CIS 6200 Spring 2024, Final Project Report

Vision-Language-Action Navigation

GPTeam Members:

- Jason Hughes; Email: `jasonah@seas.upenn.edu`
- Xiaoye Zuo; Email: `zuoxy@seas.upenn.edu`
- Deeksha Sethi; Email: `deesethi@seas.upenn.edu`

Abstract

Visual-Language Navigation (VLN) has emerged as a dynamic field of study, particularly following the efficacy demonstrated by contrastive learning techniques in multimodal contexts. While current leading approaches leverage CLIP to predict the next optimal action for an agent, they may miss out on leveraging the rich contextual and relational insights provided by large pre-trained vision and language models, thus limiting long-term planning capabilities. This paper introduces a novel three-dimensional contrastive learning framework encompassing Vision, Language, and Action modalities. The framework consists of three phases: pre-training, fine-tuning, and inference. During inference, the model receives a single image, a set of goal-directed instructions, and potential action sequences for navigating intermediate landmarks toward the desired goal. The model then outputs the optimal sequence of actions for goal attainment. Experimental evaluations conducted on the RxR benchmark dataset for VLN tasks demonstrate that our tuned model achieves a success rate of 43.1%, surpassing a baseline CLIP-ViL model with a success rate of 40.5%. By integrating action sequences within the contrastive learning space, our approach enhances the performance of existing models and augments their robustness for long-term planning scenarios.

1 Introduction

Imagine exploring a new urban environment, depending on the advice of locals: "Continue straight until you reach a minor crossroads. Then, turn left and proceed until you notice a yellow edifice on your right." These directions seamlessly integrate directional guidance with identifiable landmarks, facilitating human navigation by utilizing a mental map rich in visual cues such as the "yellow building". Similarly, the task of instructing an autonomous robot should emulate conversing with a person. With the advancements in large language models (LLMs) and vision-language models, a novel domain of vision-language navigation (VLN) models has emerged within robotics. These models intake data from a camera depicting the surrounding environment and language that either delineates the scene, describes a task to be accomplished, or a combination of both. The fusion of language and vision in VLN lends itself effectively to contrastive learning methods (CLIP) [8]. Although the current state-of-the-art employs CLIP to forecast the next optimal action for an agent, it overlooks the opportunity to exploit the comprehensive contextual and relational insights offered by large pre-trained vision and language models and plan over the longer term.

Consider a task such as "navigate to the kitchen and open a window" while the robot commences from a bedroom. Through vision, a VLN model discerns its location in a bedroom rather than a kitchen, thus possessing the essential information to chart a course to the bedroom door, which it should recognize. Predicting the optimal actions from a discrete action set proves overly simplistic and sluggish, necessitating multiple steps to reach the bedroom door. We propose a novel VLN model constructed upon CLIP capable of forecasting a path or sequence of actions rather than a singular action. We anticipate this approach will enhance the model's comprehension of the relationships between language, vision, and actions, as it is compelled to plan beyond visual input alone. This is achieved by embedding the path alongside the image and text and engaging in contrastive learning across all three embeddings. We include all of our code, scripts, and logs on our GitHub.[1]

[1] https://github.com/jhughes50/VLA-Nav

2 Related Work

The current state-of-the-art in vision-language navigation model is CLIP-Nav [2], which employs large language and vision encoder models to anticipate actions from a discrete action set. This is achieved by evaluating the similarity between task-specific language and the camera view in each potential direction the agent can move. Additionally, they introduce a sequential model capable of backtracking, essential for long-term planning. CLIP-ViL [9] also utilizes CLIP to train agents to select the next best actions from a discrete action set. While similar coarse planning is employed in [10], it does not utilize CLIP in training.

The incorporation of a shared language and vision embedding within a contrastive learning framework was pioneered in [8]. CLIP was initially adapted for vision-language navigation tasks in [9].

For the training and evaluation of our model, we leverage the Matterport3D simulator, which contains 3D scans of numerous houses [1]. We utilize utilities provided by [7] to extract images and paths from the simulator. Lastly, we utilize the RxR dataset [5], which supplies task-oriented text alongside associated paths and viewpoints within the Matterport3D simulator environments which are shown in Fig. 2.

