PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
Computer Science

ENHANCED VISION-LANGUAGE NAVIGATION BY USING SCENE RECOGNITION AUXILIARY TASK

RAIMUNDO MANTEROLA VALENZUELA

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Advisor:
ÁLVARO SOTO

Santiago de Chile, January 2021

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
Computer Science

ENHANCED VISION-LANGUAGE NAVIGATION BY USING SCENE RECOGNITION AUXILIARY TASK

RAIMUNDO MANTEROLA VALENZUELA

Members of the Committee:
ÁLVARO SOTO
WERHNER BREVIS
DENIS PARRA
IVAN LILLO

Thesis submitted to the Office of Research and Graduate Studies
in partial fulfillment of the requirements for the degree of
Master of Science in Engineering

Santiago de Chile, January 2021

To my parents, siblings, and friends

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Álvaro Soto for teaching me everything I know about Machine Learning, believing in me and my skills, and always pushing me forward and making me question assumptions on how things should be done.

I would also like to thank the whole IA Lab PUC group for always promoting a learning environment and being open to help whenever needed. I would especially like to thank Cristobal Eyzaguirre and Francisco Rencoret, who proof-read this document and gave me wise pieces of advice.

Special thanks to my family, who always supported my interests in Science and gave me the opportunity to study this Master.

Thanks to my friends who were always there to support me in whatever decision I made and gave me wise advice on my life path.

Finally, I would like to thank all the PUC Computer Science Department for all the teaching, motivation, openness, and friendliness.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Vision-Language Navigation is a highly demanding cognitive task that approached from a Machine Learning perspective, involves training an agent to navigate different scenarios following natural-language instructions. This task gets us one step closer to having smooth human-robot interactions. However, there is still a big gap between human performance and current Vision-Language Navigation models. Instructions usually describe paths making reference to places (i.e., turn right at the end of the kitchen), so understanding the semantics of different rooms is necessary to achieve correct navigation. Nevertheless, this understanding is usually not directly supervised and left to be learned implicitly. In this work, we propose an auxiliary task in which agents need to classify the different types of rooms they navigate and show that by adding this task, models learn how to navigate better and more efficiently, resulting in an increase in most Vision-Language Navigation metrics for seen and unseen scenarios during the training phase.

Keywords: Vision-Language Navigation, Deep Learning, Computer Vision, Natural Language Processing, Auxiliary Tasks.

RESUMEN

Vision-Language Navigation es una tarea cognitiva altamente exigente que abordada desde una perspectiva de *Machine Learning*, implica entrenar a un agente para navegar por diferentes escenarios siguiendo instrucciones en lenguaje natural. Esta tarea nos acerca un paso más a tener interacciones fluidas entre humanos y robots. Sin embargo, todavía existe una gran brecha entre el desempeño humano y los modelos actuales de *Vision-Language Navigation*. Las instrucciones suelen describir caminos que hacen referencia a lugares, por ejemplo, girar a la derecha al final de la cocina. Esto hace que sea necesario comprender la semántica de las diferentes habitaciones para lograr una correcta navegación. Sin embargo, esta comprensión por lo general no se supervisa directamente y se deja para ser aprendida de manera implícita. En este trabajo, proponemos una tarea auxiliar en la que los agentes deben clasificar los diferentes tipos de habitaciones por las que navegan, y demostramos empíricamente que al agregar esta tarea, los modelos aprenden a navegar mejor y de manera más eficiente. Esto se ve reflejado en un aumento en la mayoría de las métricas de *Vision-Language Navigation* tanto para escenarios vistos como no vistos durante la fase de entrenamiento.

Palabras Claves: Vision-Language Navigation, Aprendizaje Profundo, Visión por Computador, Procesamiento de Lenguaje Natural, Tareas Auxiliares.

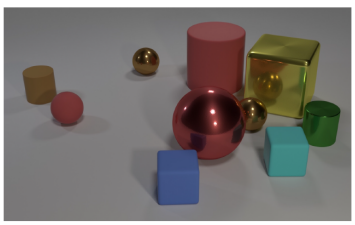
1. INTRODUCTION

1.1. Motivation

Machine Learning has gained considerable interest in the last few years, especially with the arrival of powerful Deep Learning models (Krizhevsky, Sutskever, & Hinton, 2012). Recent developments in areas such as Computer Vision and Natural Language Processing have enabled applications that were previously impossible, like real-time object tracking (Bochkovskiy, Wang, & Liao, 2020), facial recognition (Schroff, Kalenichenko, & Philbin, 2015), and language translation (Vaswani et al., 2017), amongst many others.

Computer Vision and Natural Language Processing are two research areas that have made substantial advances, where models sometimes have better-than-human abilities. However, there are still some applications where models lack the generalization capacity that humans possess, like understanding the hierarchy of concepts (Forbes, Holtzman, & Choi, 2019), or counting elements on images (Johnson et al., 2017). Some research lines suggest that in order to have accurate, complete language and vision models, it is not enough to treat them as separate areas, but there needs to be a vision and language grounding to achieve a better understanding of the world (Fjelland, 2020).

New applications involving both visual and language reasoning have appeared recently in order to tackle these challenges. Some examples of tasks are (1) Visual Question Answering (Antol et al., 2015), where models are trained to answer questions about an image (Figure 1.1.a.), (2) Image Captioning (X. Chen et al., 2015), where the goal is to produce text to describe what is happening on an image (Figure 1.1.b.), and (3) Vision-Language Navigation (Anderson et al., 2018), where an agent is taught how to navigate an environment following natural language instructions (Figure 1.1.c.).



Q: Are there an equal number of large things and metal spheres?
Q: What size is the cylinder that is left of the brown metal thing that is left of the big sphere? **Q:** There is a sphere with the same size as the metal cube; is it made of the same material as the small red sphere?
Q: How many objects are either small cylinders or metal things?

(a) Visual Question Answering

"two young girls are playing with lego toy."

(b) Image Captioning

Instruction: Head upstairs and walk past the piano through an archway directly in front. Turn right when the hallway ends at pictures and table. Wait by the moose antlers hanging on the wall.

(c) Vision-Language Navigation

Figure 1.1. Examples of Vision and Language tasks.

We will concentrate on the Vision-Language Navigation (VLN) task, where there is an agent that has to navigate following natural-language instructions. We believe that by making agents navigate and follow instructions better, we are getting one step closer to having a world in which there are robots that assist humans in daily tasks.

This is a field that has enormous potential for everyday applications. Being able to interact with robots with natural language instructions is a dream since The Jetsons aired in 1962. For robots, one super-important ability is to be able to navigate, and better if they can follow natural language instructions. Vision-Language Navigation is a research area that helps us get closer to that point and push science forward.

VLN setting consists of an indoor or outdoor scenario and a natural language instruction describing how to navigate this environment towards a specific goal. To perform well, models need to have language understanding skills, visual perception skills, and the ability to relate words with the visual environment in order to translate them into navigation actions. VLN is a new task in which the state of the art models still are under human performance (Zhu, Zhu, Chang, & Liang, 2020), leaving space for improvements and research to be done.

Following instructions in VLN requires the agent to understand some semantics of the environment and differentiate different kinds of scenarios. For example, to complete an instruction that indicates to enter the kitchen, agents must understand what a kitchen is. Previous works have left this knowledge to be left implicitly. In this work, we introduce a new auxiliary task that, besides making Vision-Language Navigation models predict navigation actions, they must classify in which kind of room they are standing at each timestep (i.e., kitchen, bedroom, hallway). This auxiliary task aims to regularize that models understand the semantic of different scenarios, expecting it makes them perform better on the main task.

1.2. Thesis Outline

Chapter 2 introduces the background information required to understand the method proposed in this work. This chapter is a brief overview of Deep Learning techniques, such as Convolutional Neural Networks for Computer Vision (Krizhevsky et al., 2012) and Recurrent Neural Networks for Natural Language Processing (Hochreiter & Schmidhuber, 1997).

Chapter 3 explains the Vision-Navigation task in detail and reviews the previous work done in the field, such as environment simulators, datasets, models, metrics, training, and evaluation methods. This chapter aims to provide all the knowledge needed to understand the task and its challenges.

Chapter 4 presents the method proposed in this work. It introduces the auxiliary task, all the preprocessing work to supervise it, the code's implementation, and the different experiments regarding this auxiliary task.

Chapter 5 shows the results obtained from the experiments made in different situations, such as previously seen or unseen environments, and presents some experiments to do a more qualitative analysis of the results. Later, there is a discussion section, analyzing the different advantages and implications of the results.

Chapter 6 is a Conclusion section that summarizes the work done, the results obtained, and its implications for the field, motivating further research in the area.

