Conditional Driving from Natural Language Instructions

Junha Roh[†], **Chris Paxton**[‡], **Andrzej Pronobis**[†*], **Ali Farhadi**[†§], **Dieter Fox**[†‡]
University of Washington[†], NVIDIA[‡]
KTH Royal Institute of Technology[*], Allen Institute for AI[§]
`{rohjunha,pronobis,ali}@cs.washington.edu`
`{cpaxton,dieterf}@nvidia.com`

Abstract: Widespread adoption of self-driving cars will depend not only on their safety but largely on their ability to interact with human users. Just like human drivers, self-driving cars will be expected to understand and safely follow natural-language directions that suddenly alter the pre-planned route according to user's preference or in presence of ambiguities, particularly in locations with poor or outdated map coverage. To this end, we propose a language-grounded driving agent implementing a hierarchical policy using recurrent layers and gated attention. The hierarchical approach enables us to reason both in terms of high-level language instructions describing long time horizons and low-level, complex, continuous state/action spaces required for real-time control of a self-driving car. We train our policy with conditional imitation learning from realistic language data collected from human drivers and navigators. Through quantitative and interactive experiments within the CARLA framework, we show that our model can successfully interpret language instructions and follow them safely, even when generalizing to previously unseen environments. Code and video are available at: https://sites.google.com/view/language-grounded-driving.

Keywords: Language to control, autonomous vehicles, imitation learning

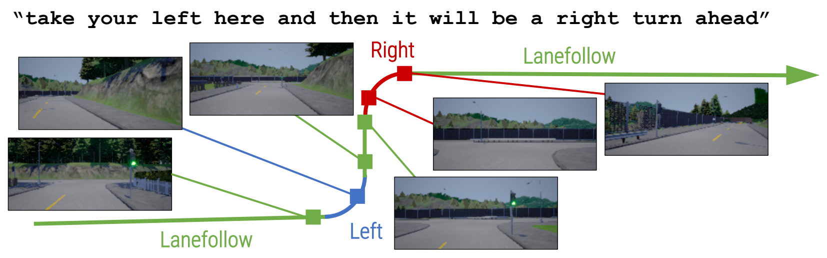
"take your left here and then it will be a right turn ahead"
Right
Lanefollow
Left
Lanefollow

Figure 1: Natural language control of self-driving vehicles. The user provides a high-level instruction; the vehicle must then (a) translate natural language into the correct sequence of high-level sub-tasks and (b) correctly execute these, by steering and applying the throttle as appropriate.

1 Introduction

Passengers of self-driving cars will expect to interact with their vehicles in the same way as they do with human ride-share drivers. This includes providing specific directions to the precise drop-off locations, preferences about the chosen route, or clarifications in case of ambiguities. Furthermore, a car equipped with a skill to interpret natural-language, able to rely on the help of its user, will be more robust to navigation errors resulting from poor map coverage and inaccurate information about dynamic road conditions.

As shown in Fig. 1, our goal is to learn to understand language instructions, such as "you are going to go a little bit further for one block and make a left at the intersection," and use them to condition a policy that will drive a car safely using only image observations. The problem of end-to-end

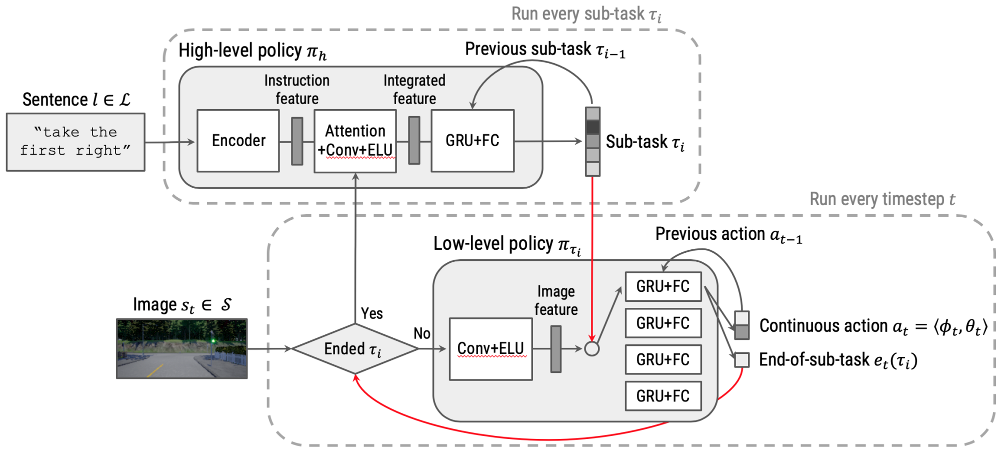
Run every sub-task τ_i
High-level policy π_h
Previous sub-task τ_{i-1}
Instruction feature
Integrated feature
Sentence $l \in \mathcal{L}$
"take the first right"
Encoder
Attention +Conv+ELU
GRU+FC
Sub-task τ_i
Run every timestep t
Low-level policy π_{τ_i}
Previous action a_{t-1}
Image $s_t \in \mathcal{S}$
Yes
No
Ended τ_i
Image feature
Conv+ELU
GRU+FC
GRU+FC
GRU+FC
GRU+FC
Continuous action $a_t = \langle \phi_t, \theta_t \rangle$
End-of-sub-task $e_t(\tau_i)$