3 Problem Formulation

Our project aims to explore the impact of multimodal learning involving vision, language, and action on overall navigation in both familiar and unfamiliar environments. During the training phase, our model processes a sequence of images captured from the agent's point of view (POV) along its trajectory, the corresponding instructions for that trajectory, and the ground truth actions taken from start to finish. We utilize a Vision Transformer as an encoder to embed our images, a RoBERTa text encoder for embedding our textual instructions, and a pre-trained encoder to embed our actions. We pre-train this action encoder along with an action decoder as an autoencoder. The model generates a series of poses corresponding to the agent's trajectory from start to finish as its output. Subsequently, we evaluate the performance of our agent based on the success rate of the generated trajectories.

Our first baseline model is a cross-modal attention model proposed by Jacob Krantz et al. [4]. In their approach, tokenized instructions are converted to GLoVE embeddings and processed by recurrent encoders for textual input. For visual input, RGB and depth observations are encoded separately using ResNet50 for feature localization and point goal navigation. The encoded instructions are utilized to attend to visual and depth features through an attention mechanism. Once the point goal is identified, a recurrent network is employed to predict an action. Their model achieves a success rate of 37% on the Val-Seen data of RXR and 29% on Val-Unseen data.

Our second baseline model is CLIP-Nav [2], which leverages CLIP for sequential decision-making. At each time step, the model extracts images from viewpoints after each of the four available actions (move forward, move backward, turn left, and turn right) and selects the image with the highest CLIP score given the current navigation context. CLIP-Nav iteratively runs until the goal point is reached. While CLIP-Nav demonstrates a zero-shot success rate of 4.56% on seen data and 5.79% on unseen data, it excels in the success rate weighted by the inverse path length, establishing the state-of-the-art (SOTA) performance in this metric.

Specifically, we employ CLIP-ViL[2] as a robust baseline for Vision-and-Language Navigation (VLN) tasks due to its integration of CLIP's powerful visual encoding capabilities with language understanding. In VLN tasks, where the objective is to navigate an environment based on verbal instructions aligned with visual cues, CLIP-ViL leverages the pre-trained CLIP model to enhance the understanding of complex visual scenes in relation to language descriptions. This approach allows for improved generalization across various settings without extensive domain-specific training. CLIP-ViL records a success rate of 40.5% in VLN tasks.

4 Methods

The training architecture draws inspiration from CLIP, leveraging its principles to construct a robust framework. Initially, we execute a three-dimensional embedding process encompassing potential paths, images,

[2]Our baseline of can be found at: https://github.com/jhughes50/CLIP-ViL

and textual data. This approach adopts a 3D contrastive learning paradigm, facilitating the computation of cosine similarities across these three dimensions. By doing so, the model encapsulates intricate relationships between paths, images, and text, enhancing its ability to interpret and synthesize complex multimodal inputs. Figure 1 shows the above-described architecture.

1. Model Architecture
   (a) Encoders
      i. Text Encoder: We encoded our text instructions using a RoBERTa [6] model pre-trained on a large corpus of text using a masked language modeling objective. RoBERTa's architecture contains a stack of transformer layers with multi-head self-attention mechanisms and feedforward neural networks. RoBERTa has the same architecture as BERT, but uses a byte-level Byte-Pair-Encoding as a tokenizer.
      ii. Image Encoder: We embedded our images using a Vision Transformer [3] model pre-trained on ImageNet-21k and fine-tuned on ImageNet2012. The Vision Transformer(ViT) takes in images as a sequence of fixed-size patches and positional embeddings and learns an inner representation of images for downstream tasks. The last hidden state of the classification layer can be seen as a representation of the entire image and we extract that as our image embedding.
      iii. Pose Encoder: We trained a small 5-layer feed-forward network and a large encoder based on the transformer encoder architecture. Each trajectory is interpolated up to 32 waypoints before being passed through the encoder. The loss for epoch of training the medium encoder is shown in fig 4. We show this plot since this the model we use for fine-tuning and evaluation.
   (b) Linear Layers: Since our image and text encoders, ViT-Base and RoBERTa respectively, output an embedding of 768, and our path encoder outputs an embedding of 512 we have two learned linear layers that bring the 768 embeddings down to 512 for similarity comparison. We also needed to tune the output pooling layers from ViT and RoBERTa as those are not trained in the pretraining.
2. Model Training
   (a) Data processing: each data entry contains an image corresponding to the starting view, a text instruction detailing the steps the agent needs to take, and a trajectory represented using a sequence of 3D positions. We normalize each embedded feature before calculating cosine similarities.
   (b) Contrastive loss: we calculate the cosine similarity score for each pair of image and path embeddings and text and path embeddings. The contrastive loss function compares the output logits to the ground truth label that indicates whether the pair belongs to the same instruction or different ones. The loss penalizes the model based on the discrepancy between the predicted similarity and the ground truth label. Since cosine similarity only gives the similarity between two vectors we need to aggregate the similarities across all pairs. We take the batch cosine similarity from [8] and add to it. First, we compute the logits $\mathcal{L}$ for each pair and use i, t and p to denote image, text, and path respectively, additionally, we use V to denote the embedding vector and T to denote the label vector:

