

UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

Explainable and Advisable Learning for Self-driving Vehicles

Permalink

<https://escholarship.org/uc/item/1b97h2dg>

Author

Kim, Jinkyu

Publication Date

2019

Peer reviewed|Thesis/dissertation

Explainable and Advisable Learning for Self-driving Vehicles

by

Jinkyu Kim

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor John Canny, Chair

Professor Trevor Darrell

Professor David Whitney

Fall 2019

The dissertation of Jinkyu Kim, titled Explainable and Advisable Learning for Self-driving Vehicles, is approved:

Chair	_____	Date	_____
	_____	Date	_____
	_____	Date	_____

University of California, Berkeley

Explainable and Advisable Learning for Self-driving Vehicles

Copyright 2019
by
Jinkyu Kim

Abstract

Explainable and Advisable Learning for Self-driving Vehicles

by

Jinkyu Kim

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor John Canny, Chair

Deep neural perception and control networks are likely to be a key component of self-driving vehicles. These models need to be explainable - they should provide easy-to-interpret rationales for their behavior – so that passengers, insurance companies, law enforcement, developers, etc., can understand what triggered a particular behavior. Explanations may be triggered by the neural controller, namely *introspective* explanations, or informed by the neural controller’s output, namely *rationalizations*.

Our work has focused on the challenge of generating introspective explanations of deep models for self-driving vehicles. In Chapter 3, we begin by exploring the use of visual explanations. These explanations take the form of real-time highlighted regions of an image that causally influence the network’s output (steering control). In the first stage, we use a visual attention model to train a convolution network end-to-end from images to steering angle. The attention model highlights image regions that *potentially influence* the network’s output. Some of these are true influences, but some are spurious. We then apply a causal filtering step to determine which input regions actually influence the output. This produces more succinct visual explanations and more accurately exposes the network’s behavior. In Chapter 4, we add an attention-based video-to-text model to produce textual explanations of model actions, *e.g.* “the car slows down because the road is wet”. The attention maps of controller and explanation model are aligned so that explanations are grounded in the parts of the scene that mattered to the controller. We explore two approaches to attention alignment, strong- and weak-alignment.

These explainable systems represent an externalization of tacit knowledge. The network’s opaque reasoning is simplified to a situation-specific dependence on a visible object in the image. This makes them brittle and potentially unsafe in situations that do not match training data. In Chapter 5, we propose to address this issue by augmenting training data with natural language advice from a human. Advice includes guidance about what to do and where to attend. We present the first step toward advice-giving, where we train an end-to-end

vehicle controller that accepts advice. The controller adapts the way it attends to the scene (visual attention) and the control (steering and speed). Further, in Chapter 6, we propose a new approach that learns vehicle control with the help of long-term (global) human advice. Specifically, our system learns to summarize its visual observations in natural language, predict an appropriate action response (e.g. “I see a pedestrian crossing, so I stop”), and predict the controls, accordingly.

To Hyerin and Minoo

Contents

Contents	ii
1 Introduction	1
2 Deep Traffic Light Detection for Self-driving Cars from a Large-scale Dataset	4
2.1 Problem Statement	4
2.2 Deep Traffic Light Detection Model	6
2.3 Dataset	10
2.4 Experiments	11
2.5 Related Work	15
3 Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention	17
3.1 Problem Statement	17
3.2 Interpretable Driving Model	19
3.3 Experiments	26
3.4 Related Work	30
4 Textual Explanations for Self-driving Vehicles	34
4.1 Problem Statement	34
4.2 Explainable Driving Model	37
4.3 Berkeley DeepDrive eXplanation Dataset (BDD-X)	41
4.4 Experiments	45
4.5 Related Work	51
5 Grounding Human-to-Vehicle Advice for Self-driving Vehicles	55
5.1 Problem Statement	55
5.2 Advisable Driving Model	57
5.3 Honda Research Institute-Advice Dataset (HAD)	61
5.4 Experiments	63
5.5 Related Work	68

6	Advisable Learning for Self-driving Vehicles by Internalizing Observation-to-Action Rules	70
6.1	Problem Statement	70
6.2	Advisable Learning	73
6.3	Experiments	78
6.4	Related Work	84
	Bibliography	87

Acknowledgments

I would like to thank my advisor John Canny for his patience, support, and motivation through my PhD studies. With his guidance, I have been able to discover a research field I am truly passionate about. Additionally, I would like to thank Anca Dragan for serving my qualification committee, Trevor Darrell and David Whitney for serving on both my qualification and dissertation committees. I would also like to thank all those who have mentored me through my PhD, in particular Anna Rohrbach, Zeynep Akata, Laura Waller, Brian Barsky, and Ren Ng. I would also like to thank my Phantom AI internship collaborators Chankyu Lee, Hyunggi Cho, Myung Hwangbo, Youngwook Paul Kwon, Jaehyung Choi, and Jihyun Yoon, my Honda Research Institute USA internship collaborators Teruhisa Misu, Yi-Ting Chen, Ashish Tawari, and Chiho Choi, and my Waymo internship collaborators Mayank Bansal, Woojong Koh, and Drago Anguelov.

Throughout my years at Berkeley, I have been fortunate to have so many collaborators at Berkeley: Cecilia Zhang, Ye Xia, Donghan Lee, Biye Jiang, Daniel Seita, Xinlei Pan, Philippe Laban, Forrest Huang, David Chan, Roshan Rao, Suhong Moon, Yang Gao, Dequan Wang, Coline Devin. I would like to thank my friends for providing emotional support: Saehong Park, Yeojun Kim, Kiwoo Shin, Philjoon Kang, Hyungdong Ha, Sangjae Bae, Bumjoon Seo, Wonyeol Lee and Seunghyun Park.

I would also like to thank Samsung Scholarship for a financial support covering both tuition and stipend.

Finally, I would like to thank my family including my father Hangbok, mother Bangyeo, and sister Chohee. Thank you for providing me with everything I needed to succeed in life. Most of all, I would like to thank my wife Hyerin and son Minoo for their supports and love throughout my life in Berkeley. This dissertation is dedicated to you.

Chapter 1

Introduction

Whereas classical AI systems involved carefully-crafted features and representations, one of the new powers of deep learning methods is the ability to learn very effective latent representations from data. Deep neural perception and control networks are likely to be a key component of self-driving vehicles. In Chapter 2, we first explore that these deep learning methods can be integrated into a self-driving vehicle control model. Unfortunately, whereas human-designed feature maps are often easy to understand, deep representations may not be. While there have been some successes in visualizing deep models on image data, many models remain cryptic. And even among the successes, success is partial: *i.e.* while many individual features are interpretable, many others in the same network are not. A deep network such as a visually-driven action policy embodies tacit, situated knowledge. It is represented as a complex set of learned weights and produces action in response to inputs, with a priori no other abstraction or higher explanation.