Figure 2: The proposed model for language-grounded driving. The model takes an image from the dashboard-mounted camera and a natural language instruction and generates steering and throttle values for control. Gray and red arrows represent flows of tensors and control switching signals, respectively.

policy learning for self-driving cars is often formulated as imitation learning [1, 2, 3, 4]. We take a hierarchical approach. We use conditional imitation learning to learn policies for steering and throttle control, as in prior work [3, 4]. However, we also learn a high-level policy, which predicts high-level actions based on language instructions. This leads to a solution able to translate language into actions executed over long time horizons, including navigating multiple streets and turns before reaching the destination.

At the same time, we need to ensure that safety is not compromised, even in presence of incorrect and misleading instructions. A self-driving car will be used by non-experts, who might instruct the car to execute maneuvers that are not safe given the state of the world (e.g. to turn left when no left turn is possible). This is a known problem in language-to-control [5]. Moreover, artificial systems often struggle with understanding the intricacies of realistic human language, in particular for such a dynamic task as driving. We provide two pathways to mitigate the harm this can cause: first, our policy is designed to ensure that only safe actions are taken, even when invalid input is given by the passenger. Second, the agent will complete maneuvers, such as driving through an intersection, even if new instructions from the user interrupt the current high-level plan.

We validate our proposed approach using CARLA [6], an open-source driving simulator. We perform a set of quantitative transfer experiments, showing that our hierarchical models navigate correctly and safely, and can generalize between different environments. Furthermore, we perform a series of ablation tests to study the properties of our model. Finally, we design an interactive experiment, with users instructing the car in real-time with randomly timed and misleading instructions.

To summarize, our core contributions are: (1) an end-to-end policy controlling a self-driving car from language and images; (2) a hierarchical architecture reasoning about both short and long time horizons as well as both high-level inputs and low-level continuous states and actions; (3) an implementation of interactive language-grounded driving robust to misleading user instructions.

2 Related work

Our approach is closely related to work on Visual Question Answering (VQA), a growing area of research in which an agent is trained to move about in a home environment and find the answers to specific questions [7, 8, 9, 10, 11, 12]. In particular Das et al. [7] proposed Neural Modular Control (NMC), which used a multi-level model to predict a "program" of actions that need to be taken. While closely related, our method uses continuous state and action spaces with a realistic car model, while the vision-language navigation problem is mainly focused on dealing with complicated language expressions with relatively limited discrete state and action spaces.

Other recent work has explored learning driving policies from images. Liang et al. [13] use a mixture of imitation learning and reinforcement learning via DDPG [14]. Codevilla et al. [3] proposed a system for conditional end-to-end driving. Müller et al. [4] extends this work, adding a segmentation model which improves generalization performance, but largely keeping the same structure from Codevilla et al. [3]. Paxton et al. [15] also learn hierarchical policies for driving through intersections, but focus on interacting with other vehicles and do not use images.

Some work has also looked at learning representations based on images and language, but not for driving. Chaplot et al. [16] proposed a Gated Attention model for learning navigation policies based on images and language, an approach we borrow for our high-level model. Paxton et al. [5] learned to generate task plans and execute pick-and-place tasks. Blukis et al. [17] learn a semantic map for navigation and demonstrate on a simulated quadrotor, which can be used to follow natural language instructions.

While we provide a manual decomposition of the task when training models as seen in some previous work [5, 7], some prior work weakens these assumptions. Shiarlis et al. [18] propose TACO, which learns to break tasks up based only on a policy sketch. Krishnan et al. [19] also propose a method for discovery of continuous actions from demonstrations, which could be applied to our problem in the future. Andreas et al. [20] uses policy sketches together with curriculum reinforcement learning.

3 Hierarchical Language-Grounded Driving Model

The goal of our agent is to drive safely, following given language directions and a stream of images from a single camera. Driving requires a very long time horizon, with high-frequency controls but low-frequency decisions. This makes it difficult to directly apply a sequence-to-sequence approach to the problem. Instead, we introduce a hierarchical driving model where a high-level policy π_h chooses a series of sub-tasks $\{\tau_0, \dots, \tau_{N-1}\}$ to achieve a specified task, and low-level policies π_{τ_i} generate the controls necessary to achieve each sub-task τ_i in sequence. This decomposes the problem into tractable sub-problems and enables efficient use of short data sequences for training complex, language-conditioned control policies. This approach is similar to that taken in the vision-language navigation problem [7], but with increased complexity of low-level controls.