Finally, Chapter 7 introduces future research that could be done to extend this work. This chapter explores different directions in which variations of the same idea could be applied to get better performing Vision-Language Navigation models.

2. BACKGROUND INFORMATION

2.1. Deep Learning

Deep Learning is a branch of Machine Learning that consists of multiple layers of perceptrons (LeCun, Bengio, & Hinton, 2015). The appearance of the backpropagation algorithm (Rumelhart, Hinton, & Williams, 1985), and efficient model training in GPUs (Krizhevsky et al., 2012), have enabled this more than 60 years old technology (Rosenblatt, 1958) to become a powerful tool to solve recent Machine Learning problems like Image Classification, Natural Language Processing, and Vision-Language Navigation.

This technology's core parts are a network architecture, a training dataset, a loss function, and a training algorithm, usually some form of stochastic gradient descent (LeCun et al., 2015).

Besides Multilayer Perceptron, different structures have appeared to tackle more specific areas. In the next section, we will review two of the most popular architectures, which will be used in this work.

2.1.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are one of the most popular Deep Learning architectures for computer vision. They also have applications in other areas like NLP (Y. Zhang & Wallace, 2015) or Recommender Systems (S. Zhang, Yao, Sun, & Tay, 2019), but for this work, we are going to focus on the vision applications.

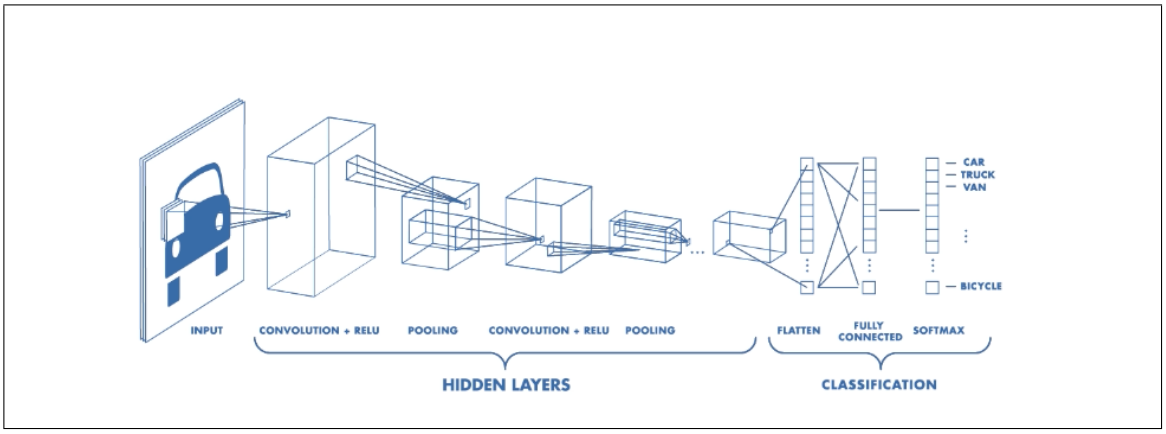


Figure 2.1. CNN architecture example.

The idea of this learning architecture is that it has multiple convolution layers, in which each layer detects the most common patterns, and then the next layer compose them to form new patterns (Krizhevsky et al., 2012). Consider a CNN trained to detect faces. The first layers would probably detect borders and lines; then, middle layers would detect parts such as noses, eyes, and lips; finally, the last layers would detect faces. This architecture has the advantage of being translation-invariant.

In this work we will use CNNs to process the visual field during navigation.

2.1.2. Recurrent Neural Networks

Recurrent Neural Networks are a class of neural networks that are particularly good for sequential forms of data, making them ideal for Natural Language Processing and other tasks involving time-series (Lipton, Berkowitz, & Elkan, 2015).

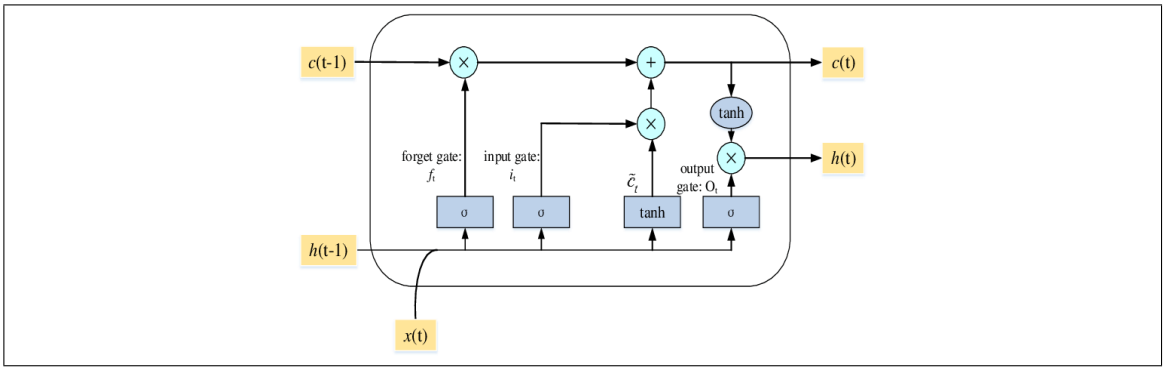


Figure 2.2. LSTM cell architecture.

LSTMs (Hochreiter & Schmidhuber, 1997) are among the most popular architectures of Recurrent Neural Networks and the one used in this work. This architecture's main idea is that it is composed of multiple sequential cells, where each of them is a function of an input vector representing part of the sequence, and a context vector coming from the previous cell.

For this work we will use RNNs for two things, understand natural language instructions and predict navigation actions at each timestep.

2.1.3. Auxiliary Tasks

Supervised Machine Learning models always have a main task for which the loss function is minimized. There are works that have shown that adding an auxiliary task has a regularizing effect on the main task (Ruder, 2017), reducing overfitting and improving results in the main task. An example of an auxiliary task would be adding a gender classification task to a model trained to recognize people's age based on face pictures.

There is evidence of auxiliary task being helpful both in navigation problems (Mirowski et al., 2016), and Vision-Language Navigation (Huang et al., 2019).

3. PREVIOUS WORK

VLN task consists of navigating through complex environments following natural language instructions. Scenarios can be both indoors (Anderson et al., 2018; Qi et al., 2019; Shridhar et al., 2020) and outdoors (H. Chen, Suhr, Misra, Snavely, & Artzi, 2019; de Vries et al., 2018), and instructions may be granular (Anderson et al., 2018), or high-level (Qi et al., 2019). Besides navigating, some datasets also require other tasks such as interacting with the environment (Shridhar et al., 2020) and objects localization (Qi et al., 2019).

Following the steps used by most state-of-the-art works, the setup to train a Vision-Language Navigation agent usually consists of an environment simulator, a training dataset, a decision taking agent (usually a Machine Learning model), and a decoding algorithm. Each of these building blocks is going to be reviewed in this chapter.

3.1. Simulators

Simulators used for VLN are usually photo-realistic 3D environments that aim to be a representation of the real world. These environments are often based on game engines (Beattie et al., 2016; Kempka, Wydmuch, Runc, Toczek, & Jaśkowski, 2016); however, newer simulators made with the purpose of this task have emerged. Here, we are going to review two of the most popular simulators.

3.1.1. Matterport3D Simulator

Matterport3D (Chang et al., 2017) is a photo-realistic indoor environment simulator that contains 90 realistic buildings. Each scenario represents a whole building and is discretized into viewpoints that represent a 360-panoramic image. There is a total of 10,800 panoramic views. Figure 3.1. shows an example of one of the houses simulated in Matterport3D.

Figure 3.1. Matterport building example (Chang et al., 2017).

3.1.2. AI2-THOR 2.0 Simulator

AI2 THOR (Kolve et al., 2017) is a photo-realistic indoor environment simulator that introduces physics to the simulator, making possible tasks such as object manipulation and environment interactions. It has 120 3D scenes, where each scene corresponds to one specific room, like a kitchen or a bedroom.

3.2. Datasets

This section reviews three of the most popular VLN datasets. The first one, R2R (Anderson et al., 2018), is the one we will use to evaluate our hypothesis. The other two are more complex sets that require other actions beside navigating, serving as motivation for further research.

3.2.1. Room-to-Room

Room-to-Room (R2R) (Anderson et al., 2018) is a Vision-Language Navigation dataset built on top of the Matterport3D simulator. It contains granular natural language instructions indicating each navigation step and an associated path to that instruction, for example; ”Walk straight passed bathtub and stop with closet on the left and toilet on the right.”.

There is a total of 21,567 instruction-path pairs. Figure 3.2. shows an example of a R2R instruction and the visual field received at a specific timestep.

Figure 3.2. Room-to-Room Dataset example (Anderson et al., 2018).