In Chapter 3, we first explore to make the model interpretable using visual attention. The attention model weights different areas of the image differently and effectively ignores certain areas completely. This can be visualized with a dynamic heatmap. We have observed that visual attention can be integrated into a state-of-the-art vehicle control model without loss of control accuracy. The collateral benefit of using spatial attention is that it is immediately interpretable: areas that are “dark” in the attention map are masked in the middle of the network and can have no effect on its control output. The converse assertion is not sound however: “bright” areas in the attention map are not necessarily the most important for the vehicle’s control behavior. Attention maps (human attention as well) must be conservative in discarding visual input. The goal of attention is to focus on areas of the image that might be important. Only after images are processed, we can infer the actual influence of those images on the full network. So attention maps are likely to generate false positives (to be concrete, the model often attends to foliage where there might be street signs). Therefore, we add an extra layer of causal filtering: the attention map is clustered into spatio-temporal blobs that roughly correspond to objects in the underlying images (they do not have perfect correspondence with objects since we leave it to human observers to interpret them). We then systematically remove each blob in the attention layer by setting the weights to zero

over its support. If there is a significant change in controller output, then the blob is retained in the attention map. If not, it is discarded. A typical reduction in the number of blobs is more than 2x. By doing this we obtain a much more focused (specific) attention map and we know that the blobs that remain do causally affect the controller.

In Chapter 4, we next move to textual explanations. We use the Berkeley DeepDrive-eXplanation (BDD-X) dataset which is constructed the following way. Video from a number of drivers in the United States was collected using small camera devices placed on the dashboard just above the steering wheel. This data was subsequently annotated using Amazon Mechanical Turk where the task was to view the video and provide both a description of the drivers’ actions and then an explanation for them. The turker was asked to put themselves in the position of a driving instructor providing explanations to a student driver. The explanations obtained in this way are pure rationalizations since they were generated by an observer, not the driver. While it might be more desirable to use self-report data from drivers themselves, the collection of such a dataset poses a number of challenges. For now we have used the existing Berkeley DeepDrive-Video (BDD-V) dataset, which has the advantage that we at least do not need to determine whether human explanations are based on true introspection or rationalization since it is always the latter for this dataset. We use this data to train an explanation generator. It remains for us to try to generate explanations that are grounded in the model’s actual behavior even though there is no such information in the training dataset. We return again to spatial attention – our controller has already highlighted regions in the image that affect its output using its own attention map augmented by causal filtering. Spatial and temporal attention has also been successfully applied for generating text annotations of videos. That is, the explanation module also highlights the areas of the video that it used in its explanations. These two attention models should align in an appropriate sense: The explanation module should not causally attend to regions of the image that did not causally affect the controller output. In other words, the explanation module’s causal attention map should be a subset of the controller’s. We have to take care in dealing with time – the controller’s “time” is the same as the timestamp in the video stream. A grammatical explanation will often generate words (and attend to corresponding image regions) in a quite different order from their prominence for the controller. Thus our two attention maps are compared using a metric which allows flexibility in temporal alignment.

These explainable systems represent an externalization of tacit knowledge. The network’s opaque reasoning is simplified to a situation-specific dependence on a visible object in the image. It is better considered as part of socialization, where explanations are offered in a master-apprentice context with the control policy serving as an instructor. We explore advisable AI systems that are built directly on our work on explanations. The explainable driving model project has connected with a number of industry researchers through the Berkeley Deep Drive initiative. From them, and from other prospective users we have had a lot of feedback about the need for users not only to understand the driving controller, but to influence it. Since users are typically not paying attention to the vehicle’s detailed behavior, such influence should be high-level. We use the term “advisable AI” to convey the notion of partnership between human and machine. It contrasts with commanding, or

even constraining the robot’s actions. It is a form of user customization, although it is typically quite dynamic since the user’s preferences for driving behavior are likely to change frequently. Human-to-vehicle advice can take a variety of forms, with different levels of urgency, *e.g.* “drive more slowly/gently”, “Avoid roads with speed bumps”. Some of these are commands that should always be followed. Markers such as “always”, “don’t” “avoid” indicate that the user expects their directions to be followed.

In Chapter 5, we observe that deep neural control networks, which are trained on large datasets to imitate human actions, lack semantic understanding of image contents. This makes them brittle and potentially unsafe in situations that do not match training data. We propose to address this issue by augmenting training data with natural language advice from a human. Advice includes guidance about what to do and where to attend. We present a first step toward advice giving, where we train an end-to-end vehicle controller that accepts advice. The controller adapts the way it attends to the scene (visual attention) and the control (steering and speed). Attention mechanisms tie controller behavior to salient objects in the advice. We evaluate our model on a novel advisable driving dataset with manually annotated human-to-vehicle advice called Honda Research Institute-Advice Dataset (HAD). We show that taking advice improves the performance of the end-to-end network, while the network cues on a variety of visual features that are provided by advice. This was the first paper on the use of advice, but this design is most appropriate for turn-by-turn (short duration) advice. Since our data comprised short clips, advice was effective throughout the clip. It will be worth exploring other styles of advice, such as per-ride advice (gentle, fast, etc) and rule-based global advice.

In Chapter 6, we propose to use human advice in the form of observation-action rules. Specifically, we propose a new approach that learns vehicle control with the help of human advice. Specifically, our system learns to summarize its visual observations in natural language, predict an appropriate action response (*e.g.* “I see a pedestrian crossing, so I stop”), and predict the controls, accordingly. Moreover, to enhance interpretability of our system, we introduce a fine-grained attention mechanism which relies on semantic segmentation and object-centric RoI pooling. We show that our approach of training the autonomous system with human advice, grounded in a rich semantic representation, matches or outperforms prior work in terms of control prediction and explanation generation. Our approach also results in more interpretable visual explanations by visualizing object-centric attention maps.

Chapter 2

Deep Traffic Light Detection for Self-driving Cars from a Large-scale Dataset

2.1 Problem Statement

Traffic lights detection problem is one of the key challenges for autonomous vehicle controllers in urban areas. While a number of approaches for traffic light detection have been proposed, these methods often require a prior knowledge of map and/or show high false positive rates. Recent successes suggest that deep neural networks will be widely used in self-driving cars, but current public datasets do not provide sufficient amount of labels for training such large deep neural networks. In this paper, we developed a two-step computational method that can detect traffic lights from images in a real-time manner. The first step exploits a deep neural object detection architecture to find true traffic light candidates. In the second step, a point-based reward system is used to eliminate false traffic lights out of the candidates. To evaluate the proposed approach, we collected a human-annotated large-scale traffic lights dataset (over 60 hours). We also designed a real-world experiment with an instrumented self-driving vehicle and observed that the proposed method was able to handle false traffic lights substantially better compared with the baseline considered.

Self-driving vehicle control has recently made remarkable progress. These controllers involve a variety of sophisticated algorithms for perception, behavioral/motion planner, and dynamics controllers. Despite their recent success in some driving scenarios (*i.e.* highway driving), there still remain new challenges for urban driving that involves more complex driving scenarios that need interaction with traffic controls, vehicles, pedestrians, etc. Especially,

*This work has been presented at the 20th IEEE International Conference on Intelligent Transportation Systems (ITSC), 2017.