Algorithm 3.2 shows the pseudo code for execution of our language-grounded driving model. Figure 2 shows the architecture of our model. Consider the world $W : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ with a continuous state observation $s \in \mathcal{S}$ and a continuous action $a = \langle \phi, \theta \rangle \in \mathcal{A}$, where $\phi \in [0, 1]$ is the normalized throttle control and $\theta \in [-1, 1]$ is the normalized steering angle for the vehicle. We assume that our state observation s_t consists of an image from a dashboard-mounted camera at time t. Then the directions for the language-grounded driving are specified by a natural-language input $l \in \mathcal{L}$, such as "take the next right" or "go straight through this intersection, then turn left." Finally, our problem is defined by learning a policy $\pi : \mathcal{S} \times \mathcal{L} \rightarrow \mathcal{A}$.

We break up the original problem into a two-level hierarchy by introducing a sub-task $\tau \in \mathcal{T}$ where the set of possible sub-tasks $\mathcal{T} = \{\texttt{left}, \texttt{right}, \texttt{straight}, \texttt{lanefollow}\}$. The sub-task `straight` represents the case of going straight through an intersection while `lanefollow` corresponds to a policy ensuring safe lane following outside intersections. We extend $\mathcal{T}$ to include a `finish` token, indicating the end of the entire task: $\hat{\mathcal{T}} = \mathcal{T} \cup \{\texttt{finish}\}$.

With the hierarchical model, our problem is to learn a high level policy $\pi_h : \mathcal{S} \times \mathcal{L} \rightarrow \hat{\mathcal{T}}$, and a corresponding low-level policy $\pi_\tau : \mathcal{S} \times \mathcal{T} \rightarrow \mathcal{A}$. Also, our model detects when the sub-task is achieved. This determines whether the high or the low-level policy takes control. We define the end-of-sub-task signal as $e_t(\tau_i) \in \mathcal{E} = \{\texttt{True}, \texttt{False}\}$ which is an indicator that the current sub-task τ_i is finished and the high-level policy should regain control to generate the next sub-task. Therefore, the revised low-level policy can be expressed as $\pi_\tau : \mathcal{S} \times \mathcal{T} \rightarrow \mathcal{A} \times \mathcal{E}$.

3.1 High-level policy

The high-level policy consists of an encoder, the Gated Attention (GA) unit [16], and a recurrent unit. First, we generate a list of 50-dimensional GloVe [21] embedding vectors from words in the language instruction. Then the embedding vectors are fed to a single-layer GRU [22] and the attention mechanism introduced in [23] combines hidden states from the GRU to generate a single instruction feature vector. The image is fed to a series of convolution and ELU [24] layers to generate an image feature vector.

Then the GA takes the instruction and image feature vectors and generates an integrated feature vector. The GA computes attention weights from the instruction to focus on the essential part of the image feature vector. Finally, the integrated feature vector is concatenated with another feature vector from the previous sub-task vector and fed into another single-layer GRU and a fully connected layer. We use a softmax function to generate 5-dimensional sub-task probability distribution for $\hat{\mathcal{T}}$.

3.2 Low-level policy

Once the sub-task τ_i is determined, the low-level policy takes control from the high-level policy and generates actions required to achieve the sub-task. First, it converts the input image to an image feature vector by applying a series of convolutional and ELU layers. Then, sub-task probabilities determined by the high-level policy are used to select one of a few sub-task-specific GRU layers. The activated GRU layer combined with an FC layer generates the final 2-dimensional control vector a_t and the end-of-sub-task signal $e_t(\tau_i)$.

The low-level policy remains in control until the end of the sub-task indicated by the $e_t(\tau_i)$ value. In order to make this transition robust to noisy predictions, we require that at least two out of the three recent predictions of $e_t(\tau_i)$ indicate the end of sub-task. In our implementation, we rely on two different low-level policies for predicting a_t and $e_t(\tau_i)$.

```
Require: Initial state $s_0 \in \mathcal{S}$
Require: Language direction $l \in \mathcal{L}$
Require: World $W : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$
Require: High-level policy $\pi_h : \mathcal{S} \times \mathcal{L} \rightarrow \hat{\mathcal{T}}$
Require: Low-level policies $\pi_\tau : \mathcal{S} \times \mathcal{T} \rightarrow \mathcal{A} \times \mathcal{E}$
  $i, t \leftarrow 0, 0$
  $\tau_0 \leftarrow \pi_h(s_0, l)$
  $e_0(\tau_0) \leftarrow$ False
  while $\tau_i \neq$ finish do
      while $e_t(\tau_i) ==$ False do
          $a_t, e_t(\tau_i) \leftarrow \pi_{\tau_i}(s_t)$
          $s_{t+1} \leftarrow W(s_t, a_t)$
          $t \leftarrow t + 1$
      end while
      $i \leftarrow i + 1$
      $\tau_i \leftarrow \pi_h(s_t, l)$
  end while
  return
```

Algorithm 1: Hierarchical policies for language-grounded conditional driving

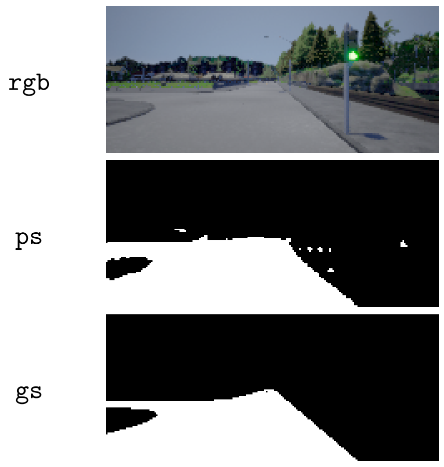

Figure 3: Examples of input dashboard images used in experiments: we compare performance on raw color images (`rgb`) with images trained on predicted (`ps`) or ground-truth segmentation (`gs`) from CARLA [6].