$$\begin{aligned} \mathcal{L}_{it} &= V_i \cdot V_t^T \\ \mathcal{L}_{ip} &= V_i \cdot V_p^T \\ \mathcal{L}_{tp} &= V_t \cdot V_p^T. \end{aligned} \tag{1}$$

   Now we can use cross-entropy loss as [8] does.

$$\begin{aligned} \text{loss}_{it} &= (\texttt{cross\_entropy}(\mathcal{L}_{it}, T) + \texttt{cross\_entropy}(\mathcal{L}_{it}^T, T))/2 \\ \text{loss}_{ip} &= (\texttt{cross\_entropy}(\mathcal{L}_{ip}, T) + \texttt{cross\_entropy}(\mathcal{L}_{ip}^T, T))/2 \\ \text{loss}_{tp} &= (\texttt{cross\_entropy}(\mathcal{L}_{tp}, T) + \texttt{cross\_entropy}(\mathcal{L}_{tp}^T, T))/2 \end{aligned} \tag{2}$$

Finally, we average them all together to get one loss to backpropagate:

$$\text{loss} = (\text{loss}_{it} + \text{loss}_{ip} + \text{loss}_{tp})/3. \tag{3}$$

We also note that we tried computing similarity in other ways, like using the singular values of the covariance matrix, but ultimately saw similar results.