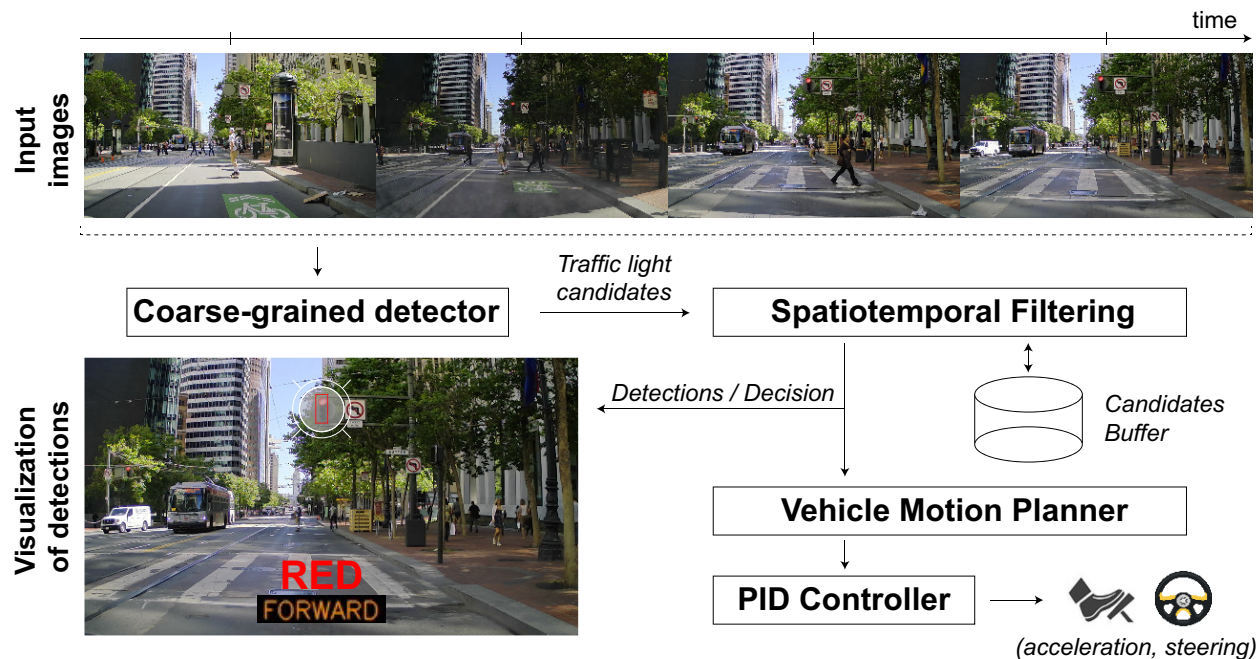


Figure 2.1: Our model detects traffic lights, *e.g.* a red circle, from an input raw image at each timestep. Our model consists of two major steps: (1) coarse-grained detector that utilizes deep neural object detection architecture and is tuned to discover as many true traffic lights as possible. (2) the spatiotemporal filtering step that eliminates false traffic lights with respect to features extracted from both spatial and temporal domains. The output of our proposed detector is then fed into the vehicle motion planner and the PID controller that computes corresponding acceleration and steering angle commands.

traffic lights pose a challenging (computer vision) problem when subject to varying lighting, view distances, and weather conditions. Though its importance for automated driving in urban areas, conventional approaches showed insufficient reliability and robustness enough to be used in autonomous systems in the urban environment without utilizing prior knowledge.

Recent successes suggest that deep neural networks will be widely used in self-driving cars, especially for a perception part. Behrendt *et al.* [8] utilized the “You Only Look Once” (YOLO) network architecture followed by a small classification convolutional network to detect traffic lights. For obtaining a reliable and robust performance from such a large deep neural network, a large amount of dataset is strongly required to provide a large variation in environmental conditions. However, the current publicly available datasets show lacks such variation. For example, the VIVA challenge [62] for traffic lights only provide 44 minutes of data and the Bosch Small Traffic Lights Dataset [8] provides only about 5,000 images (less than 3 hours), which is insufficient from the view of the conventional way of training

such large deep neural networks. We, therefore, create a large dataset, which provides over 60 hours of driving images that cover diverse driving conditions (*i.e.* lighting and weather). Thus, we argue this dataset will be ideal for further traffic light detection studies.

Collecting a large-scale dataset is only part of a story. Reliable traffic light detector strongly requires low false negative (or discovering as many true traffic lights as possible) and false positive (or eliminating false traffic lights) rates, while maintaining a high detection accuracy. Here, we propose a new computational method for traffic light detection, which consists of two major steps: (1) coarse-grained traffic light detection and (2) spatiotemporal filtering of the detected traffic lights. The first step considers individual images and collects traffic light candidates using a deep neural object detection architecture. The focus of this step is to reduce false negatives (FNs) or to discover as many true traffic lights as possible. The second step is then to eliminate false positives (FPs) by considering spatial and temporal characteristics of traffic lights. To distinguish true and false traffic lights, we propose a point-based reward system where each detected traffic lights earn rewards and the final decision is made based on these rewards. To demonstrate the effectiveness of applying the proposed method to self-driving vehicles, we test with an instrumented vehicle and successfully drive 6 kilometers on city streets in the San Francisco Bay Area, California, USA.

Our contributions can be summarized as follows: (i) We propose a new computational method for accurately detecting traffic lights from a raw input image in a real-time manner. (ii) We generated a large-scale traffic lights dataset with over 71,771 images (over 60 hours) with human annotated bounding boxes. (iii) We demonstrate the effectiveness of applying our proposed approach by conducting a real-world experiment (driving over 6 kilometers including 17 intersections with traffic lights) with an instrumented vehicle.

2.2 Deep Traffic Light Detection Model

Here, we propose a method that accurately and reliably detects traffic lights from a stream of images captured by a front-view dash-cam attached to the windshield. As we depicted in Figure 2.2, the proposed method contains two major steps: (1) coarse-grained traffic light detector and (2) spatiotemporal filtering of the traffic lights candidates. In the first step (coarse-grained detector), traffic light candidates from each image are collected by utilizing a deep neural object detection architecture. The main focus of this step is to discover the true traffic lights as many as possible (*i.e.* reducing the number of false negatives). Thus, it is possible that the traffic light candidate collection may contain false positives. In the second step (spatiotemporal filtering), we eliminate such erroneously detected traffic lights by simultaneously considering other traffic lights over time and space. To distinguish between true and false traffic lights, we use a point-based reward system where each detected traffic lights earn rewards with respect to features extracted from both spatial and temporal domains.