4 Training and Environment

We used CARLA [6] to generate data for training and run experiments. CARLA provides road annotations and an auto-pilot function. We relied on the auto-pilot during training.

In the environments provided in CARLA, all the roads are annotated and an auto-pilot function is implemented. First, we deployed a roaming agent, which randomly decided a direction at each intersection, and recorded observations from the agent at 10 Hz. We use two towns provided by the simulator, Town1 and Town2, as in previous work [3, 4]. Second, we partitioned the trajectory into a set of trajectory snippets corresponding to different sub-tasks. State observation, action, sub-task and end-of-sub-task values, $\langle s_t, a_t, \tau_t, e_t \rangle$, were generated for the snippets. Finally, we combined language data gathered from human users with the information about the snippets to generate realistic natural language instructions corresponding to our environment. In following subsections, we give details of the environment, data generation, and training.

`single`	`double`	`ordinal`
turn left	turn left at first and then right	you re going to take your second left up here
make a left turn	take a left here and then you re going to take a another right turn	you re going to go a little bit further for one block and make a left at the intersection
left	take your left here and then it will be a right turn ahead	take the second left

Table 1: Examples of generated sentences for left turns based on data gathered from realistic interactions. Examples are grouped into three categories, depending on time horizon and complexity of the instruction.

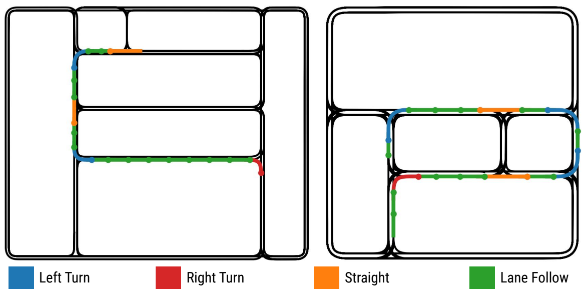

Figure 4: Top-down view showing trajectory segments with sub-task annotations.

4.1 Language generation

We collected language data by designing a two-player driving game with human subjects. In the game, one player was tasked with navigating the vehicle to a goal without any knowledge about the map of the world. The second player was tasked with instructing the first player about directions to goal using only natural languages. From this data, we designed templates to generate language instructions for training the high-level policy that matched our test environments. For more detailed procedure of the language generation, please see Appendix A.

We generated instructions of varying complexity that would be typical for interactions between a human and an autonomous vehicle. We grouped them into three categories (see Table 1 for examples). The first category (`single`) contains instructions that tell the car how the behave at the next intersection. The second category (`double`) contains instructions about the behavior at the two upcoming intersections. Finally, the third category (`ordinal`) contains instructions including ordinal expressions relating to any of two upcoming intersections.

4.2 Training and Trajectory Generation

We collected expert trajectories in simulation and trained each model with supervision analogous to prior work [4, 7]. First, we collected an expert trajectory, $d = \{p_t : t \in T\}$, by releasing a randomly roaming expert with a global planner and PID controller similar to [4], and obtained training sub-task labels based on the annotated road structure, where $p_t = \langle s_t, a_t, \tau_t \rangle$, $s_t \in \mathcal{S} = \mathbb{R}^{3\times200\times88}$ is an image used as a state observation, $a_t \in \mathcal{A}$ is an action, and $\tau_t \in \mathcal{T}$ is a sub-task label at time t.

Then, we partitioned the trajectories into a set of trajectory snippets $D = \{d_i\}$ for training both low-level and high-level policies based on the sub-task labels where $d_i = \{p_t : t \in R_i\}$ and $R_i = [p_i, q_i]$ such that $p_i \leq q_i \wedge p_i \in T \wedge q_i \in T \wedge \tau_a = \tau_b \ \forall a, b \in [p_i, q_i]$. Snippets around intersections were segmented according to which turn was taken into the `left`, `right`, and `straight` policies, corresponding to each of the three possible choices. Intersections were connected by the sub-task `lanefollow`.

In order to make the low-level policy robust, we added a margin before and after each snippet, so that each low-level policy also learns to follow the lane. We used L^1 loss for training the control model and binary cross entropy loss for training the end-of-sub-task model.

