On Device Vision Language Navigation

Dhruv Naik
Carnegie Mellon University
Pittsburgh, PA 15213
drn@andrew.cmu.edu

Saloni Mittal
Carnegie Mellon University
Pittsburgh, PA 15213
salonim@andrew.cmu.edu

Thomas Xu
Carnegie Mellon University
Pittsburgh, PA 15213
twx@andrew.cmu.edu

1 Motivation

Enabling robotic agents to smoothly navigate through realistic 3D visual environments using natural language has been a long-standing challenge. In **vision-and-language navigation** (VLN) tasks, agents must first interpret a set of language instructions and then use visual perception information regarding its environment to carry out those instructions. In particular, these tasks require the agents to use visual perception to successfully navigate to specified locations. Given the extremely diverse nature of possible visual and language inputs, the generalization of VLN agents to unseen environments remains a substantial challenge.

However, there are many compelling reasons to develop VLN models and deploy them to edge devices. Most notably, there already exists significant commercial viability for VLN-empowered robots in today's markets. Examples include Samsung's robotic butlers Bot Care and Bot Handy and Hello Robot's home assistance robots, all equipped with the necessary compute and sensor modalities to perform VLN tasks in real-life for real consumers.

While these commercial use cases are encouraging, many of the state-of-the-art (SOTA) VLN models today remain too large to be deployed feasibly on edge devices. Therefore, our project aimed to apply compression techniques to current SOTA VLN models such that they could be run on a lightweight edge device (2GB Jetson Nano).

2 Task Definition

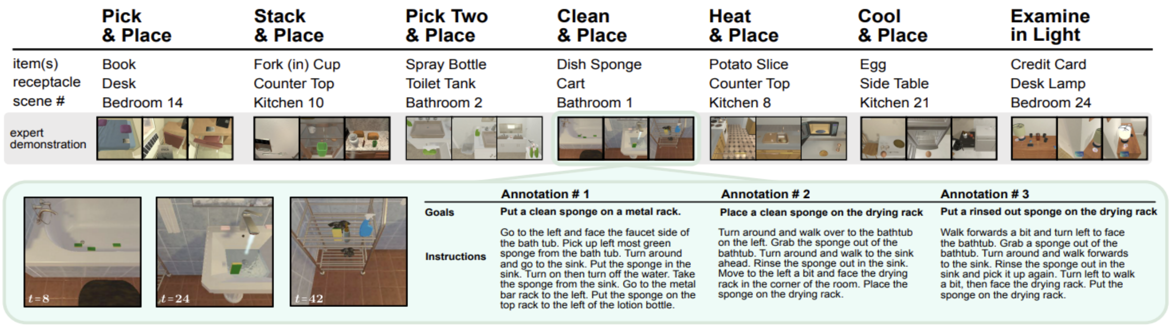
We make use of the ALFRED dataset (Shridhar et al., 2020a) to perform our experiments. ALFRED is a benchmark for mapping from natural language instructions and egocentric vision to sequences of actions for household tasks. It contains about 25000 action sequences based on natural language instructions (Figure 1). The instructions given are of two different types: 1. High level goal instructions (”Slice the potatoes and place on plate”), (2) Low level instructions (”Take a left from the hall, pick up the knife from the drawer”).

ALFRED is the first dataset to provide high low level instructions and test the model's ability to complete a given set of sub-goals and tasks. The dataset also benchmark's a model's commonsense reasoning by testing object positions and state change knowledge. The starting positions of objects are based on common knowledge (a spoon can be inside a drawer, but an apple will not), and the simulator enables object state changes (potatoes can be slices, but a fork cannot). The dataset also has visual variations such as different shapes, sizes and textures for class objects.

3 Related Work

Vision-language navigation has had numerous advances in recent years. Many of these advances have come about thanks to new datasets of realistic 3D environments as well as corresponding simulators to simulate agents navigating through those environments. Some notable examples include the Matterport Simulator and Habitat simulation platform.