3.2.2. REVERIE

REVERIE (Qi et al., 2019) is a Vision-Language Navigation dataset built on top of the Matterport3D simulator. It has high-level instructions indicating only the goal, followed by an object-identification task, for example, ”Bring me the bottom picture that is next to the top of stairs on level one.”. It has a total of 21,702 instruction-path pairs.

3.2.3. ALFRED

ALFRED (Shridhar et al., 2020) is a Vision-Language Navigation dataset built on top of the AI2-THOR 2.0 simulator. It has granular instructions that involve interacting with the environment, such as ”Rinse off a mug and place it in the coffee maker.”. It contains 25,753 instruction-path pairs.

3.3. Models

In this section, we are going to review the baseline used for this work. We decided to use the Self-Monitoring agent model (Ma, Lu, et al., 2019) for this purpose since it is one of the top-performer works at the time. Our method is complimentary and could easily be implemented on top of other models as well. Recent works obtained better results (Zhu et al., 2020) at the moment of the realization of this research, but they had no available code implementation. This State-of-the-art agent uses four auxiliary tasks, none of them semantic-understanding related. We believe this model would also be benefited by adding scene grounding auxiliary tasks like room classification.

3.3.1. Self-Monitoring Agent

Self-Monitoring Agent (Ma, Lu, et al., 2019) is one of the top-performing models in the Room-to-Room dataset. Its architecture is described in figure 3.3. One important contribution of this work is that it introduces a Progress Estimation Auxiliary Task, being one of the first models that use an auxiliary task to improve performance in Vision-Language Navigation. It also introduces a visual-textual cogrounding method in which both visual and textual encoders feedback to each other. In Section 4.1.4, where we present the proposed method, Self-Monitoring Agent will be deeply described.

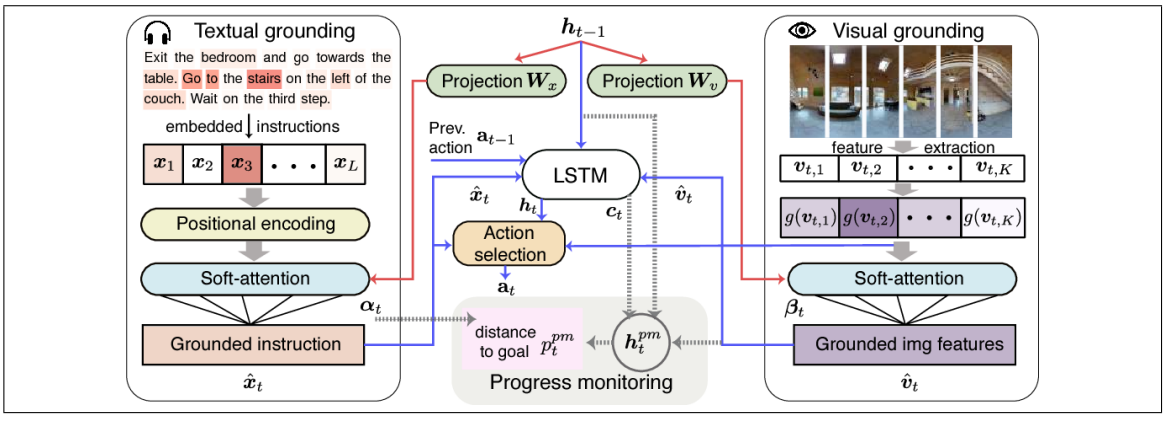


Figure 3.3. Self-Monitoring agent architecture (Ma, Lu, et al., 2019).

3.4. Metrics

The following metrics are used to measure the performance of this research.

Path Length represents the total distance in meters of the path predicted by the agent.

Navigation Error represents the distance in meters between the goal and the point the agent stopped.

Oracle Error is the same as Navigation Error, but with an oracle that stops once the agent is less than three meters from the goal.

Success Rate measures whether the agent completed a path successfully, meaning the stopping point is less than three meters from the goal.

Oracle Rate is the same as Success Rate, but with an oracle that stops once the agent is less than three meters from the goal.

Success weighted by Path Length (SPL) is the Success Rate weighted by the normalized Path Length, measuring whether the agent takes efficient paths to the goal.

3.5. Training Methods

Teacher Forcing: At each step during training, the ground-truth target action is selected to be conditioned on for the prediction of later outputs (Lamb et al., 2016).

Student Forcing: At each step, the next action is sampled from the agent's output probability distribution. Student-Forcing evaluates models the same way they were trained.

Behavioral Cloning + REINFORCE: First, there is a warming face of imitation learning using the ground-truth paths, followed by a reinforcement learning approach allowing more freedom on the paths taken (Bain & Sammut, 1995; Williams, 1992).

3.6. Decoding Methods

Greedy Decoding: Greedy Decoding consists of decoding the path during evaluation in the same way as while training, meaning each movement made by the agent is passed as the input for the next step.

Grid Search: Grid Search strategy is a decoding strategy that exhaustively searches all path possibilities and then chooses the one with a better chance of being correct. It may produce better results but has the disadvantage of being very slow compared to other strategies, making it less feasible for real applications.

Progress Inference: Progress Inference is a decoding strategy developed for the Self-Monitoring agent (Ma, Lu, et al., 2019) model based on the progress inference auxiliary task. In this strategy, if the progress inference output is lower than the previous step by more than X (defined threshold), then the agent does a step backward.

3.7. Data Augmentation

Speaker-Follower Agent is another model (Fried et al., 2018) for the VLN task, that introduces a pre-training task, in which a speaker agent is trained to create natural language instructions given a navigation path as input. This produces artificial data that can be used as a pre-training dataset to augment datasets like R2R (Anderson et al., 2018).

4. PROPOSED METHOD

4.1. Scene Recognition Auxiliary Task

Instructions usually involve references to the environment; for example, "follow the hallway and enter the bathroom." For these instructions to be understood and followed, there needs to be a comprehension of the scenario. Currently, this comprehension of the environment has not been directly supervised, delegating its learning as a consequence of teaching how to follow instructions.

We propose an auxiliary task that supervises this by making the agent predict which room category it is standing in at each timestep. We expect this semantic supervision helps the model build a more robust understanding of the world, translating into better navigation and better ability to follow instructions.

For this work, we will work with the Room-to-Room dataset (Anderson et al., 2018), based on the MatterPort3D Simulator (Chang et al., 2017).

4.1.1. Intuition

To learn how to navigate and follow natural language instructions, agents need to learn prior knowledge of environments. Instructions may require to *turn right and enter the kitchen*, or *follow the corridor and enter to the bathroom.* In order to complete these tasks, there needs to be a comprehension of what a kitchen, a corridor, or a bathroom means. This knowledge is usually left to be learned implicitly by the optimization algorithm by supervising only the instruction's completion.

We believe that by explicitly supervising that agents identify different classes of environments, this room-classification knowledge acquired will improve models' ability to follow navigation instructions.

(a) Kitchen

(b) Hallway

(c) Bedroom

Figure 4.1. Examples of auxiliary task predictions.

4.1.2. Getting Supervision

Since the R2R (Anderson et al., 2018) dataset does not have information about the different types of rooms, we need to get this supervision from somewhere else.

Fortunately, the Matterport3D simulator (Chang et al., 2017) contains meta-data that allows us to obtain this supervision. Each building is divided into regions, where each region has a label associated. From this, we can match data and get a label for each viewpoint. The next section is going to describe this process.

4.1.3. Preprocessing

In the MatterPort3D simulator, each building has a house file that divides the building into regions, where each region represents a different scenario. These regions have a category label associated amongst 31 possible categories (See appendix C.1). Each building is also represented as a graph, where each node represents a 360-panoramic viewpoint, and node connections represent two viewpoints being accessible from each other. Every viewpoint belongs to one of the regions described before.

To get the supervision we need, we have to get a viewpoint level label, so we can supervise that the VLN agent predicts the room category at each step. To do this, we did the mapping $viewpoint \longrightarrow region \longrightarrow category$, obtaining a label for each node.

After initial tests, we found out that the model had bad performance when asked to predict all 31 possible classes. Aiming to reduce the complexity of this problem, we did a manual search of the different categories and realized some similar classes could be grouped (Appendix C.2). For example, there is a different label for "bathroom" and "toilet," having the difference that one has a sink and the other does not. To reduce the number of classes and simplify the problem, we did a manual aggrupation of similar classes, leaving a total of 26 categories. Then, we filtered out classes that appeared in less than 300 viewpoints and grouped them into an "other class" category. Finally, we ended up with 11 possible categories for each viewpoint (Appendix C.2).

