purple), serving as waypoints for the robot. These batteries act as landmarks that the robot must visit based on user commands.

3.4.2 Natural Language Command Processing

The process begins with a user providing a high-level instruction, such as "Go to the red battery and then go to the green battery." This command is processed by the **fine-tuned T5-Small model**, designed to interpret and translate natural language instructions into actionable navigation goals. The model produces a sequence of target identifiers (e.g. 'red, green') that represent the order in which the robot must visit the batteries.

3.4.3 Transforms Broadcasting and Localization

The position of each target is continuously broadcast on the ROS2 TF tree in order to locate the batteries in the simulated environment. Each battery's relative position in relation to the "map" frame is updated in real time via a transformed system. Throughout the simulation, this broadcasting makes sure the robot can dynamically acquire the batteries' spatial locations, enabling precise localization for navigation.

3.4.4 Waypoint Generation

The sequence of battery colors generated by the NLP model is used to query the positional data of each corresponding battery. These positions are extracted from the TF tree by referencing the specific frames associated with each battery. The retrieved coordinates are converted into a series of waypoints in the form of a 'PoseArray', representing the robot's navigation path. The waypoint

generation process ensures that the robot visits the targets in the specified order.

3.4.5 Autonomous Navigation

The generated waypoints are published in the robot's navigation system, which uses the ROS2 'FollowWaypoints' action interface to execute the path. The robot is initialized with a predefined starting position in the "map" frame to ensure accurate localization. The navigation stack manages path planning, obstacle avoidance, and real-time adjustments to guide the robot through each waypoint.

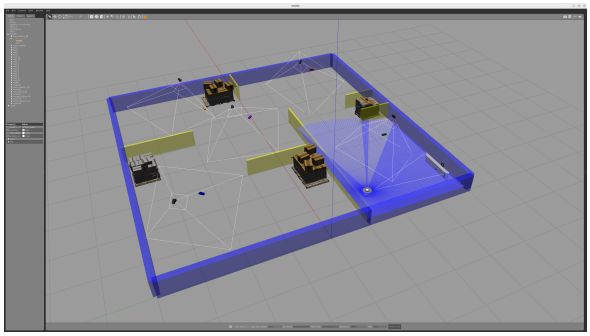

Figure 5: Gazebo Simulation Environment

3.4.6 Outcome

The simulation demonstrates the practical application of NLP in robotics, highlighting the system's ability to: Translate abstract natural language instructions into precise navigation commands; Execute multi-step navigation tasks with high accuracy [Figure 4]; and Adapt to dynamic environmental conditions in a simulated world [Figure 3]. To view the video results of the testing simulations and observations follow the link **https://youtu.be/v8OAqORMvxo**

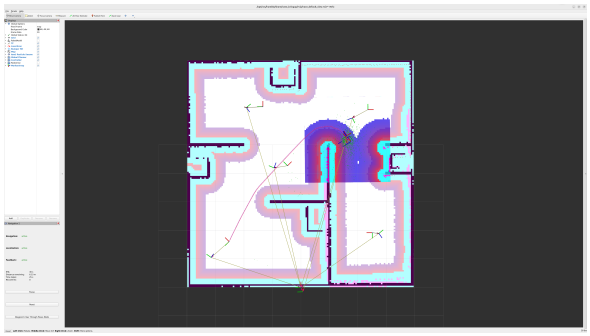

Figure 6: Path planning and autonomous navigation visualized on Rviz

4 Results and Discussion

Our work demonstrates the following. First, t5-small, as a baseline, performs poorly when fed a few navigation prompts. Second, t5-small shows some ability to generalize when fed a few navigation prompts after few-shot learning. Finally, fine-tuning a T5-Small model with synthetic task-specific datasets and employing LoRA-based adaptation results in robust and efficient navigation capabilities for robots. Below, we present an analysis of the results and their implications for task-specific navigation and ROS 2 integration.

4.1 Evaluation of Methods and Results

We evaluated four methods: the baseline T5-Small model, few-shot learning, task-specific fine-tuning, and LoRA-based fine-tuning. Table 2 provides a comparison of training time.

Table 2: **Training Time for Task-Specific Fine-Tuning and LoRA**

Method	Training Time (s)
Baseline	Not Applicable
Few-shot Learning	336.1
Fine-Tuned T5-Small	4577.2
LoRA-Based Fine-Tuning	3415.0

4.1.1 Baseline

The baseline T5-Small model failed to understand the navigation task, achieving a sequence and position accuracy of **0%**. Figures 7 and 8 illustrate these results.