There has also been significant advancement due to the development of large visiolinguistic transformer-based models; these models are pretrained on large image-text pairs from the web and apply fine-tuning to improve performance on VLN tasks. One such pretrained model is AirBert (Guhur et al., 2021) which is trained on millions of VLN path-instruction (PI) pairs and was released in 2021. It uses BnB (a large scale VLN dataset created from AirBnb data) for pretraining and outperforms the state-of-the-art for the Room-to-Room (R2R) navigation and Remote Referring Expression (REVERIE) (Qi et al., 2020) benchmarks in



	Pick & Place	Stack & Place	Pick Two & Place	Clean & Place	Heat & Place	Cool & Place	Examine in Light
item(s)	Book	Fork (in) Cup	Spray Bottle	Dish Sponge	Potato Slice	Egg	Credit Card
receptacle	Desk	Counter Top	Toilet Tank	Cart	Counter Top	Side Table	Desk Lamp
scene #	Bedroom 14	Kitchen 10	Bathroom 2	Bathroom 1	Kitchen 8	Kitchen 21	Bedroom 24
expert demonstration							

	Annotation # 1	Annotation # 2	Annotation # 3
Goals	Put a clean sponge on a metal rack.	Place a clean sponge on the drying rack	Put a rinsed out sponge on the drying rack
Instructions	Go to the left and face the faucet side of the bath tub. Pick up left most green sponge from the bath tub. Turn around and go to the sink. Put the sponge in the sink. Turn on then turn off the water. Take the sponge from the sink. Go to the metal bar rack to the left. Put the sponge on the top rack to the left of the lotion bottle.	Turn around and walk over to the bathtub on the left. Grab the sponge out of the bathtub. Turn around and walk to the sink ahead. Rinse the sponge out in the sink. Move to the left a bit and face the drying rack in the corner of the room. Place the sponge on the drying rack.	Walk forwards a bit and turn left to face the bathtub. Grab a sponge out of the bathtub. Turn around and walk forwards to the sink. Rinse the sponge out in the sink and pick it up again. Turn left to walk a bit, then face the drying rack. Put the sponge on the drying rack.

Figure 1: ALFRED Annotations (Shridhar et al., 2020a)

simulated indoor environments. The BnB dataset, introduced by this work, was made openly available and can be utilized in the future for other VLN-related research.

There has also been recent work in this area from industry; for example, Facebook Research recently developed VLN-BERT (Majumdar et al., 2020): a visiolinguistic transformer-based model for VLN tasks. In addition to outperforming other contemporary models on certain tasks, the authors showed that pretraining on image-text pairs scraped from the web followed by fine-tuning on embodied path-instruction data led to significantly improved performance.

There has been little prior work on **compressing large-scale visual-linguistic transformers** but many model compression techniques that are applied to large transformer-based models are mainly task and domain agnostic. In *Fang et al.* (Fang et al., 2021), the authors used knowledge distillation techniques at various stages of the model as was first proposed in *Jiao et al.*(Jiao et al., 2020) to facilitate training of smaller vision-language (VL) models. They trained the student model to match the attention, hidden layer and classification distributions of the teacher model with an MLM learning objective. The proposed DistillVLM model significantly improves performance of small VL models for VL tasks, such as image captioning and visual question answering over their non-distilled counterparts. A major challenge highlighted by the authors was the inconsistent regional visual tokens extracted from different detectors of teacher and student models, resulting in the misalignment of hidden representations and attention distributions. To address this problem, they performed visual token alignment by retraining the Teacher by using the same region proposals from the Student's detector while the features came from the Teacher's own object detector.

More recently, a large survey paper (Ganesh et al., 2021) was published which discussed compression techniques for large-scale transformer models. It also conducted a case study on BERT, giving a detailed analysis and comparison of various BERT compression methods while offering several insights into the practical use of these methods and their potential future directions.