4.1.4. Model

As a baseline, we use the Self-Monitoring Agent (Ma, Lu, et al., 2019) mentioned in Chapter 2. This model can be divided into four components: 1) Visual Cogrounding, 2) Textual Cogrounding, 3) Action Selection, and 4) Progress Monitor.

(i) **Visual Cogrounding** module first process each viewpoint as 36 images, representing each possible view standing at that point. Each of these images is passed through a ResNet-152 (He, Zhang, Ren, & Sun, 2016) pre-trained in the Imagenet dataset (Deng et al., 2009), producing a 2048-dimension vector. Then a position encoding is concatenated, resulting in a 2076-d vector. We then pass each vector through a Multi-Layer Perceptron (MLP), and then a Soft-Attention (Bahdanau, Cho, & Bengio, 2014) is made, based on the previous state of the Action Selection module. The visual Soft-Attention weigth β_t can be obtained as:

$$z_{t,k}^{visual} = (W_v h_{t-1}^{\top}) g(v_{t,k}) \tag{4.1}$$

$$\beta_t = softmax(z_t^{visual}), \tag{4.2}$$

where g is a one-layer MLP, W_v are parameters to be learned, and h_{t-1} is the previous hidden state of the Action Selection LSTM. Finally, the grounded visual feature $\hat{v_t}$ can be obtained as the weighted sum over the visual features.

$$\hat{v_t} = \beta_t^{\top} V \tag{4.3}$$

(ii) **Textual Cogrounding**: This module takes as input the instruction and encodes it in a word-level using an LSTM (Hochreiter & Schmidhuber, 1997), then each encoded-word is concatenated with a Positional Encoding vector (Vaswani et al., 2017). Finally, a Soft-Attention (Bahdanau et al., 2014) operation is applied in the same way as in the Visual Grounding module. The textual attention weight α_t can be obtained as:

$$z_{t,l}^{textual} = (W_x h_{t-1}^{\top}) PE(x_l) \tag{4.4}$$

$$\alpha_t = softmax(z_t^{textual}), \tag{4.5}$$

where W_x are parameters to be learnt, x_l is the world l of the instruction, $PE()$ is the positional encoding operation, and h_{t-1} is the previous hidden state of the Action Selection LSTM. Finally, the grounded textual feature $\hat{x_t}$ can be obtained by the weighted sum over the textual features.

$$\hat{x_t} = \alpha_t^{\top} X \tag{4.6}$$

(iii) **Action Selection**: This module consists of an LSTM that selects the next action for each timestep. Actions can be whether to move to one of the reachable viewpoints or to stop at the current point. For each step, this module takes as input

the cogrounded image vector produced by the Visual Cogrounding module, the cogrounded text vector produced by the Textual Cogrounding module, and the action taken in the previous step. The probability p_t of each navigable direction at time t is then obtained as:

$$o_{t,k} = (W_a[h_{t-1}, \hat{x_t}])^\top g(v_{t,k}) \tag{4.7}$$

$$p_t = softmax(o_t), \tag{4.8}$$

where W_a are parameters to be learnt, g is a one-layer MLP, and h_t is the hidden state at time t of the LSTM.

(iv) **Progress Monitor**: This is an auxiliary task module, in which for each timestep, a value between 0 and 1 is predicted, indicating how close the agent thinks he is to the goal, conditioned to the output of the other three modules. To supervise this task, the normalized graph distance in meters from the current viewpoint to the goal node is used. The output of the progress monitor module p_t^{pm} can be computed as:

$$h_t^{pm} = \sigma(W_h([h_{t-1}, \hat{v_t}]) \bigotimes tanh(c_t)) \tag{4.9}$$

$$p_t^{pm} = tanh(W_{pm}([\alpha_t, h_t^{pm}])), \tag{4.10}$$

where W_h and W_{pm} are parameters to be learnt, c_t is the cell state of the Action Selection LSTM, $\bigotimes$ represents the element-wise product, and σ is the sigmoid funcion

The intuition is that; the Textual Cogrounding module identifies which parts of the instructions are relevant for the next action, the Visual Grounding module identifies which parts of the visual field are essential for the current action, the Action Selection module is responsible for deciding what the next action should

be, and the Progress Monitor module estimates how much of the trajectory has been completed.

4.1.5. Room Classsification module

In addition to the Self-Monitoring agent's four modules, we introduce the **Room Classification module**, where at each timestep, it predicts what class of room the agent is currently standing. To do this, the hidden state of the Action Selection LSTM is passed through a two-layer MLP, and the output is computed using a Softmax activation layer. The room classification output r_t can be obtained as:

$$r_t = softmax(f(h_t)) \tag{4.11}$$

Where f is a two-layer MLP, and h_t is the hidden state of the Action Selection LSTM. We decided what representation we use as the input of this module experimentally, and further experiments on how to calculate r_t will be explained in Section 4.3.

Finally, the loss of the model is a ponderation of the three losses of the three outputs; action selection, progress monitor, and room classification.

$$\mathcal{L}_{loss} = \lambda_1 \mathcal{L}_{action\ selection} + \lambda_2 \mathcal{L}_{progress\ monitor} + \lambda_3 \mathcal{L}_{room\ classification} \tag{4.12}$$

4.2. Implementation

To implement this model, we used the PyTorch (Paszke et al., 2017) library. We used the Self-Monitoring agent official repository as a baseline and added the Room Classification module and the required modifications for our model to work.

We used the same hyperparameters as the ones reported in the Self-Monitoring agent paper (Ma, Lu, et al., 2019). That means *batch size 64*, *Image Fully-Connected dim 1024*,

RNN hidden size 512, *learning rate 1 e-4*, and ADAM as optimizer. Further details can be checked in the original work.

For Action Selection and Room Classification losses, Cross Entropy loss was used, and for the Progress Monitor loss, we used Mean-Square error. As for the ponderators, $\lambda_1 = \lambda_2 = 0.5$, and $\lambda_3 = 0.2$. We arrived upon these values experimentally.

To keep things constant, we initialize both the baseline and the proposed model with the same weights and add the auxiliary room-classification module for the proposed model. We run experiments with five different seeds for weight initialization in order to detect variance in the results and avoid getting early conclusions.

All models were pre-trained for 300 epochs using augmented data created by the Speaker-Follower work (Fried et al., 2018).

4.3. Dataset details

To test our hypothesis, we used the Room-to-Room dataset (Anderson et al., 2018), which was previously described in Chapter 3. This dataset contains granular instructions describing how to navigate indoor environments. There are 90 different buildings containing mainly houses and offices. It has 7.189 paths extracted from its navigation graph, and for each path, there are three different ground-truth instructions annotated by human labelers, giving a total of 21.567 instruction-path pairs. Each path has an average distance of 10 meters, and each natural language instruction has an average of 26 words. The evaluation set is split into environments seen and unseen during the training stage.

4.4. Experiments

We did experiments to find out the best input to condition the Room Classification module and find out how to calculate Equation (4.11). The following combinations were tested:

- Aux_{image}: The room classification task was calculated based on the output of the Visual Cogrounding module.
- Aux_{ht}: The room classification task was calculated based on the last hidden state of the Action Selection LSTM.
- $Aux_{image+ht}$: The room classification task was calculated based on the concatenation of the Visual Cogrounding module and the last hidden state of the Action Selection LSTM.

We tested all experiments in the R2R dataset (Anderson et al., 2018) and evaluate them using greedy decoding on validation set.

Table 4.1. Results of experiment to determine the best architecture on Validation Seen set.

Model	Nav Error (m)	Oracle Error (m)	Steps	Lengths (m)	SPL	Success Rate (%)	Oracle Rate (%)
Baseline	5,90	**3,80**	8,86	16,52	0,29	0,41	**0,54**
Aux$_{Img}$	5,96	3,97	8,99	16,71	0,28	0,40	**0,54**
Aux$_{img+ht}$	**5,79**	3,83	8,64	16,04	**0,31**	**0,42**	0,53
Aux$_{ht}$	5,93	3,99	**8,58**	**15,90**	**0,31**	**0,42**	0,52

After training agents for the three combinations, we found that Aux_{ht} and Aux_{img+ht} provided the best results on the Success Rate and SPL metrics. We decided to choose Aux_{ht} as the final architecture, since it took shorter paths and had fewer parameters.

5. RESULTS AND DISCUSSION

5.1. Results

This section presents the results obtained during the evaluation of the three decoding methods introduced in Chapter 3.6. Each method is evaluated in two different subsets; seen environments, and unseen environments. The first one involving new instructions and paths in buildings already seen during training, and in the second one, both the instructions and the buildings are new.

The results reported in this section are the average of five experiments initialized with different random seeds to make sure the results are robust and not a product of a lucky initialization. The best results obtained can be checked in the appendix.