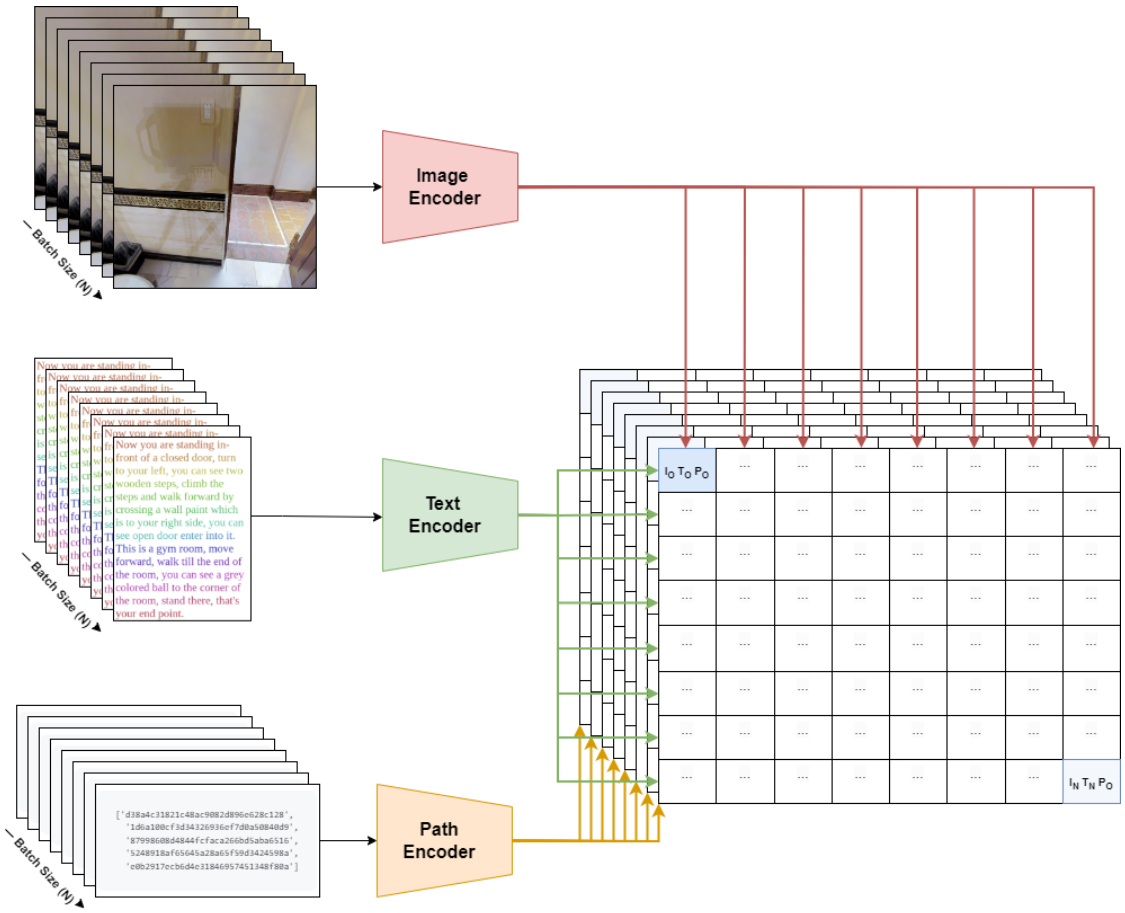

Figure 1: 3D Contrastive Training

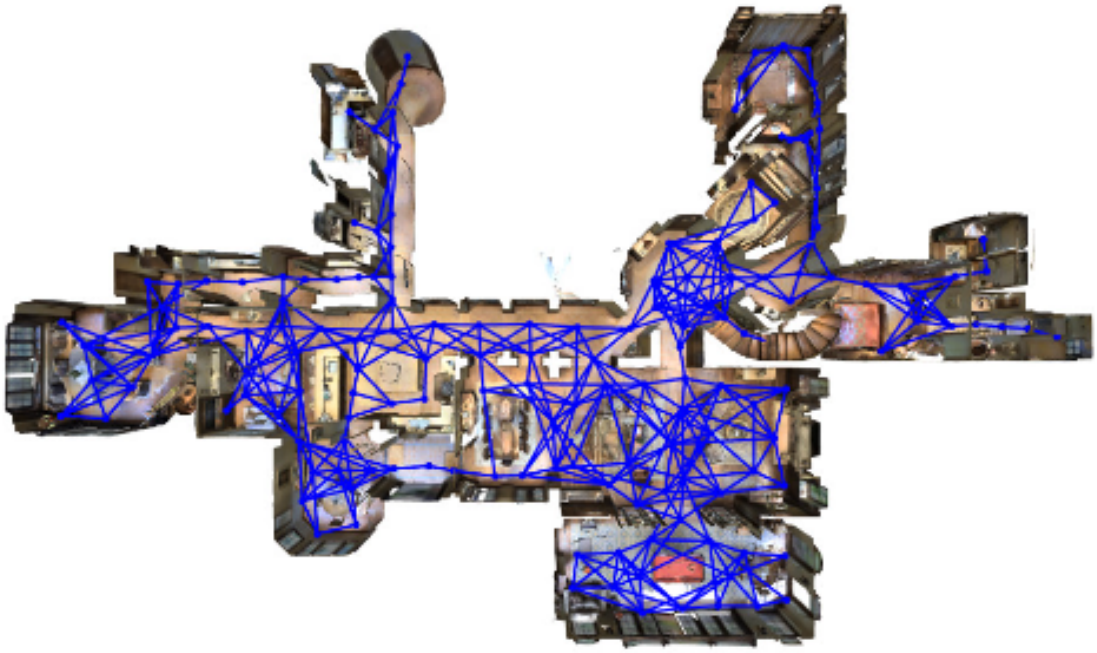

Figure 2: Depiction of the graph-like nature of the paths throughout an environment. Each node is a waypoint in the path and each edge means the agent can move between the connected nodes.

During inference, our model has two distinct stages. We consider all possible paths of reasonable length from the starting point looking at it like a graph of nodes and edges like that shown in Fig. 2. We embed the image and text and all possible paths. We then compute the similarity as in (**??**) between the embeddings and all the generated paths and take the path with the highest similarity score as the path the robot should follow. This architecture is shown in Fig. 3. The loss curve for the model pre-training is shown in Fig [**?**]

5 Evaluation

We assess the performance of our model through the analysis of several key metrics:

1. **Success Rate (SR):** This metric quantifies the efficacy of the agent in navigation tasks. It represents the proportion of episodes wherein the agent successfully reaches and halts at the designated target location. We consider a threshold of $5m$ as a success since our main focus is path planning based on a task, not task completion.

2. **Mean-Squared Error (MSE):** This metric quantifies how different the ground truth path is from the path output by the model. We also compute the distance between the final waypoint of the ground truth and the model output to see how far the agent is from its intended destination.