4 Results and Discussion

We experimented with two SOTA models for this project: Seq2Seq and HiTUT.

4.1 Seq2Seq-LSTM

The first baseline model we experimented with is a LSTM based Seq2Seq model presented in the ALFRED (Shridhar et al., 2020a) paper as a baseline for the benchmark.

The model uses an LSTM Encoder to create language instruction representations for the low level and high level instructions. The language encoder also makes use of attention to weight the words in the instruction, conditioned on the decoder embeddings from the previous time step.

At each time step, the visual encoder generates visual representation of the current observation of the agent using a ResNet-18 based model with a linear layer.

The language, visual representations along with the action from the previous time step are passed to an LSTM based decoder to generate the hidden representation for time step *t*.

The inputs of the decoder, together with the hidden representation output, are used to make the final predictions. For each time step, the model generates outputs for two different tasks. Firstly,

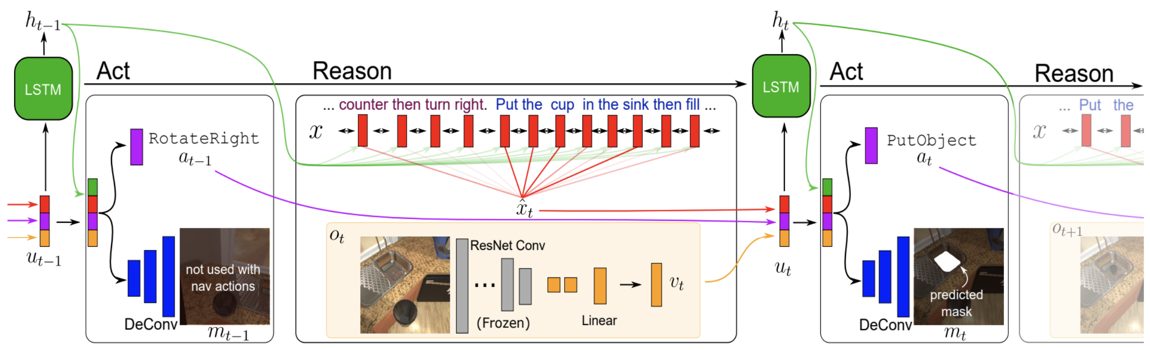


Figure 2: Seq2Seq Architecture (Shridhar et al., 2020a)

the combined features are passed to a linear layer to generate the next action (a_t). Secondly, the model generates a pixel-wise interaction mask for the observed image, to indicate an object to interact with, using a 3-layer deconvolution network. The authors also explained this architecture through a visualization, as given in Figure 2.

4.1.1 Quantization and Inference Latency

The original model is of size 176 MB, and dropped down to approximately 43 MB after INT8 quantization, with almost no drop in performance and loss scores. Even though the model size reduced by 75%, we did not notice any reduction in inference latency, which was approximately 8 seconds for completing a single task sequence.

4.1.2 Energy and Carbon

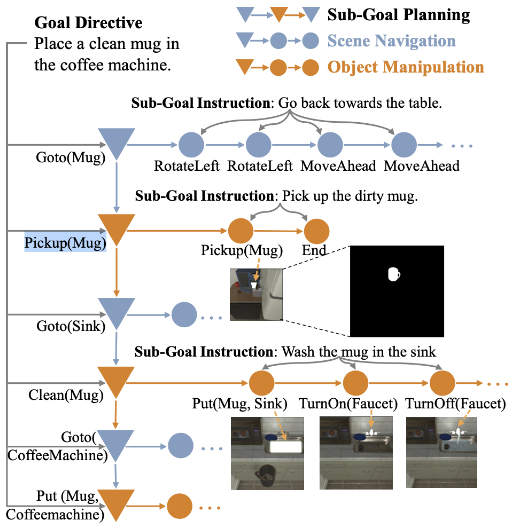
In addition to the the above results, we also estimated average energy and carbon use of the Seq2Seq model when deployed on device. On average, we observed an energy consumption rate of 6.4 W with an average inference time of 21.5 seconds. Thus, we estimate an energy use of approximately **137.6 J** per inference. In other words, on a typical phone battery with 1400 mAh capacity and 3.7 V rating (i.e., 5.18 Wh) approximately **136** inferences can be run.