Input Modality $\rightarrow$	`rgb`		`gs`		`ps`	
Language Type $\downarrow$	train	test	train	test	train	test
`single`	1.000	1.000	1.000	1.000	0.982	1.000
`double`	0.809	0.439	1.000	0.986	0.976	0.874
`ordinal`	0.813	0.333	1.000	0.938	1.000	0.938
`all`	0.880	0.613	1.000	0.970	0.982	0.926

Table 2: Comparison of results for three different input modalities: ground-truth segmentation `gs`, predicted segmentation `ps`, and raw color images `rgb`.

For high-level policies, we used three or five snippets as a longer segment which included one or two intersections and neighboring `lanefollow` snippets. Within the segment, we drew data points from boundary regions between sub-tasks for training. We use cross entropy loss for training the high-level model. Fig. 4 shows a couple of examples of road segments with sub-tasks annotations.

In addition to one front-facing camera, we put two additional cameras rotated about 14 degrees to the left and to the right, to simulate the images from drifted states. We used three types of images to examine the effects of input modality on generalization from one town to the next, shown in see Fig. 3. These are: (1) `rgb`: color images from the dash-mounted camera, (2) `gs`: ground-truth binary road segmentation images from the simulator; and (3) `ps`: predicted binary road segmentation images from DeepLabv3+ [25] with Mobilenetv2 [26] pretrained with COCO [27] and fine-tuned on the Cityscapes dataset [28].

5 Experiments and Results

We perform a comprehensive evaluation of different properties of our model, and compare it to established baselines. We begin with a quantitative evaluation of generalization abilities of the model for different types of observations. We follow with ablation experiments and comparisons to previously published baselines. Then, we demonstrate robustness to misleading instructions and randomly timed commands. Finally, we show how our model can be used in interactive, real-time scenarios. In the following evaluations, we trained the model on Town1 and tested on both Town1 and Town2. In the training procedure, we draw fixed-length trajectories from the trajectory snippets.

5.1 Input comparison

One challenge with training policies on unstructured input such as images and language is transferring models to new environments. We explored the effects of different input modalities on performance and generalization inspired by the previous work [4] which has shown that segmentation-based policies transfer well between environments. The results can be seen in Table 2.

We performed language-grounded driving by starting from the beginning of the trajectory, given a randomly sampled sentence, and measure the rate of successful episodes. Overall, the model achieved almost perfect results for all levels of language complexity as long as ground truth segmentation was used as observations, even when generalizing across different environments. The performance dropped slightly, when predicted segmentation was used. Finally, raw color images resulted in largest performance drop when the model was transferred across environments, despite good performance on the training environment. Table 2 shows the quantitative evaluation result. The average performance drop from Town1 to Town2 with `rgb` is about 30.13% while the averages performance drops of `ps` and `gs` are about 5.038% and 1.301%. This reaffirms good generalization performance of semantic segmentation. This trend was primarily noticeable for language instructions of highest complexity (`ordinal`). A full comparison with model ablations and input modalities is provided in Table A3 in Appendix.

5.2 Model Comparison

We compared our model (H_{ih}) with three variants of the model and two baseline models, a single policy and a Neural Modular Control (NMC) [7], given ground-truth road segmentation as input (`gs`). A single policy was implemented by extending a high-level policy to directly predict actions. We implemented NMC without attention for post-navigation question answering. The first variant ($H_{\emptyset}$)

Language Type $\rightarrow$ **Model** $\downarrow$	`single` train	`single` test	`double` train	`double` test	`ordinal` train	`ordinal` test	`all` train	`all` test
Single policy	0.721	0.583	0.033	0.000	0.028	0.000	0.288	0.197
Single policy with history	0.684	0.667	0.077	0.000	0.142	0.000	0.311	0.225
NMC [7]	0.218	0.208	0.000	0.000	0.000	0.000	0.081	0.070
$H_\emptyset$: hierarchical baseline	**1.000**	**1.000**	**1.000**	**1.000**	0.969	0.906	0.996	0.986
H_i: $H_\emptyset$ with images	**1.000**	**1.000**	0.979	0.982	**1.000**	0.813	0.990	**0.991**
H_{ih}: H_i with history **(full model)**	**1.000**	**1.000**	**1.000**	0.960	**1.000**	**0.938**	**1.000**	0.970
H_{ihg}: H_{ih} with gated attention	0.984	**1.000**	0.976	0.943	0.972	0.886	0.979	0.954

Table 3: Comparison of our method to both a single policy and a Neural Modular Control (NMC) [7] baseline, and ablation of several different key components, given ground-truth road segmentation as input (`gs`.) Models used in ablation, $H_\emptyset$, H_i, and H_{ih}, are explained in Section 5.2. H_{ihg} replaces the original low-level model with Gated Attention model [16].