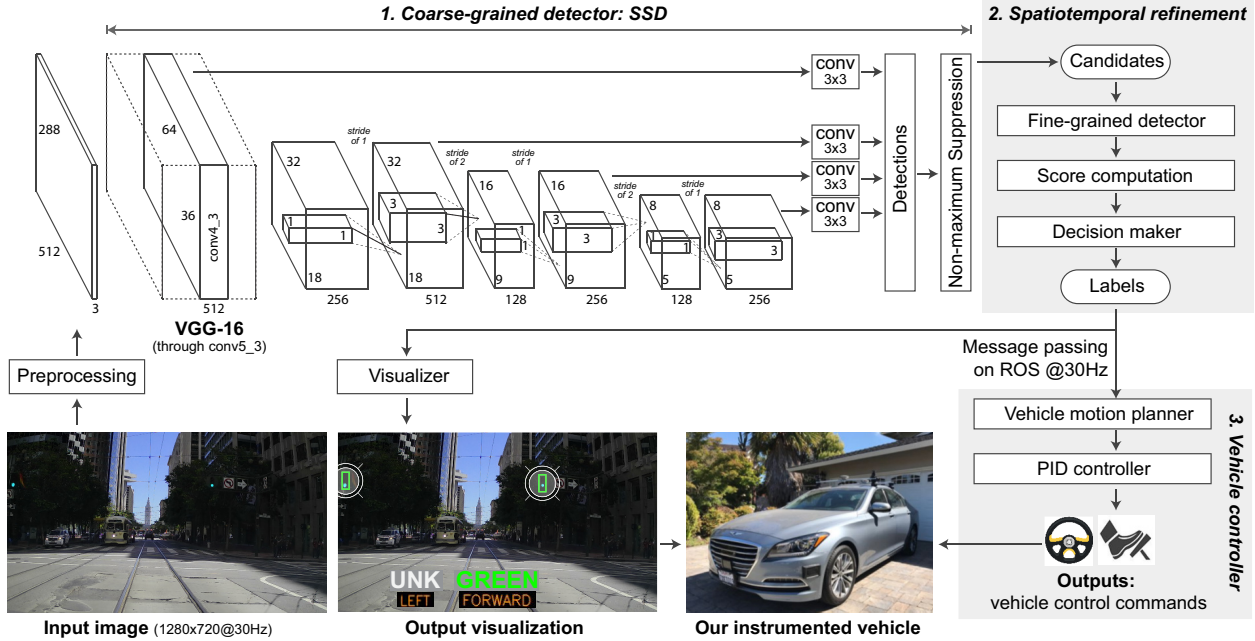


Figure 2.2: An overview of our proposed model. It can be understood in three parts: (i) a coarse-grained detector that utilizes the deep neural perception network architecture called SSD (Single-Shot multi-box Detector [50]), (ii) a spatiotemporal filtering, and (iii) a vehicle controller. To demonstrate the feasibility of applying our model to self-driving cars, we use an instrumented autonomous vehicle which uses the output of our model as an input to control its dynamics.

2.2.1 Preprocessing

We use an input image that is resized to $288 \times 512 \times 3$ with bilinear interpolation algorithm, hence to reduce computational burdens for a real-time detection. For the images with different aspect ratios, we cropped the height to match the ratio. Following a common practice in image classification tasks, we subtracted the mean RGB value to achieve zero-centered inputs, which are originally in different scales. Note that our dataset contains images where the camera gains are automatically calibrated to obtain high-quality images. During the testing process, we also used a cropped image in the center part of the image, where traffic lights are commonly observed in that area. Thus, a batch of two images (*i.e.* whole and cropped images) are fed into our detector.

2.2.2 Coarse-grained Traffic Light Detection

Traffic light detector strongly requires showing reliable performance in real-time and working for both small (*i.e.* 3×9 pixels) and large objects with low false positive and low false negative rates, while maintaining a high detection accuracy. For example, a false red traffic light will lead the autonomous vehicle to abruptly stop while driving, while a missed red light will cause the vehicle to go through an intersection originally with red lights in its course of driving.

In this coarse-grained traffic light detection step, we focus to reduce false negative (FN) rates or to collect as many true traffic lights as possible. We utilize the Single-Shot multi-box Detector (SSD) [50] that has been shown to be an effective tool for an object detection task. Note that we use the SSD architecture that has shown improved detection accuracy in other benchmarks than YOLO network architecture, which was utilized in the existing work by Behrendt *et al.* [8]. More modern architecture, such as Mask R-CNN [25], may provide better detection accuracy, but we leave this comparison for future work. The SSD model is based on a convolutional network and takes the whole image as an input and predicts a fixed-size collection of bounding boxes and corresponding confident scores for the presence of object instances in those boxes. The final detections are then produced followed by a non-maximum suppression step – all detection boxes are sorted on the basis of their predicted scores, and the detections with maximum score is then selected, while other detections with a significant overlap are suppressed. As we described in Figure 2.2, we use a standard VGG-16 network architecture [72] as a base convolutional network, which is pre-trained on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset [69]. Auxiliary structures – convolutional predictor and the additional convolutional feature extractor – are used following the work by Liu *et al.* [50].

2.2.2.1 Training Objective.

The loss function \mathcal{L} ($= \mathcal{L}_{loc} + \mathcal{L}_{conf}$) is a weighted sum of two types of loss: (1) the localization loss \mathcal{L}_{loc} measures a Smooth L_1 loss between the predicted and the ground-truth bounding box in a feature space. (2) The confidence loss \mathcal{L}_{conf} is a softmax loss over multiple classes confidences. For more rigorous details, refer to Lie *et al.* [50].

2.2.2.2 Data Augmentation.

To train a robust detector to various object sizes, we use random cropping (the size of each sampled image is $[0.5, 1]$ of the original image size with fixed aspect ratio) and flipping to yield consistent improvement. Following the work by Liu *et al.* [50], we also sample an image so that the minimum Jaccard overlap with the objects is $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Note that each sampled image is then resized to a fixed size followed by photometric distortions with respect to brightness, contrast, and saturation.

2.2.3 Characterizing Traffic Lights

According to our analysis, the traffic lights appearing in the detection pipeline possess the following characteristics:

- (C.1) As the confidence of a traffic light candidate decreases, so does the possibility of this being a true traffic light.
- (C.2) The possibility of a traffic light candidate being true increases if the traffic light candidate is detected again in next timestep at almost the same location.
- (C.3) If multiple traffic lights of the same category (*i.e.* red, yellow, and green) are detected in a scene, then they are usually true traffic lights differently located (*i.e.* multiple traffic lights are installed at an intersection).
- (C.4) Traffic lights shall be located following the governmental guideline (*i.e.* at least one of the signal faces shall be located at an intersection mounted on the mast arm.), hence the possibility of a traffic light candidate being true increases as its location gets close to the usual.

As is evident above, examining traffic lights individually is not sufficient, and multiple traffic light candidates over space and time should be considered simultaneously.