During the project, the device was powered in a facility (Carnegie Mellon University campus, Newell-Simon Hall) within the RFCW region defined by the EPA eGRID standard, which has a carbon intensity of **1067.7 lbs CO2/MWh**. Thus, we estimate the corresponding carbon usage per inference for Seq2Seq on our device to be **4.08 * 10^{-5} lbs** (0.0276 g) CO2.

4.2 Transformer-based model

This section will first explain our second model and then discuss in detail the experiments performed to improve the computational efficiency of this model on the Jetson.

The authors of the paper (Zhang and Chai, 2021) developed a model HiTUT (stands for Hierarchical Tasks via Unified Transformers) that decomposes the ALFRED learning task into three sub-problems: sub-goal planning, scene navigation, and object manipulation. The model is trained with a RoBERTa-base backbone and comprises of 124.76 M parameters.



Goal Directive
Place a clean mug in the coffee machine.
Sub-Goal Planning
Scene Navigation
Object Manipulation
Sub-Goal Instruction: Go back towards the table.
Goto(Mug)
RotateLeft RotateLeft MoveAhead MoveAhead
Sub-Goal Instruction: Pick up the dirty mug.
Pickup(Mug)
Pickup(Mug) End
Goto(Sink)
Sub-Goal Instruction: Wash the mug in the sink
Clean(Mug)
Put(Mug, Sink) TurnOn(Faucet) TurnOff(Faucet)
Goto(CoffeeMachine)
Put (Mug, Coffeemachine)

Figure 3: Task example from ALFRED, Source: (Zhang and Chai, 2021)

As shown in Figure 3, a high-level goal directive ("place a clean mug in the coffee machine") can be decomposed to a sequence of sub-goals. Some sub-goals involve navigation in space (e.g., Goto(Mug), Goto(Sink)) and others require manipulation of objects (e.g., Pickup(Mug), Clean(Mug)). These

Actions	Models	Action Type Acc	Argument Acc	Mask Acc
Sub-goals	Original	0.982	0.934	NA
	Layer Pruning	0.980	0.914	NA
	Layer Prun. + Emb. Factor.	0.973	0.896	NA
	Dynamic Quantization	0.314	0.200	NA
Navi. Actions	Original	0.889	NA	NA
	Layer Pruning	0.863	NA	NA
	Layer Prun. + Emb. Factor.	0.854	NA	NA
	Dynamic Quantization	0.394	NA	NA
Mani. Actions	Original	0.996	0.991	0.969
	Layer Pruning	0.992	0.990	0.966
	Layer Prun. + Emb. Factor.	0.995	0.992	0.950
	Dynamic Quantization	0.15	0.27	0.189

Table 1: Results of the reduced HiTUT variants on the valid seen data of ALFRED; the higher the better

sub-goals can be further decomposed into navigation actions such as RotateLeft and MoveAhead, and manipulation actions such as Put(Mug, Sink), TurnOn(Faucet).

This model differs from the previous Seq2Seq LSTM-based model (Shridhar et al., 2020b) in the aspect where it does not apply recurrent state transitions, but feed the prediction history as the input to each subsequent prediction. It projects input from different modalities into the language embedding space, and adopt a transformer to integrate the information together. Multiple prediction heads are constructed on top of the transformer encoder to make predictions for the sub-goal type and argument, the action type and argument and object masks respectively.