5.1.1. Greedy Decoding

5.1.1.1. Seen Environments

Table 5.1. Results of model v/s baseline performance in previously seen environments using Greedy decoding.

Model	Nav Error (m)	Oracle Error (m)	Steps	Lengths (m)	SPL	Success Rate (%)	Oracle Rate (%)
Baseline	3,28 ($\pm 0,05$)	1,96 ($\pm 0,09$)	7,25 ($\pm 0,12$)	12,75 ($\pm 0,22$)	0,59 ($\pm 0,01$)	0,67 ($\pm 0,01$)	0,76 ($\pm 0,01$)
Aux_{ht}	**3,18** ($\pm 0,07$)	**1,91** ($\pm 0,04$)	**7,10** ($\pm 0,15$)	**12,42** ($\pm 0,43$)	**0,62** ($\pm 0,02$)	**0,68** ($\pm 0,01$)	**0,78** ($\pm 0,01$)
Gain	*+0,10*	*+0,05*	*+0,15*	*+0,31*	*+0,03*	*+0,01*	*+0,02*

5.1.1.2. Unseen Environments

Table 5.2. Results of model v/s baseline performance in previously unseen environments using Greedy decoding.

Model	Nav Error (m)	Oracle Error (m)	Steps	Lengths (m)	SPL	Success Rate (%)	Oracle Rate (%)
Baseline	5,92 ($\pm 0,06$)	**3,82** ($\pm 0,16$)	8,80 ($\pm 0,24$)	16,35 ($\pm 0,58$)	0,30 ($\pm 0,02$)	0,42 ($\pm 0,01$)	**0,54** ($\pm 0,02$)
Aux_{ht}	**5,86** ($\pm 0,09$)	3,85 ($\pm 0,07$)	**8,72** ($\pm 0,19$)	**16,25** ($\pm 0,55$)	**0,31** ($\pm 0,01$)	**0,43** ($\pm 0,01$)	**0,54** ($\pm 0,01$)
Gain	*+0,06*	*-0,03*	*+0,08*	*+0,10*	*+0,01*	*+0,01*	*0,00*

Results using Greedy decoding are the most important for us since it is the most efficient method, it is comparable to other works, and it tests the agent in the same conditions it was trained. Results showed an improvement in both seen and unseen environments. However, gains were more significant in already known scenarios. We see a gain of 0.03 in SPL and 0.01 Success Rate for seen environments, showing an improvement in making more accurate, shorter paths. Having a higher gain in SPL, and having paths 0.31 meters shorter, make us think the auxiliary task helps the agent navigate more efficiently through environments he already knows. On the other side, in previously unseen environments, the gains are sustained and not remarkably higher in any metric, showing an improved overall performance, but not a specific bigger gain.

5.1.2. Beam Search

5.1.2.1. Seen Environments

Table 5.3. Results of model v/s baseline performance in previously seen environments using Beam Search decoding.

Model	Nav Error (m)	Oracle Error (m)	Steps	Lengths (m)	SPL	Success Rate (%)	Oracle Rate (%)
Baseline	3,09 $(\pm 0,17)$	1,89 $(\pm 0,08)$	6,09 $(\pm 0,10)$	**10,32** $(\pm 0,15)$	0,67 $(\pm 0,01)$	0,70 $(\pm 0,01)$	0,77 $(\pm 0,01)$
Aux_{ht}	**3,06** $(\pm 0,09)$	**1,87** $(\pm 0,03)$	**6,05** $(\pm 0,02)$	**10,36** $(\pm 0,07)$	**0,68** $(\pm 0,01)$	**0,71** $(\pm 0,02)$	**0,78** $(\pm 0,01)$
Gain	*+0,03*	*+0,02*	*+0,04*	*-0,04*	*+0,01*	*+0,01*	*+0,01*

5.1.2.2. Unseen Environments

Table 5.4. Results of model v/s baseline performance in previously unseen environments using Beam Search decoding.

Model	Nav Error (m)	Oracle Error (m)	Steps	Lengths (m)	SPL	Success Rate (%)	Oracle Rate (%)
Baseline	5,10 $(\pm 0,17)$	2,87 $(\pm 0,09)$	6,41 $(\pm 0,23)$	10,70 $(\pm 0,54)$	0,49 $(\pm 0,03)$	0,55 $(\pm 0,02)$	0,66 $(\pm 0,01)$
Aux_{ht}	**4,92** $(\pm 0,09)$	**2,82** $(\pm 0,10)$	**6,33** $(\pm 0,10)$	**10,52** $(\pm 0,20)$	**0,52** $(\pm 0,01)$	**0,57** $(\pm 0,01)$	**0,67** $(\pm 0,02)$
Gain	*0,18*	*0,05*	*0,08*	*0,18*	*0,03*	*0,02*	*0,01*

Beam Search is the decoding method that gives the best results, although we do not believe it is a suitable method. It is computationally inefficient to test all paths before deciding which one to choose, and infeasible for practical applications. We still show the results for the sake of being comparable with other works, which usually include this method. We can see improvements in both Success Rate and SPL for both seen and unseen environments, although in this case, the gains in unseen environments are more significant.

5.1.3. Progress Inference

5.1.3.1. Seen Environments

Table 5.5. Results of model v/s baseline performance in previously seen environments using Progress Inference decoding.

Model	Nav Error (m)	Oracle Error (m)	Steps	Lengths (m)	SPL	Success Rate (%)	Oracle Rate (%)
Baseline	3,38 ($\pm 0,14$)	2,00 ($\pm 0,11$)	7,46 ($\pm 0,21$)	12,89 ($\pm 0,44$)	0,60 ($\pm 0,00$)	0,68 ($\pm 0,02$)	0,76 ($\pm 0,02$)
Aux_{ht}	**3,24** ($\pm 0,16$)	**1,90** ($\pm 0,06$)	**7,19** ($\pm 0,22$)	**12,38** ($\pm 0,46$)	**0,62** ($\pm 0,01$)	**0,69** ($\pm 0,01$)	**0,78** ($\pm 0,01$)
Gain	*0,14*	*0,10*	*0,27*	*0,51*	*0,02*	*0,01*	*0,02*

5.1.3.2. Unseen Environments

Table 5.6. Results of model v/s baseline performance in previously unseen environments using Progress Inference decoding.

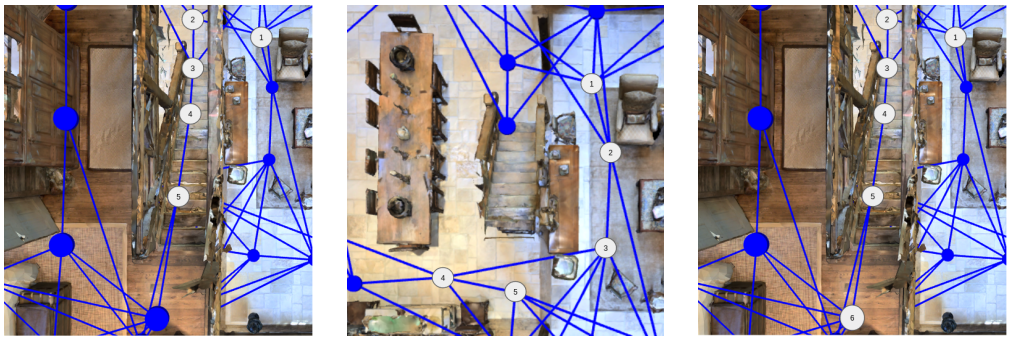
Model	Nav Error (m)	Oracle Error (m)	Steps	Lengths (m)	SPL	Success Rate (%)	Oracle Rate (%)
Baseline	5,79 ($\pm 0,04$)	**3,47** ($\pm 0,11$)	9,45 ($\pm 0,36$)	17,50 ($\pm 0,88$)	0,32 ($\pm 0,03$)	0,45 ($\pm 0,01$)	**0,58** ($\pm 0,01$)
Aux_{ht}	**5,75** ($\pm 0,12$)	3,53 ($\pm 0,09$)	**9,17** ($\pm 0,42$)	**17,06** ($\pm 1,07$)	**0,34** ($\pm 0,01$)	**0,46** ($\pm 0,01$)	0,57 ($\pm 0,01$)
Gain	*0,04*	*-0,06*	*0,28*	*0,44*	*0,02*	*0,01*	*-0,01*

Progress Inference is a good decoding method and one of the main contributions of the Self-Monitoring Agent paper (Ma, Lu, et al., 2019), but since it is bounded to one model, the results are less comparable. This method was also deprecated by the author in their next work (Ma, Wu, AlRegib, Xiong, & Kira, 2019), where they introduced a differentiable way of doing the same idea. Results here are consistent with the other decoding methods, producing a gain of 0.2 in SPL and 0.1 in Success Rate both in seen and unseen environments.