2.2.4 Spatiotemporal Filtering

2.2.4.1 Fine-grained Detector

Recall from Section 2.2.1, we re-scaled images by 40% to reduce the computational burdens for a real-time system. We observe that the classification performance of our coarse-grained detector slightly decreases as we have smaller traffic lights (*i.e.* seen from a farther distance). Thus, we utilize an additional small classification network, called fine-grained detector, that has a high-resolution input. All bounding boxes from the coarse-grained detector are cropped and rescaled to 100×100 pixels, they are then fed into the fine-grained detector. For training, we collect image patches that are cropped and rescaled from the ground-truth dataset. Overall, we collect 24,991 and 6,248 patches for training and validation, respectively.

2.2.4.2 Score Function

According to our traffic light characterization (see C.1–C.4), we need to examine multiple traffic light candidates simultaneously for accurate traffic light status recognition. In addition, we set the confidence threshold value of the coarse-grained detector so as to minimize the number of FNs (*i.e.* the true traffic lights that are erroneously left undetected). Consequently, it is likely that the traffic lights detected in the previous step contain false traffic lights that further need to be filtered out. In this spatiotemporal filtering step, we seek to

resolve these issues using a point-based reward system where each detected traffic lights earn points with respect to the following characterizations:

- (S.1) Each traffic light candidate has its own score for being true, and its score is accumulated in the next timestep if detected again under our matching criterion (*i.e.* euclidean distance between centers of each candidate).
- (S.2) Every traffic light candidates from coarse-grained detector earn a reward R at each timestep.
- (S.3) Scores are discounted by a pre-specified discount rate γ at each timestep.

Concretely, the score function $s_j(t)$ for a candidate j is defined as follows:

$$s_j(t) = \min(S_{max}, Rc_j(t) + \gamma s_j(t - 1)) \quad (2.1)$$

where $c_j(t) \in [0, 1]$ is the confidence value computed by the coarse-grained detector. A maximum score is set to S_{max} .

2.2.4.3 Decision

The output of this step is a tuple of the current traffic light status. For each type k of traffic light signals (*i.e.* $k \in \{\text{turning left, going forward, turning right}\}$) and each traffic light status (*i.e.* unknown, red, yellow, and green), we accumulate scores over traffic light candidates and output the status of the maximum score.

$$o_k(t) = \underset{i \in \{\text{red, yellow, green, unknown}\}}{\text{argmax}} \sum_j \mathbf{1}(i, j) s_j(t) \quad (2.2)$$

where $\mathbf{1}(i, j)$ is an indicator function that is 1 if j -th candidate has the same status as i , otherwise 0.

2.3 Dataset

In order to effectively train and evaluate a deep neural perception approach, we have collected a large-scale traffic lights dataset. Our dataset contains RGB color images captured by a dashcam mounted behind the front mirror of the vehicle. Each image has the resolution of 1280×720 pixels. We provide the dataset statistics in Table 2.1. Our dataset is composed of over 60 hours of driving taken in diverse driving conditions, *e.g.* day/night, city/residential areas, etc. We have collected 71,771 images mainly in the San Francisco Bay Area in California, USA. To avoid high similarity between images, we sample images at every 3 seconds. Overall, 34,604 are labeled, the minimum size of labeled traffic lights is approximately 3 (width) $\times 9$ (height) pixels. We also introduce a training and a test set, containing 64,607 and 7,164 images, respectively. In Figure 2.3, we illustrate the distribution of the different traffic light states, which have eight categories: off, too small to annotate, green (circle), red (circle), yellow (circle), green (left-turn), red (left-turn), and yellow (left-turn).

2.4 Experiments

2.4.1 Training and Evaluation Details

For training a coarse-grained detection model, we use the stochastic gradient algorithm (SGD). Unless stated, we use default hyper-parameters following the work by Liu [50]. Our model took less than 3 days to train on three NVIDIA Titan X Pascal GPUs. Our implementation is based on a deep learning framework called Caffe [35].

2.4.2 Quantitative Analysis

As shown in Figure 2.4, we first measured the classification performance of our coarse-grained traffic light detector in terms of recall and precision. We tested with different threshold values in $[0, 1]$. Note that a good classifier will have higher precision and recall values. The coarse-grained detector seeks to minimize the number of FNs (or undetected true traffic lights) to maintain high recall – suggesting that few true traffic lights went undetected by using this method. In the cases tested, maintaining a high recall increases the number of FPs (or detected false traffic lights), which support the need to use the refinement step. The numbers of training example for the yellow circle and left lights are smaller than other colors, and we observe the classifier shows poor classification performance for the yellow circle and left lights. It would also worth exploring the use of other types of more expressive neural networks, which may give a performance improvement over our network configuration [32]. However, exploration of other architectures would be out of our scope.

Table 2.1: Dataset details with the comparison to other publicly available datasets: the VIVA Challenge for traffic lights [62] and the Bosch Small Traffic Lights dataset [8].

	VIVA [62]		Bosch [8]		Ours	
	Training	Testing	Training	Testing	Training	Testing
#Images	20,526	22,481	5,093	8,334	64,607	7,164
FPS	16	16	1/2	15.6	1/3	1/3
#Annotations	54,161	64,170	10,756	13,493	31,239	3,365
#Hours	$\approx 22\text{min}$	$\approx 22\text{min}$	$\approx 2.8\text{ hours}$	$\approx 8.9\text{min}$	$\approx 53\text{ hours}$	$\approx 6\text{ hours}$
Image Res.	1280×960		1280×720		1280×720	
Location	San Diego, USA		The SF Bay Area, USA		The SF Bay Area, USA	