Language Type $\rightarrow$ **Model** $\downarrow$	`single` train	`single` test	`double` train	`double` test	`all` train	`all` test
$H_\emptyset$: hierarchical baseline	**1.000**	**1.000**	0.758	0.652	0.828	0.742
H_i: $H_\emptyset$ with images	**1.000**	**1.000**	**1.000**	**0.957**	**1.000**	**0.968**
H_{ih}: H_i with sub-task history **(full model)**	**1.000**	**1.000**	**1.000**	0.928	**1.000**	0.946

Table 4: Evaluation of our approach for misleading language instructions (e.g. "go straight" when no straight road exists). We used ground-truth segmentation images `gs` as input.

is a hierarchical baseline model with a high-level policy which only takes language instruction as its input. A high-level policy in the second variant (H_i) takes the image along with the language instruction but it does not use the sub-task history. Our model (H_{ih}) uses the language instruction, the image, and the sub-task history in the high-level policy. The last variant (H_{ihg}) uses the same high-level model as our model but it replaces the original low-level policy using a few sub-task-specific GRU layers by a new low-level policy which is conditioned by a sub-task label using GA. A single policy was implemented by extending a high-level policy to directly predict actions. Table 3 shows that all variants of our model outperformed the baselines.

We see that both baselines struggled to interpret more complex commands (`double` or `ordinal`). The single policy could learn a turning behavior from fixed-length sub-trajectories but failed to learn to distinguish multiple turns and plan a series of turns. This is understandable, given the length of the trajectories (hundreds of frames), which do not fit in a single recurrent unit. The hierarchical decomposition of the task in our model reduces the complexity of the problem and makes the model trainable with long-time horizon data. The NMC baseline performed poorly even for simple directions (`single`). Lack of an attention mechanisms resulted in poor performance of end-of-sub-task and sub-task prediction.

Among our model and two hierarchical variants of the models, H_{ih}, $H_\emptyset$ and H_i, we could not see a huge performance gap. Transition between sub-tasks is highly dependent on the end-of-sub-task value from low-level policy and that gives the model with a simple high-level policy high performance.

In addition, we replace the original low-level model with the gated-attention model, H_{ihg}, which takes sub-task values as a conditional input. Though this conditional low-level model performs slightly worse than the original model, H_{ih}, its performance is comparable to other ablation models. It implies that switching between a fixed number of special layers is not necessarily needed; for higher-level tasks in future, we can generalize the intermediate representation not restricted to a fixed number of sub-tasks.

5.3 Misleading Instructions

Realistic natural language instructions are often inconsistent and ambiguous. For an autonomous system, it is paramount to handle such instructions and generate only safe behavior. To evaluate our model in such conditions, we generated misleading language instructions containing directions not

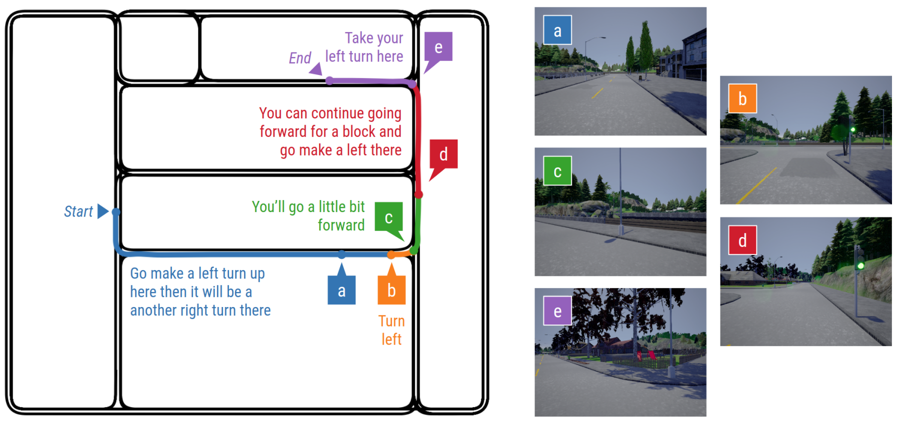
Take your left turn here
End
e
You can continue going forward for a block and go make a left there
d
Start
You'll go a little bit forward
c
Go make a left turn up here then it will be a another right turn there
a
b
Turn left
a
b
c
d
e

Figure 5: An example of interactive driving. The trajectory and the instructions provided by the user are shown on the left. The right side shows images corresponding to the indicated points along the trajectory.

possible to execute given the current world map (e.g. "go straight" for a T-shaped intersection). In such cases, we expect a safe behavior and choose to train the model to stop for impossible straight directions or go straight for impossible turns. We evaluated the resulting model only on misleading instructions.

Table 4 shows the quantitative evaluation of our model for misleading language directions using ground-truth segmentation images as input. Usage of the image in the high-level policy seemed to be an important factor on performance. Performance of the model $H_{\emptyset}$ on complicated language directions is degraded in comparison to other models which use images in the high-level policy. This demonstrates the importance of using image in decision making when it has to deal with misleading language instructions. Our model shows the robustness to misleading instructions.

5.4 Interactive Driving

In realistic settings, a self-driving car will receive instructions from the user at different moments in time, even if the car is currently executing a previous command. To illustrate the robustness of our model to imperfectly timed commands as well as random interruptions, we designed an interactive, real-time driving protocol, where users could provide natural language instructions at any moment in time. The agent interrupts the current plan whenever a new instruction s received. Here, we present and analyse an example of such experiment (see Fig. 5).