5.2. Experiments Analysis

The previous section showed quantitative results of the improvements made by adding a room-classification module. Here, we aim to do some qualitative research to determine the intuition of what is happening behind.

We started by adapting two tools extracted from other VLN repositories to visualize the results; one to get the bird-eye view of the building with the respective connectivity graph (Ke et al., 2019), and one for visualizing the first-person view of the paths followed (Anderson et al., 2018).



(a) Ground Truth Path (b) Baseline Path (c) Proposed Method Path

Figure 5.1. Examples of paths followed for instruction "Turn and go up the stairway. Stop and wait at the second step from the top."

After manually reviewing examples, we concluded that instructions that made reference to places were easier to complete for the proposed model. For example, we can see in Figure 5.1. the baseline method could not complete an instruction that required to go up through a stairway, while the proposed method that had an auxiliary task that made him identify scenarios like stairways was able to complete right. On the opposite side, instructions that did not reference places and only made reference to objects (i.e., turn right after the white couch and continue walking until you reach a piano) were harder to complete correctly.

To check if this hypothesis was right, we separated the evaluation data into two categories; instructions referring to places and instructions that did not refer to any room category. After that, we compared the baseline's performance and the proposed method in each of these splits.

To separate the instructions, we used the FLAIR python NLP library (Akbik, Blythe, & Vollgraf, 2018). We used a pre-trained model to PoS-tag words and extracted all nouns in instructions. In the context of VLN instructions, nouns usually make reference either to objects or places, so we manually checked the most common nouns that made reference to places and created a list we used to filter the dataset.

For seen environments, once filtered, we ended up with 767 path-instruction pairs that referenced to places and 253 that did not. For the unseen environments split, we ended with 1817 instruction that referenced places, and 532 did not.

Then we grabbed the best weights and compared the Success Rate metric in both splits using Greedy decoding to find out where the gains were biggest.

Table 5.7. Success Rate of baseline versus proposed method in instructions that made and made not reference to places. Results produced using the best weights.

	Val Seen SR		Val Unseen SR	
Split	Places	Not Places	Places	Not Places
Baseline	0,67	0,59	0,44	0,33
Aux_{ht}	**0,73**	**0,62**	**0,47**	**0,34**
Gain	*0,06*	0,03	0,03	*0,01*

5.3. Discussion

After analyzing the results, we can see improvements in almost every single metric for every combination of decoding algorithms and seen/unseen environments. The two

most important metrics we use for our analysis are Success Rate, meaning how many instructions are completed, and SPL, meaning how many instructions are completed with an efficient path.

A bigger gain in SPL than Success Rate for every decoding algorithm shows that the most significant impact of the proposed method is in making models take shorter, more efficient paths. Gains in Success Rate also shows that the agent with the scene-recognition auxiliary task was able to complete instructions that the baseline agent was not. This made us do the experiments described in Section 5.2 to find out which were the examples that produced gains.

We split the evaluation set in instructions that referred to places, and instructions that did not, measuring the success rate for both splits. Table 5.7 shows that the gain was more significant for instructions that made reference to places, two times bigger for seen environments and three times bigger for unseen ones. Another interesting fact is that for both agents the success rate is much higher in instructions that reference places. This shows that the hardest instructions for this architecture are those that do not make reference to places and mainly use objects to give instructions, showing a big room for improvements for future works aiming to tackle these issues.

Although we implemented the idea of a room-classification auxiliary task for the Self-Monitoring Agent (Ma, Lu, et al., 2019) model, we expect other VLN models would also benefit from supervising room-classification at each step. We believe this is a necessary ability for a correct navigation, and by explicitly supervising it, models should learn better.

6. CONCLUSIONS

Vision-Language Navigation works aim to teach agents how to navigate environments following natural language instructions by giving a ground truth path and the instruction associated, hoping for the optimization algorithms to learn all the abilities needed to complete this task. One of these abilities required to be learned is to identify different kinds of scenarios. This work showed that explicitly teaching how to classify room categories helps the agent learn to navigate better, translating into better performance in every metric measured.

Our principal contribution is we demonstrated that by adding an auxiliary task that classifies in which class of environment the agent is standing at each timestep, agents achieve better and more efficient navigation. This translated in an improvement in Success Rate and Success Rate weighted by Path Length metrics in both previously seen and unseen environments.

Even though we show progress in the field, agents are still under human performance, and there is much more work to do before seeing this technology applied in industrial applications. We hope this work contributes a grain of sand towards the goal of having machine assistants able to follow natural language instructions.

This work also exhibited flaws in agents' ability to follow instructions that only reference objects and not places, showing a great space for improvement. New auxiliary tasks that force agents to acquire the different building blocks necessary for navigation, like room-classification or object recognition tasks, should benefit agents and improve navigation.

We hope this work also contributed to having better language models by making a grounding in simulators of the real world. We believe learning how the world operates from text only is not sufficient and should be accompanied by having visual grounding and the ability to perform actions to achieve its full potential.

7. FUTURE WORK

Vision-Language Navigation is a research domain that still is far from solved. Having Success Rates of around 40% for unseen environments in R2R indicates that this technology is still far from achieving industry-level applications. Much research is still to be done, and in this section, we present different scenarios in which we believe we could apply our idea of semantic-understanding regularization.

7.1. Other Grounding Auxiliary Tasks

One area this research could continue is to propose other auxiliary tasks that demand a semantic understanding of the environment. This work showed that by adding supervision of scene comprehension to the agent, the navigation performance is positively impacted. Besides room classification, other auxiliary tasks could serve as an environment understanding regularizers.

7.1.1. Object Detection

Recent research has shown that object segmentation and classification task improves navigation (Chaplot, Gandhi, Gupta, & Salakhutdinov, 2020) on the new Habitat ObjectNav Challenge (Kadian et al., 2019), also based on the MatterPort3D simulator. This task is probably helpful for Vision-Navigation datasets and could act as a complement to Room-Classification for semantic scene understanding. This could help tackle the issues shown in table 5.7, which reveal that agents had lower performance in instructions that did not reference scenes but only referenced objects.

7.2. Train Regretful Agent

The authors of the Self-Monitoring agent (Ma, Lu, et al., 2019) we used as the architecture for our model, published an extension of their work, where they introduced

Regretful Agent (Ma, Wu, et al., 2019), a new way of training Self-Monitoring agent that grabbed the Progress Inference decoding idea and applied it in a differentiable way during training. This showed better results in both seen and unseen environments. We believe our proposed method could also be benefited by training the agent in this same way.

7.3. More Datasets

Another line this research could be expanded is to test if adding a Room Classification auxiliary task helps achieve better navigation in more complex datasets, like REVERIE (Qi et al., 2019) or ALFRED (Shridhar et al., 2020), where other abilities are required besides navigation, like interactions with the environment and object localization.

REFERENCES

Akbik, A., Blythe, D., & Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics* (pp. 1638–1649).

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., ... Van Den Hengel, A. (2018). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3674–3683).

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence Zitnick, C., & Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2425–2433).

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bain, M., & Sammut, C. (1995). A framework for behavioural cloning. In *Machine Intelligence 15* (pp. 103–129).

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., ... Sadik, A. (2016). Deepmind lab. *arXiv preprint arXiv:1612.03801*.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., ... Zhang, Y. (2017). Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*.

Chaplot, D. S., Gandhi, D., Gupta, A., & Salakhutdinov, R. (2020). *Object goal navigation using goal-oriented semantic exploration.*

Chen, H., Suhr, A., Misra, D., Snavely, N., & Artzi, Y. (2019). Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 12538–12547).

Chen, X., Fang, H., Lin, T., Vedantam, R., Gupta, S., Dollár, P., & Zitnick, C. L. (2015). Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255).

de Vries, H., Shuster, K., Batra, D., Parikh, D., Weston, J., & Kiela, D. (2018). Talk the walk: Navigating new york city through grounded dialogue. *arXiv preprint arXiv:1807.03367*.

Fjelland, R. (2020). Why general artificial intelligence will not be realized. *Humanities and Social Sciences Communications*, *7*(1), 1–9.

Forbes, M., Holtzman, A., & Choi, Y. (2019). Do neural language representations learn physical commonsense? *arXiv preprint arXiv:1908.02899*.

Fried, D., Hu, R., Cirik, V., Rohrbach, A., Andreas, J., Morency, L.-P., … Darrell, T. (2018). Speaker-follower models for vision-and-language navigation. In *Advances in Neural Information Processing Systems* (pp. 3314–3325).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Huang, H., Jain, V., Mehta, H., Ku, A., Magalhaes, G., Baldridge, J., & Ie, E. (2019). *Transferable Representation Learning in Vision-and-Language Navigation.*

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., & Girshick, R. (2017). CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2901–2910).

Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., … Batra, D. (2019). Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation. In *arXiv:1912.06321*.

Ke, L., Li, X., Bisk, Y., Holtzman, A., Gan, Z., Liu, J., … Srinivasa, S. (2019). Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6741–6749).

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., & Jaśkowski, W. (2016). Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)* (pp. 1–8).

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., … Farhadi, A. (2017). Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lamb, A. M., Goyal, A. G. A. P., Zhang, Y., Zhang, S., Courville, A. C., & Bengio, Y.

(2016). Professor forcing: A new algorithm for training recurrent networks. In *Advances in neural information processing systems* (pp. 4601–4609).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Ma, C.-Y., Lu, J., Wu, Z., AlRegib, G., Kira, Z., Socher, R., & Xiong, C. (2019). Self-monitoring navigation agent via auxiliary progress estimation. *arXiv preprint arXiv:1901.03035*.

Ma, C.-Y., Wu, Z., AlRegib, G., Xiong, C., & Kira, Z. (2019). The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 6732–6740).

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A. J., Banino, A., ... others (2016). Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... Lerer, A. (2017). Automatic differentiation in pytorch.

Qi, Y., Wu, Q., Anderson, P., Liu, M., Shen, C., & Van Den Hengel, A. (2019). Rerere: Remote embodied referring expressions in real indoor environments. *arXiv preprint arXiv:1904.10151*, *2*.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation* (Tech. Rep.). California Univ San Diego La Jolla Inst for Cognitive Science.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., ... Fox, D. (2020). Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10740–10749).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*(3-4), 229–256.

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, *52*(1), 1–38.

Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Zhu, F., Zhu, Y., Chang, X., & Liang, X. (2020). Vision-language navigation with self-supervised auxiliary reasoning tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10012–10022).

APPENDIX

A. BEST RESULTS

Here we present the results of the best weigth initialization, using 3542 as random seed.

A.1. Greedy Decoding

Table A.1. Best results of model v/s baseline performance in previously seen environments using Greedy Decoding.

Model	Nav Error	Oracle Error	Steps	Lengths	SPL	Success Rate	Oracle Rate
Baseline	3,32	2,09	7,12	12,44	0,59	0,65	0,74
Aux$_{ht}$	**3,11**	**1,89**	**6,96**	**12,11**	**0,64**	**0,70**	**0,78**

Table A.2. Best results of model v/s baseline performance in previously unseen environments using Greedy Decoding.

Model	Nav Error	Oracle Error	Steps	Lengths	SPL	Success Rate	Oracle Rate
Baseline	5,90	3,99	**8,45**	**15,53**	**0,32**	0,42	0,51
Aux$_{ht}$	**5,78**	**3,83**	8,53	15,68	**0,32**	**0,44**	**0,54**

A.2. Grid Search

Table A.3. Best results of model v/s baseline performance in previously seen environments using Beam-Search decoding.

Model	Nav Error	Oracle Error	Steps	Lengths	SPL	Success Rate	Oracle Rate
Baseline	3,12	1,97	**5,97**	**10,20**	0,67	0,69	0,76
Aux$_{ht}$	**2,98**	**1,84**	6,04	10,35	**0,69**	**0,73**	**0,78**

Table A.4. Best results of model v/s baseline performance in previously unseen environments using Beam-Search decoding.

Model	Nav Error	Oracle Error	Steps	Lengths	SPL	Success Rate	Oracle Rate
Baseline	5,02	3,01	**6,11**	**9,98**	**0,52**	**0,57**	**0,66**
Aux$_{ht}$	**4,94**	**2,95**	6,18	10,23	0,51	0,56	0,65

A.3. Progress Inference

Table A.5. Best results of model v/s baseline performance in previously seen environments using Progress-Inference decoding.

Model	Nav Error	Oracle Error	Steps	Lengths	SPL	Success Rate	Oracle Rate
Baseline	3,50	2,15	7,19	12,30	0,59	0,66	0,74
Aux$_{ht}$	**3,16**	**1,90**	**7,02**	**12,05**	**0,64**	**0,70**	**0,78**

Table A.6. Best results of model v/s baseline performance in previously unseen environments using Progress-Inference decoding.

Model	Nav Error	Oracle Error	Steps	Lengths	SPL	Success Rate	Oracle Rate
Baseline	5,76	3,62	8,92	16,23	**0,36**	**0,46**	0,56
Aux$_{ht}$	**5,75**	**3,57**	**8,78**	**16,11**	0,34	**0,46**	**0,57**

B. NAVIGATION EXAMPLES

In this section, we show some examples of the trajectories the different agents followed in the Room-2-Room dataset. We show the ground truth path, followed by the baseline and proposed method trajectories.

B.1. Positive Example

This is an instruction that the proposed method followed right, and the baseline wasn't able to complete.

Instruction: Exit the bathroom, head downstairs and stop in the middle of the first flight of stairs.

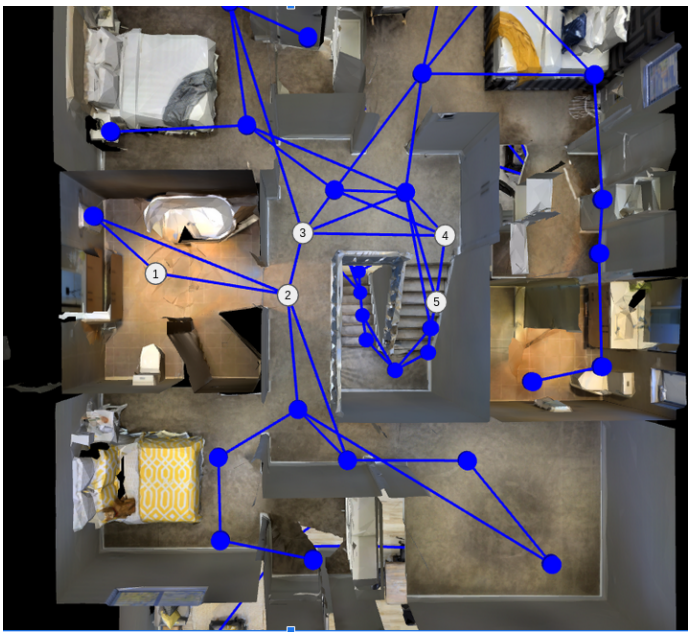


Figure B.1. Ground Truth Path.

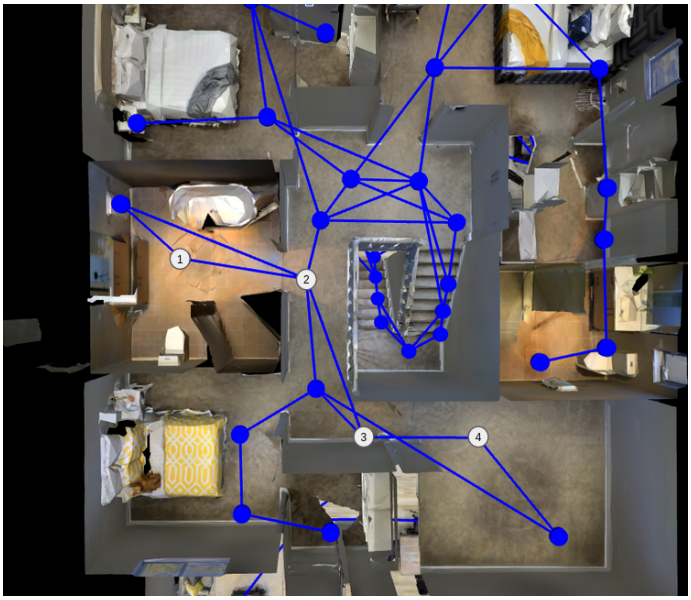


Figure B.2. Baseline Path.

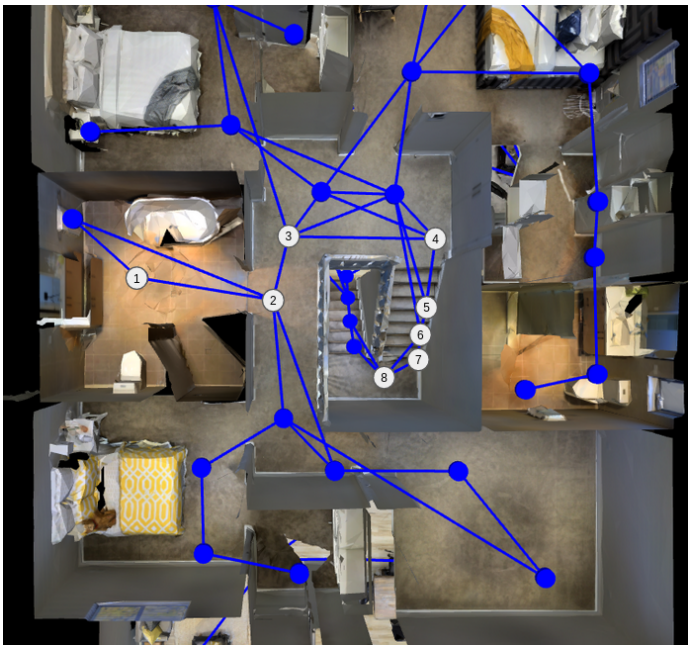


Figure B.3. Proposed Method Path.