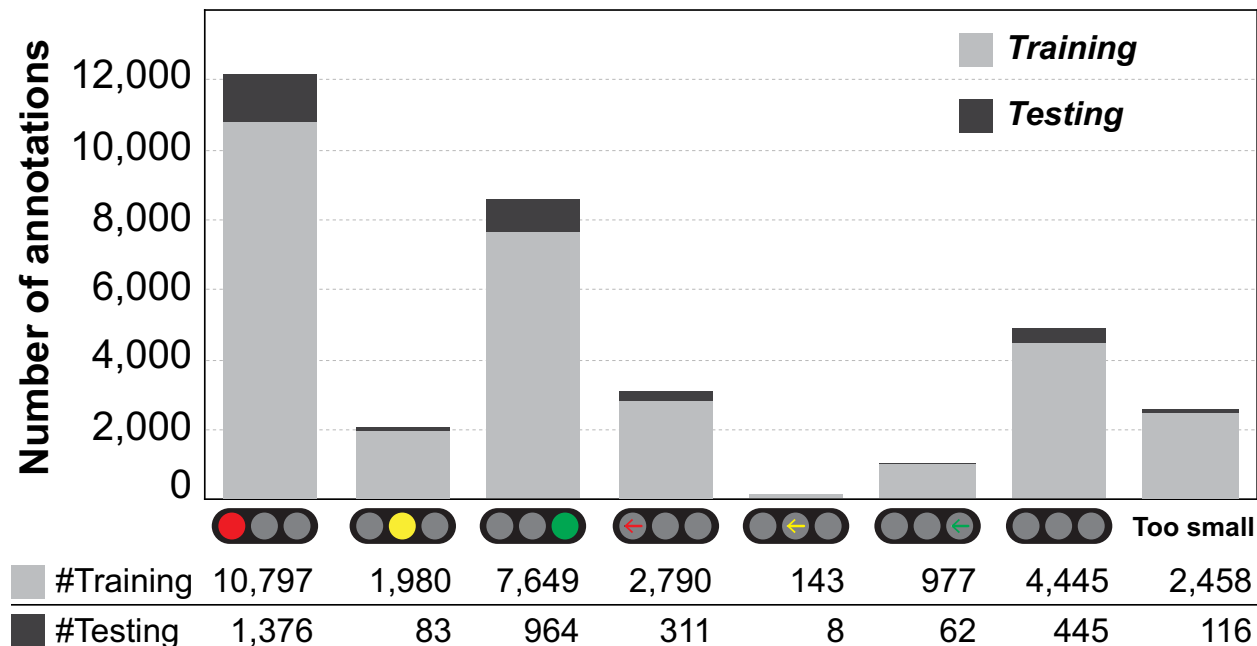


Figure 2.3: Traffic lights annotation statistics.

Recall from Section 2.2.4, we use a fine-grained detector that further examines the traffic light candidates by using an additional small neural network with high-resolution inputs. Table 2.2 shows the classification performance with and without the fine-grained detector. In the cases tested except for two classes (*i.e.* red circle and red left), using a fine-grained detector resulted in higher classification performance in F-measure values. The F-measure is 3.44 - 17.03% higher as compared to the coarse-grained detector only.

2.4.3 Real-world Experiments

To demonstrate the feasibility of applying our traffic light detector for a real self-driving car (see Figure 2.5 (A)), we utilize an instrumented vehicle equipped with the following specifications:

- (V.1) Vehicle: Hyundai Genesis G80
- (V.2) Sensors: 2×Velodyne LiDAR sensors, 4×Radar sensors, and 1×video camera (resolution: 1280x720 pixels, frame rate: 10Hz, field of view (FOV): 60 degrees).
- (V.3) PC: Intel i7 Quad-core processor, 16GB DDR3 memory, a 1TB SSD, a Titan X Pascal GPU, and Linux OS.

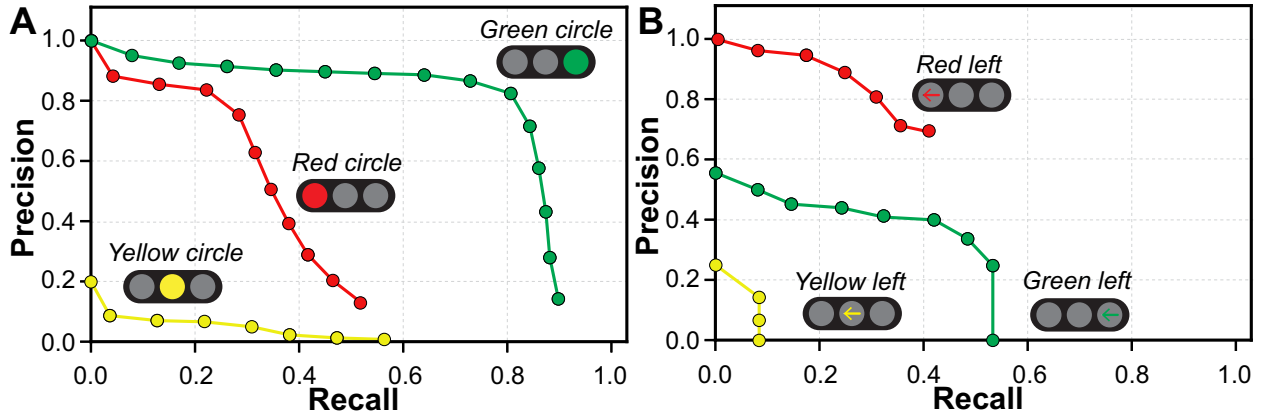


Figure 2.4: Performance evaluation of our coarse-grained detector in terms of two widely-used metrics: precision and recall. For red, green, and yellow circles, see A. For others, see B.

We use the Robot Operating System (ROS) for synchronizing the sensor data and for the message passing of perception, motion planning, and control nodes. At each timestep, the sensory data is consumed from raw sensors (camera, LiDAR, and Radar) and processed by a collection of ROS nodes that all communicate with each other. We use the PID controller for our control node, hence the final output control commands to the throttle, brake, and

Table 2.2: The effect of using fine-grained detector (see Section 2.2.4) is evaluated in terms of precision, recall, and F-measure. Scores are reported in percentage (%).

Classes	<i>without</i> Fine-grained detector			<i>with</i> Fine-grained detector		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Red circle	70.20	29.70	41.74	68.00	29.50	41.15
Yellow circle	2.20	40.00	4.17	6.00	30.90	10.05
Green circle	37.60	87.80	52.65	60.80	81.60	69.68
Red left	84.20	27.70	41.69	62.40	29.00	39.60
Yellow left	14.30	8.30	10.50	66.70	16.70	26.71
Green left	32.30	50.00	39.25	36.40	51.60	42.69