3. **Classifcation Success Rate (SR):** Finally, we show the classification success rate. This is similar to the success rate but requires the model to be more specific. Since contrastive learning works as a classification task, we show when the model classifies and chooses the ground truth path, as that is one of the possible paths in the batch.

We show the above metrics for our fine-tuned encoders and a zero-shot approach using just the pre-trained encoders in 5. We also show the results of CLIP-ViL using ViT-Base here since they use the same dataset as us. We can see that our architecture does slightly improve upon the results of CLIP-ViL. We achieve a success rate slightly higher than theirs. We also show a relatively low mean squared error of 0.10. This means that the model is doing a good job of finding a path that is at least close to the ground truth.

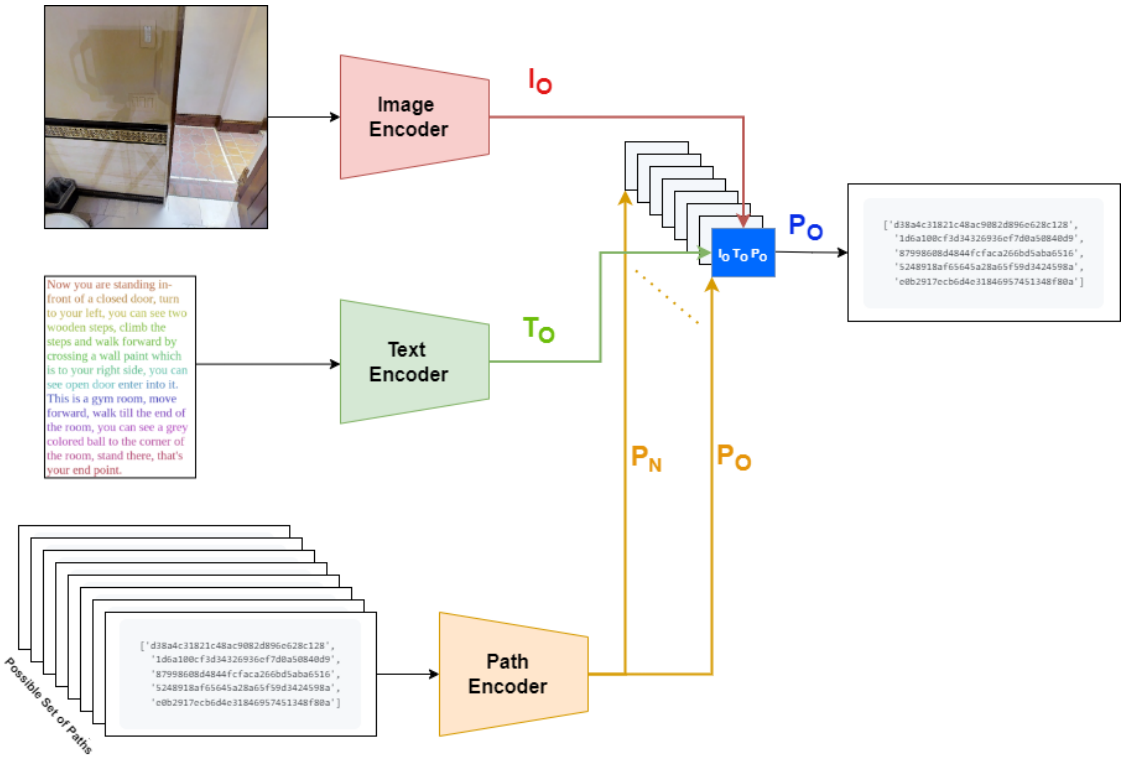

Figure 3: Inference Architecture for Best Path

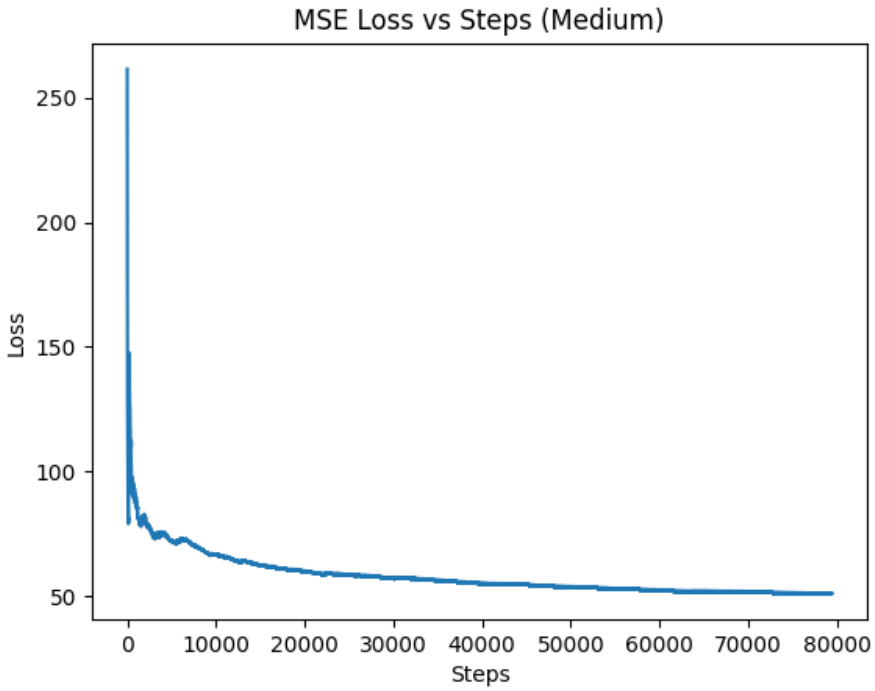

Figure 4: Average Mean-Squared Error (MSE) loss curve for path encoder pre-training.

Table 1: Depicting the metrics of our models and our baseline.

Metric	Success Rate	MSE	Classification SR
CLIP-ViL	40.5%	Not Reported	Not Reported
Ours zero-shot	8.9%	0.22	1.8%
Ours tuned	43.1%	0.10	38.2%

The success rate indicates that we are within $5m$ of the ground truth destination regardless of how we got there 43% of the time and 38% of the time predicted the path exactly right.

We also evaluate our model on the encoders without any fine-tuning to test their zero-shot performance. We see that our models do not perform well at zero-shot. With a batch size of 16, meaning we generate 15 random paths in addition to the real one, it barely outperforms random, meaning our fine-tuning helped the models learn similarity amongst the embeddings.

6 Discussion

During our project, we ran into a few hurdles that we wanted to discuss here. The first is the images. Since the agent exists in the Matterport3D environment we have to place an agent in the simulator to gather images of what it sees along the path. Placing an agent in the environment and gathering images takes as long as 20 seconds. We decided to take two approaches here. One was extracting the images at train time and simply training on less data. The other was to pre-extract the images at the cost of not having as many images, as extracting all the images would be 5TB. We additionally took approaches to training. When extracting images during training we trained on all combinations of images and paths. For example, if our path was $[1, 2, 3, 4]$, then path $[2, 3, 4]$ is also a viable path to the destination with similar task-oriented text. So we trained on all "sub-paths" and their associated starting images. We also hoped this would help the model be able to plan paths of greater varying length. The results shown in the previous section are from the approach where we extract the images at training time.