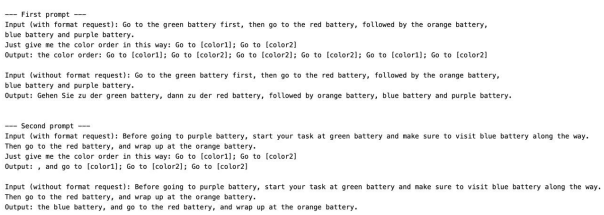

Figure 7: Baseline evaluation results showing the failure to identify task requirements.

```
100%|██████████| 299/299 [07:26<00:00,  1.49s/it]
Sequence Accuracy: 0.0000
Position Accuracy: 0.0000
Total sequences evaluated: 4780
Total positions evaluated: 22620
```

Figure 8: Baseline Evaluation Results: Sequence and Position Accuracy of 0%.

4.1.2 Few-Shot Learning

Few-shot learning with minimal examples also performed poorly, achieving a sequence and position accuracy of **0%**, as illustrated in Figures 9 and 10.

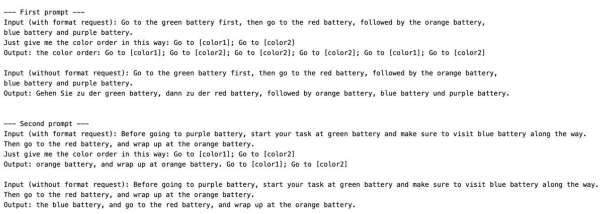

Figure 9: Few-shot evaluation results showing almost identical results as the baseline evaluation.

```
100%|██████████| 299/299 [09:48<00:00,  1.97s/it]
Sequence Accuracy: 0.0000
Position Accuracy: 0.0000
Total sequences evaluated: 4780
Total positions evaluated: 22620
```

Figure 10: Few-shot Evaluation Results: Sequence and Position Accuracy of 0%.

However, while the few-shot evaluation results may seem identical to the baseline evaluation results, we notice that the output from the second prompt with the format request in Figure 9 does differ. Indeed, while the baseline's output starts with " , and go to [color1]", the output from the few-shot learning approach starts with 'orange battery'.

So, although the battery color is incorrect, its format resembles the format that we would expect for the first position in our output sequence. This shows that the model can generalize, but requires a more substantial task-specific fine-tuning to produce good results.

4.1.3 Task-Specific Fine-Tuning

The task-specific fine-tuning of the T5-Small model resulted in significant improvements, achieving **0.9498** sequence accuracy and **0.9735** position accuracy on the validation set. On the independent test dataset, it reached perfect accuracy (1.0000 for both metrics). The model's smooth loss convergence (Figure 11) highlights its robust learning capabilities. **We used i9-13900HX with 4070 RTX graphics for training**

Training took **4577.2 seconds** with a final training loss of **0.0216**. The model's performance is attributed to:

- The diverse synthetic dataset, including varied sentence structures.
- Full fine-tuning of all model parameters, allowing the model to adapt fully to task-specific nuances.

Figure 11: Training and Validation Loss Curve for Fine-Tuned T5 Model.

```
100%|██████████| 299/299 [00:23<00:00, 12.58it/s]
Sequence Accuracy: 0.9498
Position Accuracy: 0.9735
Total sequences evaluated: 4780
Total positions evaluated: 22620
```

Figure 12: Validation Results: Sequence Accuracy of 0.9498 and Position Accuracy of 0.9735 for Fine-Tuned Model.

```
100%|██████████| 299/299 [00:28<00:00, 10.37it/s]
Sequence Accuracy: 1.0000
Position Accuracy: 1.0000
Total sequences evaluated: 4781
Total positions evaluated: 19430
```

Figure 13: Test Results: Sequence Accuracy of 1.000 and Position Accuracy of 1.000 for Fine-Tuned Model.

4.1.4 LoRA-Based Fine-Tuning

The LoRA-based fine-tuning method achieved competitive results, with sequence accuracy of **0.9404** and position accuracy of **0.9735** on the validation set. On the test set, it delivered near-perfect results (**0.9851** sequence accuracy and **0.9888** position accuracy).

The major advantage of LoRA is its computational efficiency:

- Training time was reduced to **3415.0 seconds**, which is 25% faster than full fine-tuning.

- Trainable parameters were limited to only **0.36%** of the total model parameters.
- The final training loss was slightly higher at **0.1510**, but this did not significantly affect the performance.

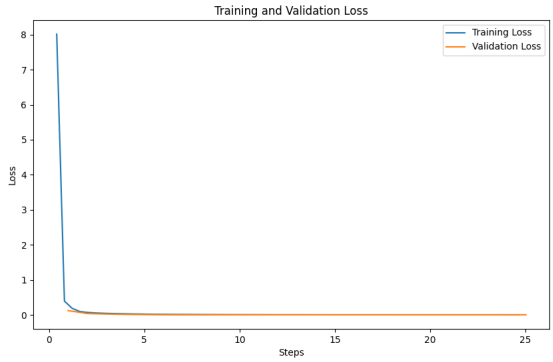

Figure 14: Training and Validation Loss Curve for LoRA-Based Fine-Tuning.

```
100%|██████████| 299/299 [00:31<00:00,  9.46it/s]
Sequence Accuracy: 0.9404
Position Accuracy: 0.9735
Total sequences evaluated: 4780
Total positions evaluated: 22620
```

Figure 15: Validation Results: Sequence Accuracy of 0.9404 and Position Accuracy of 0.9735 for LoRA-Based Fine-Tuning.