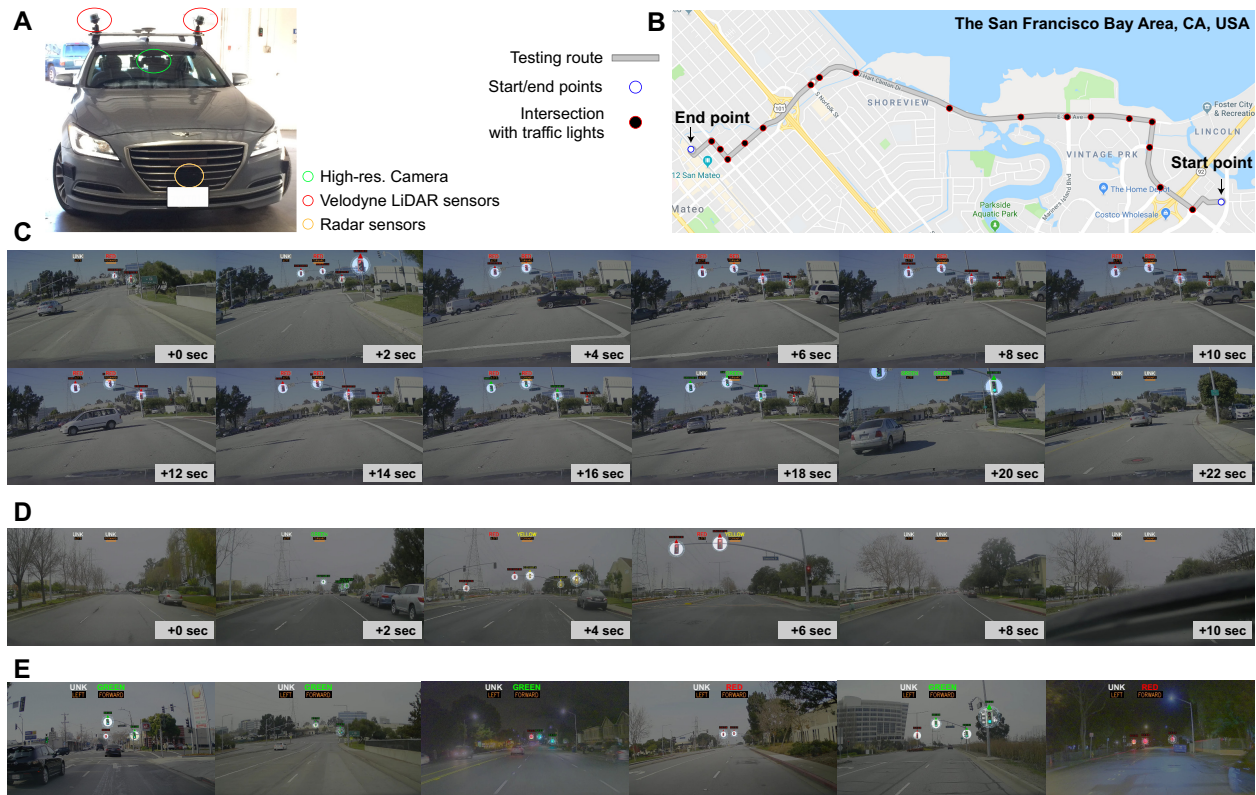


Figure 2.5: (A) Our instrumented vehicle with sensor layout for real-world evaluation of the proposed traffic light detection pipeline. (B) Our testing route (in the San Francisco Bay Area, CA, USA) for the real-world experiment. This route is over 6 kilometers including 17 intersections with traffic lights installed. *Map credit:* Google Maps. (C-D) Visualizations for traffic light detection over time. Unseen consecutive input images are sampled at every 2 seconds (see bottom-right). Our final decisions are depicted on the top of each figure, while detected traffic lights are highlighted by a white circle. (E) Additional examples of detected traffic lights during the test scenario.

steering wheel are provided through our drive-by-wire units. We depict major steps in Figure 2.2.

We test our proposed traffic light detector with an instrumented vehicle on public roads in Bay Area, California, USA. As shown in Figure 2.5 (C-E), the test runs were performed in an unseen pre-specified testing route (over 6 kilometers), and the test scenario comprised the following features: (1) The vehicle traversed 17 traffic light-controlled intersections where the vehicle will follow the rules to stop on red and go on green. (2) Different lighting conditions (day vs. night), weather (rainy vs. sunny), and different road traffic congestion levels are tested. We build a visualization to show which traffic lights are detected and its the final decision. We provide examples from our visualizer during the on-road driving test. Our real-world experiments support that the proposed traffic light detector can be successfully operated in a real-time manner.

2.5 Related Work

A number of approaches have been proposed for traffic light detection and classification for autonomous vehicles and/or for driver assistance systems to navigate in urban areas. Most of these approaches utilized a supervised learning approach with human-designated features. This literature is too wide to survey here. For a thorough review of this literature, see [34].

These approaches usually depend on strong assumptions: (1) they are based on recognizing human-designated features, which generally require demanding parameter tuning for a balanced performance. (2) Some require the detailed maps that provide prior knowledge about the specific locations of all installed traffic lights, which but demand high costs in building such a map. Furthermore, other issues may include: (i) color-tone shifting due to changes in atmospheric conditions and nearby light sources. (ii) Occlusion by other objects. (iii) High false positive rates caused by brake lights, reflections, and pedestrian crossing lights. (iv) Inconsistent traffic light lamps due to dirt, defects, over-saturation of the camera (especially during night-time).

Recent approaches suggest that deep neural networks can be successfully used for the traffic light detection task. Weber *et al.* [84] utilized a 7-layer convolutional neural network to predict the multi-class probability map followed by bounding box regression. Behrendt *et al.* [8] used the “You Only Look Once” (YOLO) network architecture to detect traffic lights, and utilized a tiny convolutional neural network to classify the categories of each detected traffic lights. They also provide a dataset, called the Bosch Small Traffic Lights Dataset, which provides approximately 5,000 images (2.8 hours of driving) and 8,334 annotations. Despite its potential, training these deep neural networks requires a large amount of annotated dataset to train a reliable detector that can address challenges in traffic light detection task. Though there exist some other open-sources of traffic light annotations, these datasets are still insufficient for training deep neural networks in terms of diversity of scenes, quality of annotations, and their limited volume. For example, the VIVA challenge dataset [62] only provides approximately 40 minutes of scenes, while the Bosch [8] Small Traffic Lights

dataset provides less than 3 hours (5,000 images). We, therefore, collect our own dataset, which provides over 60 hours (over 71,771 images) of driving images that cover diverse driving conditions (*i.e.* day vs. night and sunny vs. raining). Thus, we argue this dataset will be ideal for traffic light detection studies.

Chapter 3

Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention

3.1 Problem Statement

Self-driving vehicle control has made dramatic progress in the last several years, and many auto vendors have pledged large-scale commercialization in a 2-3 year time frame. These controllers use a variety of approaches but recent successes [9] suggest that neural networks will be widely used in self-driving vehicles. Deep neural networks have been shown to be an effective tool [9, 87] to learn vehicle controls for self-driving cars in an end-to-end manner. Despite their effectiveness as a function estimator, DNNs operate as a black-box – both network architecture and hidden layer activations may have no obvious relation to the function being estimated by the network.

- (i) **Introspective explanations:** A system is introspective through a series of understandable ways (*i.e.* Bob explains Bob’s actions).
- (ii) **Rationalizations:** We want to justify or rationalize the system through a series of logically consistent and understandable choices that can correlate model response with physical observations (*i.e.* Alice watching a video of Bob, and then asking Alice to justify Bob’s actions).

*This work has been presented at the 16th IEEE International Conference on Computer Vision (ICCV), 2017.