Evaluation Metrics The evaluation was done using the expert demonstrations that come with the ALFRED dataset. We report the next-step prediction accuracy given the gold labels in expert actions. As discussed before, the model has prediction heads for each of the three sub-tasks: sub-goals, navigation actions and manipulation actions. Each sub-goal and manipulation action has two parts, action type (*e.g. Goto or PickUp*) and action argument (*e.g. Knife*). Each navigation action refers to the type of action from {*RotateLeft, RotateRight, MoveAhead, LookUp, LookDown*}. For a manipulation sub-task, in addition to the type and argument of the action, the model also needs to generate a segmentation mask on the current visual observation to indicate which object to interact with (i.e., which object the argument is grounded to). We also report the mask prediciton accuracy for this sub-task.

We adopted two parameter-reduction techniques to reduce the computation and memory costs of this model in order to run inference on the Jetson.

4.2.1 Reduction Technique 1: Layer Pruning

The authors in this work (Sajjad et al., 2020) showed how simple pruning strategies that do not require model pre-training from scratch can work equally well as knowledge distillation. After empirically evaluating various layer dropping strategies, the authors conclude that top layer dropping in BERT works the best and can yield comparable results to DistilBERT on GLUE tasks in terms of both model size and performance. Additionally, they also found that RoBERTa is more robust to pruning layers compared to BERT. Taking cue from this, we removed the top 6 layers from the original HiTUT model and fine-tuned the remaining half for 2 epochs. The hyper-parameters used while training are same as the ones used in the original paper.

The number of model parameters reduced from 124.76 M to 82.24 M which correponds to 34% reduction in model parameters. Table 1 shows the per-step prediction accuracy of the layer-pruned model on the three sub-tasks for valid seen. We observe no significant change in the Manipulation Action results when compare to the original model. However, a drop of -2.6% is seen in the navigation action accuracy. While the Sub-goal type accuracy remains unaffected, -2% drop in the sub-goal argument accuracy is observed.

4.2.2 Reduction Technique 2: Embedding Matrix Factorization

We realized a bottleneck in the memory requirements of the HiTUT model is the embedding layer. With a vocabulary size of 50,265 and hidden layer dimension size of 768, the embedding weight matrix alone constitutes of 38.6 M parameters which is 31% of the original model size. Inspired from the approach used in ALBERT (Lan et al., 2020), we use a factorization of the embedding parameters on the layer-pruned model, decomposing them into two smaller matrices. We first project one-hot vectors from a vocabulary of size V into a lower dimensional embedding space of size E, and then project it to the hidden space of size H. By using this decomposition, we reduce the embedding parameters from O (V × H) to O (V × E + E × H). This parameter reduction is significant when H >> E. In our case, we choose E to be 128 which results into (50,265 X 128 + 128 X 768) = 6.53 M embedding parameters, which is just 16% of the original embedding matrix size. Using this simple decomposition, we were able to reduce the model size further to 50.1M, which corresponds to a 60% reduction from the original model size.

We fine-tuned the factorized-embedding model for 3 epochs. Table 1 shows the results of our final layer-pruned plus factorized-embedding model on the three sub-tasks. Interestingly, we observe no loss in the manipulation action type and argument accuracy results. However, the mask prediction accuracy drops by -1.9%. The navigation action accuracy further drops and is -3.5 points behind the original model. This time the sub-goal action type accuracy drops by -0.9% and argument accuracy by -3.8% when compared to the original model.

4.2.3 Latency Comparison

Observed inference latency on the Jetson Nano is plotted against different batch sizes for HiTUT and its reduced variants in Figure 4. Our model reduction techniques, reduce the size of the network and speeds up inference time. For a batch size of 1, our best compressed model (LP+EF) runs an inference in 1.24 seconds while the original HiTUT model runs in 2.46 seconds. This corresponds to a 50% increase in the speed-up. This increase in speed-up is further amplified for larger batch sizes as seen in Figure 4. We observe comparable latencies for layer-pruned (LP) and layer-pruned plus factorized-embedding (LP+EF) versions of HiTUT. Please note that despite the smaller size of LP+EF