Figure 14 demonstrates the convergence of the training and validation loss, which aligns with the efficiency observed in LoRA-based training.

```
100%|██████████| 299/299 [00:51<00:00,  5.84it/s]
Sequence Accuracy: 0.9851
Position Accuracy: 0.9888
Total sequences evaluated: 4781
Total positions evaluated: 19430
```

Figure 16: Test Results: Sequence Accuracy of 0.9851 and Position Accuracy of 0.9888 for LoRA-Based Fine-Tuning.

4.2 Comparative Analysis

Both fine-tuning approaches effectively addressed the task:

- **Fine-Tuned T5-Small:** Achieved perfect accuracy but required more computational resources and longer training time.

```
Input: Before going to purple battery, start your task at green battery and
make sure to visit blue battery along the way. Then go to the red battery,
and wrap up at the orange battery.
Output: green; blue; red; orange; purple
```

Figure 17: Example of Natural Language Input and Output Generated by the Model.

- **LoRA-Based Fine-Tuning:** Provided nearly equivalent results with significantly reduced training time and trainable parameters, making it ideal for resource-constrained systems.

The use of an independent test dataset further validated the models' ability to generalize to realistic, human-like instructions. Figure 17 showcases an example of the natural language input and corresponding output generated by the fine-tuned models.

5 Conclusion and Future Scope

This project successfully fine-tuned the T5-Small model to translate natural language commands into structured navigation plans for robotic systems. By leveraging task-specific datasets and Low-Rank Adaptation (LoRA), we achieved high accuracy with reduced computational requirements. The system was validated in a ROS2-Gazebo simulation, where a TurtleBot3 robot successfully navigated multi-step instructions, demonstrating the feasibility of transformer-based models for robotic control.

Future improvements include:

1. Speech-to-Text Integration: Enable voice command inputs for a more natural interface.
2. Machine Translation: Support multi-language instructions for broader usability.
3. Reinforcement Learning: Allow the robot to autonomously learn and discover waypoints instead of relying on predefined coordinates.
4. Visual Perception: Combine text-based navigation with visual object recognition for adaptive navigation.

Team Contributions

This project was a collaborative effort, with each team member contributing to different aspects:

- **Adrien Ramsamy:** Developed and implemented the evaluation framework used across all approaches. Implemented few-shot learning, including examples selection and model

training. Also, jointly conducted training for both the task-specific and LoRA-based fine-tuned models with different hyperparameters.

- **Akhil Javvadi:** Implemented the fine-tuning of the T5-small model and conducted hyperparameter optimization for best performance.
- **Sai Jagadeesh Muralikrishnan:** Conducted training and validation of both task-specific fine-tuned and LoRA based models in personal local system (used i9-13900HX equipped with 4070 RTX) to derive metrics, analyzed to further optimize the methods and documented in report. Also, created and collected the synthetic test data set to test models.
- **Suhas Nagaraj:** Worked on the end-to-end integration of the trained model output with ROS 2 for autonomous navigation simulation of the TurtleBot in the Gazebo environment. Additionally, tested the models using a designated test dataset to evaluate their performance.
- **Varun Lakshmanan:** Created the synthetic dataset for training and validation, ensuring the availability of diverse dataset with task-specific instructions.
- **Vijay Dev Reddy Chevireddi:** Designed and developed the entire T5-small model finetuning and LoRA-based adaptation for efficient fine-tuning with reduced computational cost.

References

[1] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[2] Hugging Face, "T5-small pretrained model on Hugging Face Transformers," Available: https://huggingface.co/t5-small.

[3] Google Research, "Text-to-Text Transfer Transformer (T5)," Available: https://github.com/google-research/text-to-text-transfer-transformer.

[4] P. Anderson *et al.*, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," in *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[6] BringMeASpoon, "Natural Language Navigation Tools," Available: https://bringmeaspoon.org.

[7] P. Anderson, "Matterport3D Simulator," Available: https://github.com/peteanderson80/Matterport3DSimulator.

Acknowledgments

We would like to express our sincere gratitude to Professor Dr. Zeid Kootbally, instructor of the course *Introductory Robot Programming* (ENPM 702, formerly ENPM809Y), for permitting us to use the Gazebo world setup with battery checkpoints as the base framework for our implementation and testing. This foundational setup played a crucial role in validating our model's performance within a simulated environment. We would also like to acknowledge the Maryland Applied Graduate Engineering (MAGE) program at the University of Maryland for providing us with the resources and tools necessary to complete this project.