Finally, when tuning our model we did see strange behavior. When pretraining the transformer path encoder the learning was not stable and we had an exploding gradient problem. We used gradient clipping and different weight initialization to help with this but ultimately the loss did not go down as much as our smaller FFN model so we used that instead. When fine-tuning all three encoders, the loss went down slightly and then converged but we would have expected the loss to go down much more before converging. This is something we hope to fix in the next few days and show updated results if possible.

7 Conclusion

In this project, we highlight how we used contrastive learning across vision, language, and action embeddings to plan a robot's path based on task-specific language. We used the RxR dataset [5] to obtain task-oriented text prompts, and paths to the tasks within the Matterport3D [1] environments. We show that embedding the actions space allows us to predict paths with a relatively high success rate that is comparable the current state of the art. We also show that fine-tuning the models helps greatly and outperforms when we compare them to a zero-shot approach.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 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.

[2] Vishnu Sashank Dorbala, Gunnar Sigurdsson, Robinson Piramuthu, Jesse Thomason, and Gaurav S. Sukhatme. Clip-nav: Using clip for zero-shot vision-and-language navigation, 2022.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[4] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pages 104–120. Springer, 2020.

[5] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2020.

[6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[7] Xavi Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Ruslan Partsey, Jimmy Yang, Ruta Desai, Alexander William Clegg, Michal Hlavac, Tiffany Min, Theo Gervet, VladimiÌr VondrusÍ, Vincent-Pierre Berges, John Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[9] Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How much can clip benefit vision-and-language tasks? *arXiv preprint arXiv:2107.06383*, 2021.

[10] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment, 2018.