The experiment began with the instruction "Go make a left turn up here then it will be a another right turn there." Then the user interrupted the execution of the commands three times at random moments, often while a sub-task such as left turn is currently being executed. This interactive driving example shows that our agent can be used to drive continuously according to user directions, even when frequently interrupted or when given inconsistent commands. Thanks to its hierarchical structure, our model is less sensitive to timing issues; it will complete the current maneuver before executing the next command.

6 Conclusion

We showed a system for linguistic control of a self-driving vehicle from images, and provide an ablation analysis of which components of the network are important for providing the best performance including generalization to new environments. In particular, we showed our model improves on related prior work for visual question answering [7] and extends work in driving using conditional policies. Our future work will focus on even more complex language expressions, with emphasis on objects in the environment.

Acknowledgments

This work was supported by the UW Reality Lab, Facebook, Google, and Huawei. We would also like to thank NVIDIA for generously providing DGX used for this research via the NVIDIA Robotics Lab and the UW NVIDIA AI Lab (NVAIL) and Samsung Scholarship.

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2174–2182, 2017.

[3] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018.

[4] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun. Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*, 2018.

[5] C. Paxton, Y. Bisk, J. Thomason, A. Byravan, and D. Fox. Prospection: Interpretable plans from language by predicting the future. *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[7] A. Das, G. Gkioxari, S. Lee, D. Parikh, and D. Batra. Neural modular control for embodied question answering. *arXiv preprint arXiv:1810.11181*, 2018.

[8] M. J.-Y. Chung*, A. Pronobis*, M. Cakmak, D. Fox, and R. P. N. Rao. Autonomous question answering with mobile robots in human-populated environments. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

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

[10] L. Ke, X. Li, Y. Bisk, A. Holtzman, Z. Gan, J. Liu, J. Gao, Y. Choi, and S. Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. *arXiv preprint arXiv:1903.02547*, 2019.

[11] P. Shah, M. Fiser, A. Faust, J. C. Kew, and D. Hakkani-Tur. Follownet: Robot navigation by following natural language directions with deep reinforcement learning. *arXiv preprint arXiv:1805.06150*, 2018.

[12] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[13] X. Liang, T. Wang, L. Yang, and E. Xing. CIRL: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 584–599, 2018.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[15] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov. Combining neural networks and tree search for task and motion planning in challenging environments. *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, 2017.

[16] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[17] V. Blukis, N. Brukhim, A. Bennett, R. A. Knepper, and Y. Artzi. Following high-level navigation instructions on a simulated quadcopter with imitation learning. *arXiv preprint arXiv:1806.00047*, 2018.

[18] K. Shiarlis, M. Wulfmeier, S. Salter, S. Whiteson, and I. Posner. TACO: Learning task decomposition via temporal alignment for control. *arXiv preprint arXiv:1803.01840*, 2018.

[19] S. Krishnan, R. Fox, I. Stoica, and K. Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. *arXiv preprint arXiv:1710.05421*, 2017.

[20] J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 166–175. JMLR. org, 2017.

[21] J. Pennington, R. Socher, and C. Manning. GloVe: global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[22] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[23] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[24] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[25] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[28] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

Appendix

A Language generation

To create the language dataset, we originally conducted a two-player game with human subjects to collect speech signals of commands and corresponding driving controls. In the game, two players are asked to collaboratively drive a car to reach three randomly spawned goals. While one player drives a car without knowing where the destination is, the other player reads a map and gives direction to the driver. After transcribing the collected audio data, we removed the sentences with actions that cannot be taken in the current environment and removed expressions mentioning objects or structures. Then we divide expressions into prefix, body, and suffix and cluster those expressions to transform the sentence into templates. Finally, we generated sentences for each combination of sub-tasks with the templates. In the implementation, we used a keyword to represent each type of combination.

We counted the number of expressions in the raw dataset for each. Table A1 shows the number of sentences for each keyword. The keyword '`other`' and '`extra`' represents the sentences contain the actions that cannot be taken in the current environment and the sentences that do not have any

Keywords	**Sources**	
	game	templates
`left`	1,093	150
`right`	1,016	150
`straight`	1,199	454
`left,left`	3	269,550
`left,right`	20	135,000
`left,straight`	22	408,600
`right,left`	28	135,000
`right,right`	2	269,550
`right,straight`	14	408,600
`straight,straight`	1	85
`first,left`	9	102,150
`first,right`	4	102,150
`second,left`	97	105,450
`second,right`	89	105,450
`other`	913	N/A
`extra`	379	N/A

Table A1: Number of sentences collected from the preliminary two-player driving game (game) and the templates for training (templates). From the game, we also classified sentences which are out of actions defined in the environment we used in the training as `other` and sentences which do not contain meaningful commands as `extra`.

meaningful commands, respectively. The total of the counted expressions is 4889 and 4600 sentences have single command '`left`', '`right`', '`straight`', '`other`', '`extra`'.