B.2. Negative Example

This is an instruction that the model with the Room-Classification auxiliary task was not able to complete, while the baseline model reached the objective right.

Even though the model without the auxiliary task was able to reach the objective, it followed a path far from optimal.

Instruction: Walk straight and then turn left and go between the couch and kitchen counter. Enter the next room and stop at the corner of the first white couch and wait.

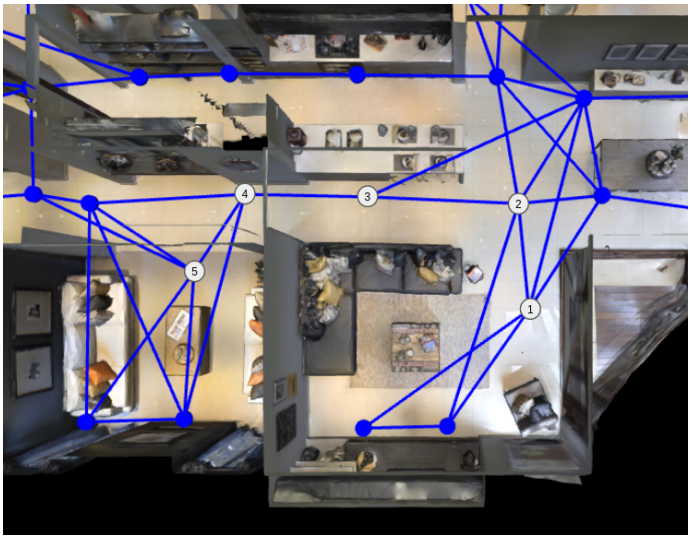


Figure B.4. Ground Truth Path.

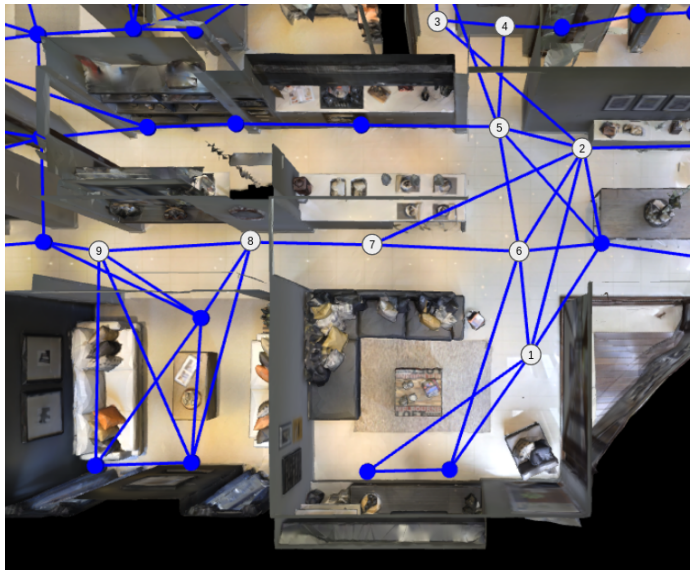


Figure B.5. Baseline Path.

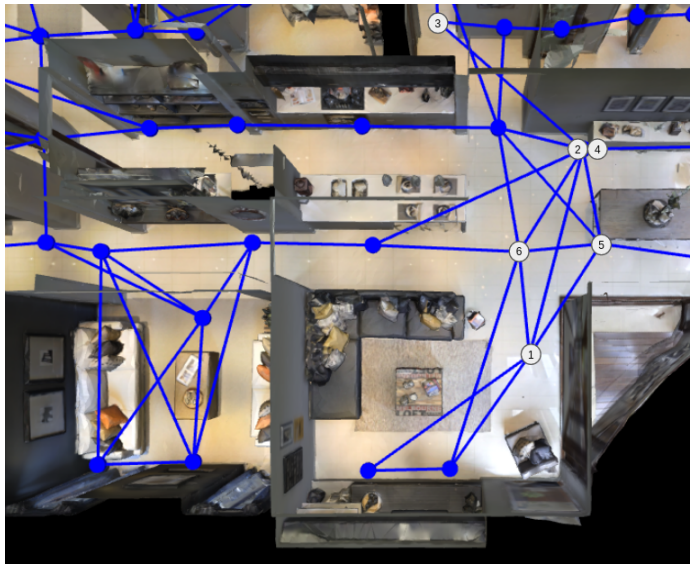


Figure B.6. Proposed Method Path.

C. ROOM CATEGORIES

C.1. Original List

- Bathroom
- Bedroom
- Closet
- Dining Room
- Lobby
- Family Room
- Garage
- Hallway
- Library
- Laundry Room
- Kitchen
- Living Room
- Conference Room
- Lounge
- Office
- Terrace
- Game Room
- Stairs
- Toilet
- Utility Room
- TV Room
- Gym
- Outdoor
- Balcony
- Other Room
- Bar

- Classroom
- Dining Booth
- Spa
- Junk

C.2. Mappings

$\{Bathroom,\ Toilet\} \longrightarrow Bathroom$

$\{Terrace,\ Outdoors,\ Balcony\} \longrightarrow Outdoors$

$\{Family\ Room,\ Living\ Room,\ Lounge\} \longrightarrow Living\ Room$

$\{Conference\ Room,\ Closet,\ Spa,\ Game\ Room,\ TV\ Room,\ Garage,\ Laundryroom,$
$Library,\ Gym,\ Utility\ Room,\ Classroom,\ Bar,\ Junk\} \longrightarrow Other\ Room$

C.3. Final Categories

- Bathroom
- Bedroom
- Dining Room
- Hallway
- Kitchen
- Living Room
- Lobby
- Office
- Outdoors
- Stairs
- Other Room

HELLOSIGN

Audit Trail

TITLE	Firma final documento tesis Raimundo Manterola
FILE NAME	Enhanced Vision-L...uxiliary Task.pdf
DOCUMENT ID	253894410b5894151c36f0dde5de886d143981c6
AUDIT TRAIL DATE FORMAT	MM / DD / YYYY
STATUS	● Completed

Document History

SENT	**01** / **11** / **2021** 15:43:40 UTC	Sent for signature to Álvaro Soto (alvarosotoa@gmail.com), Denis Parra (dparra@ing.puc.cl), Werhner Brevis (wbrevis@ing.puc.cl) and Ivan Lillo (ialillo@gmail.com) from r.manterola1@gmail.com IP: 190.96.40.106
VIEWED	**01** / **11** / **2021** 15:53:04 UTC	Viewed by Werhner Brevis (wbrevis@ing.puc.cl) IP: 181.161.243.84
SIGNED	**01** / **11** / **2021** 15:53:41 UTC	Signed by Werhner Brevis (wbrevis@ing.puc.cl) IP: 181.161.243.84
VIEWED	**01** / **11** / **2021** 16:02:15 UTC	Viewed by Álvaro Soto (alvarosotoa@gmail.com) IP: 181.43.78.76
SIGNED	**01** / **11** / **2021** 16:03:38 UTC	Signed by Álvaro Soto (alvarosotoa@gmail.com) IP: 181.43.78.76

HELLOSIGN

Audit Trail

TITLE	Firma final documento tesis Raimundo Manterola
FILE NAME	Enhanced Vision-L...uxiliary Task.pdf
DOCUMENT ID	253894410b5894151c36f0dde5de886d143981c6
AUDIT TRAIL DATE FORMAT	MM / DD / YYYY
STATUS	Completed

Document History

VIEWED	**01** / **11** / **2021** 17:45:52 UTC	Viewed by Denis Parra (dparra@ing.puc.cl) IP: 201.188.82.174
SIGNED	**01** / **11** / **2021** 17:47:17 UTC	Signed by Denis Parra (dparra@ing.puc.cl) IP: 201.188.82.174
VIEWED	**01** / **12** / **2021** 16:27:32 UTC	Viewed by Ivan Lillo (ialillo@gmail.com) IP: 181.42.23.51
SIGNED	**01** / **12** / **2021** 16:28:03 UTC	Signed by Ivan Lillo (ialillo@gmail.com) IP: 181.42.23.51
COMPLETED	**01** / **12** / **2021** 16:28:03 UTC	The document has been completed.