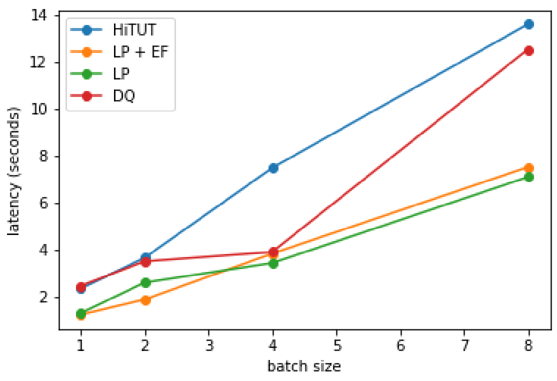


Figure 4: Inference latency in seconds versus the batch size for different HiTUT variants; LP=Layer-pruning, LP + EF: Layer-Pruning + Embedding Factorization, DQ: Dynamic Quantization

variant, the computational time is not affected because the embedding layer is O(1) in computational complexity. As a result, the embedding factorization technique contributes mainly in reducing the memory footprint of the model and not in speed-up.

4.2.4 Energy and Carbon

Again, in addition to the the above results we also estimated average energy and carbon use of the HiTUT model. When deployed on device, we observed an energy consumption rate of 8.1 W with an average inference time of 8.6 seconds on average. Thus, we estimate an energy use of approximately **69.66 J** per inference. In other words, on a typical phone battery with 1400 mAh capacity and 3.7 V rating (i.e., 5.18 Wh) approximately **268** inferences can be run. As with the Seq2Seq experiments, the HiTUT experiements were conducted within the RFCW region which has a carbon intensity of **1067.7 lbs CO2/MWh**. Thus, we estimate the corresponding carbon usage per inference for HiTUT on our device to be **2.07 * 10^{-5} lbs** (0.0094 g) CO2.

For the cloud (Google Cloud Platform) training of HiTUT, we performed a total of **36 hours** of computation on hardware of type Tesla K80 (TDP of 300W). This was done in the ”us-central1” region, which has a carbon efficiency of **0.57 kg CO2/kWh** (1.26 lbs CO2/kWh). Thus, we estimate the total emissions from our cloud usage to be **6.16 kg CO2** (13.58 lbs CO2), of which 100 percent was directly offset by the cloud provider.

5 Insights

Sub-task complexity in ALFRED: To summarize, our smallest HiTUT model after compression is 60% smaller than the original model. The highest drop of -3.5% is seen in the navigation action prediction accuracy. However, no performance loss is observed in manipulation action type and argument accuracy, while a slight drop of -1.9% is seen on segmentation mask prediction accuracy. The sub-goal action argument accuracy drops by -3.8% while no significant loss is seen in sub-goal action type prediction. Looking at these sub-task performance results, we can draw the following insights. Sub-goal planning and scene navigation require more network capacity to perform better than object manipulation. This could imply that planning and navigation are harder problems to solve than manipulation in ALFRED. The drop in mask prediction also shows that grounding the action argument to the visual input is a harder problem than predicting the type and argument of the manipulation action. We can also say that sub-goal action type prediction is a much simpler problem than predicting the sub-goal action argument.

Layer-pruning over teacher distillation? Our results from the layer-pruning experiment reinforce the claims made in (Sajjad et al., 2020). This is a simple yet effective technique that can drop half of the model layers without any significant loss in model performance. We believe that top-layer pruning is better than teacher knowledge distillation methods as it can be directly applied to trained models, without the need of any training from scratch. Note that building a DistilBERT model consumes a lot of computational power and GPU/TPU memory.

INT8 Dynamic Quantization: Quantization is a technique that comes in handy when deploying machine learning models on device. In our experiments with Seq2Seq LSTM, Quantization reduces model size by 75% but still maintains performance on the validation dataset. This is useful when working with devices under memory or storage constraints. With the right hardware, INT8 computations may also lead to inference speed gains. In the experiment with HiTUT, quantization on linear layers led to a drastic reduction (see table 1) in performance, which was unexpected and we have not been able to explain.