This high percentage of single command is due to the nature of the language in the driving setting where the reactive instruction should be given within a short amount of time. This shows that concentrating on instructive sentences is a reasonable approach in the context of driving. Another point worth noting on the dataset is that people make a lot of mistakes in commanding or driving. Sometimes a commander repeats the same command until the driver finishes that action or cancels previous actions by adding a new command. Manual pruning was necessary to make the dataset feasible to train on. As a trade-off, the distribution of sentences can be made more realistic than that coming from pre-recorded driving trajectories.

As a result of this dataset imbalance, we augment natural-language phrases according to a couple of simple rules. For the sentences with two commands, we concatenated expressions from a single keyword. The dictionary shows the 14 keywords we used in the paper and the corresponding number of sentences is shown in Table A1. When we use these sentences in the training, we draw a sentence from these lists with uniform distribution. For ordinary keywords, two groups of lists were used: one from the direct combination of two sentences of single keywords and the other one from the replacement of the keyword, such as replacing '`left`' with '`second left`'. In the training, for those with multiple groups, the group is first drawn and then the sentence is drawn from the group.

B Language examples

We show two examples of transcribed speech data from the preliminary two-player game experiments. Note that certain types of behavior such as going backward, reaching the target, and slowing down were excluded from the training dataset.

“oh there’s a map all right go straight”, “and you’re going to turn right”, “that’s good keep going straight”, “and take your first left”, “and slow down”, “all right can you see the green square”, “great”, “okay so now you want to go straight”, “and you’ll take a left at the first building”, “that’s good that’s good keep going straight”, “and take a left”, “and take a right”, “now straight”, “and take a left”, “went a little too far so reverse and back it up”, “all right you doing good”, “go a little bit forward”, “yep there it is”, “you got it”, “okay so now you’re going to want to turn around”, “you’re going to back it up a little bit”, “looking good no collisions so far”, “all right now you’ll take a right”, “yep”, “now go straight”, “now take a left”, “take a right”, “go straight as fast as you can”, “and you’ll take a left”, “now right”, “and the exit is right up here”, “congratulations”.
“go straight”, “slow down a little bit”, “make a right turn”, “it’s going to be a narrow street so go straight”, “and then you’re going to make a left turn when you see the first”, “go straight”, “and make a left turn here”, “make a left turn”, “and go straight”, “and do you see the green spot”, “park there”, “okay”, “go straight”, “turn left turn here”, “and another left turn”, “and you’re going to make a right turn here”, “and make another left turn”, “go straight”, “just go straight”, “and make another left turn”, “left turn”, “make another right turn right turn”, “go straight”, “skip this”, “and then make a left turn here left turn”, “left turn”, “left”, “and park there”, “wait for me”, “can you go back”, “reverse”, “and then left turn”, “go little more little more”, “go back back”, “back it out a little more”, “good job”, “okay go straight”, “to your left side to your left side”, “go straight”, “keep going go straight”, “pass the street intersection and then go”, “go straight”, “yeah can you go little faster”, “and then make a left turn here”, “okay try your best”, “make a left turn”, “left”, “and you’re going to make another right turn right turn here right right”, “okay”, “go straight just keep going”, “pass this”, “okay slow down a little bit”, “and you going to make a left turn okay”, “go straight”, “and then make a left turn”, “left here and then left”, “make a right turn right away”, “right here right here”, “and then another right”, “right slow down slow down”, “okay go straight”, “and then the green will be on your left side left side”, “cool we are done”.

Table A2: Language from two instances of the preliminary two-player driving game.

		Input Modality					
Model	**Language Type**	`rgb`		`gs`		`ps`	
		train	test	train	test	train	test
$H_\emptyset$: hierarchical baseline	`single`	1.000	0.958	1.000	1.000	1.000	1.000
	`double`	0.763	0.437	1.000	1.000	0.893	0.904
	`ordinal`	0.813	0.490	0.969	0.906	0.938	0.813
	`all`	0.858	**0.621**	0.996	0.986	0.939	0.923
H_i: $H_\emptyset$ with image	`single`	1.000	0.958	1.000	1.000	1.000	1.000
	`double`	0.821	0.411	0.979	0.982	0.945	0.856
	`ordinal`	0.674	0.344	1.000	1.000	1.000	0.813
	`all`	0.867	0.586	0.990	**0.991**	0.973	0.898
H_{ih} **(full model)**	`single`	1.000	1.000	1.000	1.000	0.982	1.000
	`double`	0.809	0.439	1.000	0.986	0.976	0.874
	`ordinal`	0.813	0.333	1.000	0.938	1.000	0.938
	`all`	**0.880**	0.613	**1.000**	0.970	**0.982**	**0.926**

Table A3: Comparison of three different input modalities: ground-truth segmentation `gs`, predicted segmentation `ps`, and raw color images `rgb`. The highest values from `all` language type are highlighted. Models used in ablation, $H_\emptyset$, H_i and H_{ih}, are described in Section 5.2.