6 Challenges

Simulator: We tried working with different simulators across multiple datasets such as Habitat, MatterPort and AI2Thor. There were lots of difficulties with installation of these simulators on our personal laptops and on device. While AI2Thor was easier to work with, it was computationally infeasible to run the simulator and generate the Resnet features on the fly. Thus we made use of the expert action features provided with the ALFRED dataset to run validation in a teacher-forcing manner.

GPU OOM: With both the models, the CPU RAM utilization during validation was around 65-80%. We tried to load the models on the CUDA cores available on the Jetson with *.cuda()*, but noticed a sudden spike in RAM utilization which led to the ssh sessions ending abruptly. We deduce that the spike is due to the model being loaded onto the GPU, and the RAM being shared by CPU and GPU.

Library Installations: We came across several issues with package installations and version conflicts on-device. Several packages required by HiTUT had to be compiled by source and many version-conflicts had to be resolved manually. We also had issues with quantization on device, where PyTorch threw an error involving a NoneType qconfig, while the same code worked on our personal devices/cloud without errors.

Storage Constraints: Storage was limited on the device, and we were not able to work with multiple models/dataset splits at the same time. We had to manually clear memory by removing non-essential packages, datasets and files after every experiment.

7 Ethical Considerations

We do not believe there are serious ethical considerations specific to our project, which merely aimed to apply compression techniques to VLN models such that they could be run on an edge device. That said, there are certainly significant ethical implications with regard to edge deployment of VLN models in general. These implications mainly stem from the potential to use agents equipped with advanced VLN capabilities for unethical applications; for instance, VLN-capable robotic home butlers could be hacked and given language instructions commanding the robots to harm the home occupants. Robots and drones with VLN capabilities may also be used in military or warfare contexts, and would either directly or indirectly impact loss of human life in those cases.

On the other hand, on-edge deployment of VLN models can also contribute to moral and ethical causes. For example, VLN capabilities can be highly useful for search and rescue applications, in which robots may receive real-time information updates via language instructions from human workers to better guide them toward disaster victims. Robotic agents with VLN capabilities, as mentioned previously, may also be used as personal assistants or butlers. Such use cases have the potential to significantly improve the quality of life of certain users, such as users who are physically impaired and would benefit greatly from the ability to command robots to navigate to specific locations and perform certain tasks. In extreme cases, users may be severely injured but maintain the ability to speak; giving language instructions to the robot to tell it to retrieve a phone or medical kit, for example, could directly contribute toward saving the users' lives.

Ultimately, deployment of VLN models to edge devices is a tool that can be used for both ethical and unethical uses. It is up to the researchers, engineers, and other parties involved to carry out the required due diligence and safety measures to prevent VLN deployment from leading to undesirable or immoral outcomes.

References

Zhiyuan Fang, Jianfeng Wang, Xiaowei Hu, Lijuan Wang, Yezhou Yang, and Zicheng Liu. 2021. Compressing visual-linguistic model via knowledge distillation. *ArXiv*, abs/2104.02096.

Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. Compressing large-scale transformer-based models: A case study on bert.

Pierre-Louis Guhur, Makarand Tapaswi, Shizhe Chen, Ivan Laptev, and Cordelia Schmid. 2021. Airbert: In-domain Pretraining for Vision-and-Language Navigation.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tiny{bert}: Distilling {bert} for natural language understanding.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942.

Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. 2020. Improving vision-and-language navigation with image-text pairs from the web.

Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. 2020. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. Poor man's bert: Smaller and faster transformer models. *ArXiv*, abs/2004.03844.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020a. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020b. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10737–10746.

Yichi Zhang and Joyce Yue Chai. 2021. Hierarchical task learning from language instructions with unified transformers and self-monitoring. *ArXiv*, abs/2106.03427.