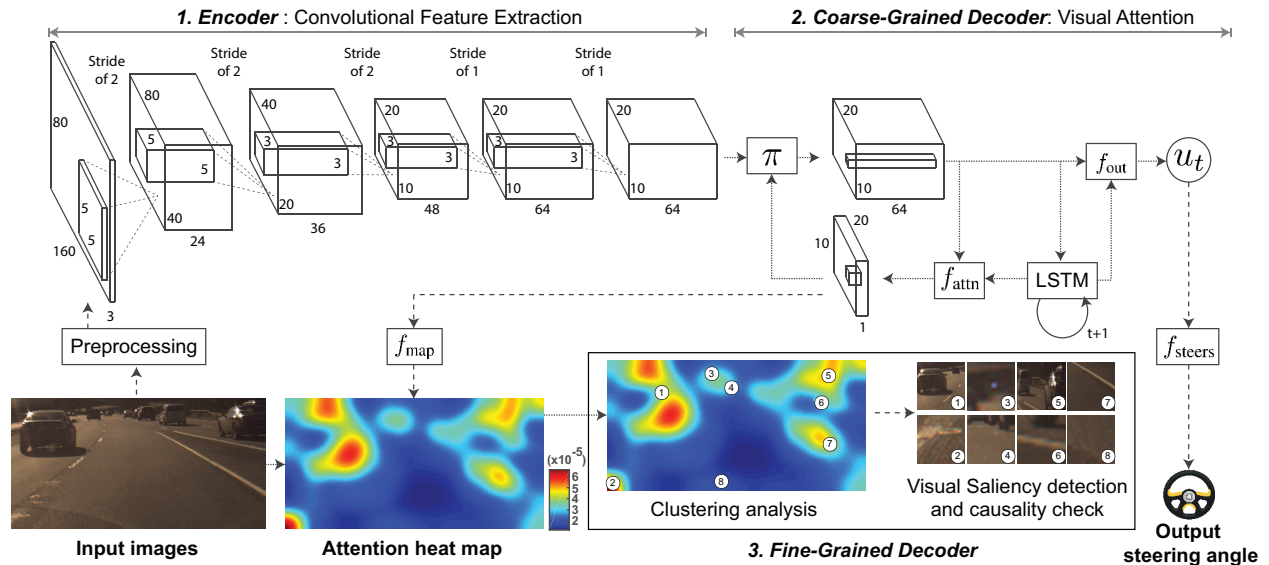


Figure 3.1: Our model predicts steering angle commands from an input raw image stream in an end-to-end manner. In addition, our model generates a heat map of attention, which can visualize where and what the model sees. To this end, we first encode images with a CNN and decode this feature into a heat map of attention, which is also used to control a vehicle. We test its causality by scrutinizing each cluster of attention blobs and produce a refined attention heat map of causal visual saliency.

To allow end-users understand what has triggered a particular behavior, hence to increase trust, these models need to be self-explanatory. There exist two main types of philosophical argument for explanations [2]: (i) Introspective explanations and (ii) rationalizations.

One way of achieving introspection is via visual attention mechanisms [88]. Visual attention filters out non-salient image regions, hence the model visually fixates on important image content that is relevant to the decision. These networks provide spatial attention maps – areas of the image that the network attends to – that can be displayed in a way that is easy for users to interpret. They provide their attention maps instantly on images that are input to the network, and in this case on the stream of images from automobile video. Providing visual attention to the user as a justification of a decision increases trust. As we show from our examples later, visual attention maps lie over image areas that have an intuitive influence on the vehicle’s control signal. Further, we show that state-of-the-art driving models can be made interpretable without sacrificing accuracy, that attention models provide more robust image annotation, and causal analysis further improves explanation saliency.

But attention maps are only part of the story. Attention is a mechanism for filtering out non-salient image content. But attention networks need to find all *potentially* salient

image areas and pass them to the main recognition network (a CNN here) for a final verdict. For instance, the attention network will attend to trees and bushes in areas of an image where road signs commonly occur. Just as a human will use peripheral vision to determine that “there is something there”, and then visually fixate on the item to determine what it actually is. We therefore post-process the attention network’s output, clustering it into attention “blobs” and then mask (set the attention weights to zero) each blob to determine the effect on the end-to-end network output. Blobs that have a causal effect on network output are retained while those that do not are removed from the visual map presented to the user.

Figure 3.1 shows an overview of our model. Our approach can be divided into three steps: (1) Encoder: convolutional feature extraction, (2) Coarse-grained decoder by visual attention mechanism, and (3) Fine-grained decoder: causal visual saliency detection and refinement of attention map. Our contributions are as follows:

- We show that visual attention heat maps are suitable “explanations” for the behavior of a deep neural vehicle controller, and do not degrade control accuracy.
- We show that attention maps comprise “blobs” that can be segmented and filtered to produce simpler and more accurate maps of visual saliency.
- We demonstrate the effectiveness of using our model with three large real-world driving datasets that contain over 1,200,000 video frames (*approx.* 16 hours).
- We illustrate typical spurious attention sources in driving video and quantify the reduction in explanation complexity from causal filtering.

3.2 Interpretable Driving Model

As we depicted in Figure 3.1, our model predicts continuous steering angle commands from input raw images end-to-end. Our model can be divided into three steps: (1) Encoder: convolutional feature extraction (Section ??) (2) Coarse-grained decoder by visual attention mechanism (Section 3.2.3), and (3) Fine-grained decoder: causal visual saliency detection and refinement of attention maps (Section 3.2.4).

3.2.1 Preprocessing

Our model predicts continuous steering angle commands from input raw pixels in an end-to-end manner. As discussed by Bojarski *et al.* [9], our model predicts the inverse turning radius $\hat{u}_t (= r_t^{-1}$, where r_t is the turning radius) at every timestep t instead of steering angle commands, which depends on the vehicle’s steering geometry and also result in numerical instability when predicting near zero steering angle commands. The relationship between

the inverse turning radius u_t and the steering angle command θ_t can be approximated by Ackermann steering geometry [64] as follows:

$$\theta_t = f_{steers}(u_t) = u_t d_w K_s (1 + K_{slip} v_t^2) \quad (3.1)$$

where θ_t in degrees and v_t (m/s) is a steering angle and a velocity at time t , respectively. K_s , K_{slip} , and d_w are vehicle-specific parameters. K_s is a steering ratio between the turn of the steering and the turn of the wheels. K_{slip} represents the relative motion between a wheel and the surface of road. d_w is the length between the front and rear wheels. Our model therefore needs two measurements for training: timestamped vehicle’s speed and steering angle commands.

To reduce computational cost, each raw input image is down-sampled and resized to $80 \times 160 \times 3$ with nearest-neighbor scaling algorithm. For images with different raw aspect ratios, we cropped the height to match the ratio before down-sampling. A common practice in image classification is to subtract the mean RGB value computed on the training set from each pixel [simonyan2014very](#). This is effective to achieve zero-centered inputs which are originally in different scales. Driving datasets, however, do not show that various scales. For instance, the camera gains are (automatically or in advance) calibrated to capture such high-quality images in a certain dynamic range. In our experiment, we could not obtain significant improvement by the use of mean subtraction. Instead, we change the range of pixel intensity values and convert to HSV colorspace, which is commonly used for its robustness in problems where color description plays an integral role.

We utilize a single exponential smoothing method [33] to reduce the effect of human factors-related performance variation and the effect of measurement noise. Formally, given a smoothing factor $0 \leq \alpha_s \leq 1$, the simple exponential smoothing method is defined as follows:

$$\begin{pmatrix} \hat{\theta}_t \\ \hat{v}_t \end{pmatrix} = \alpha_s \begin{pmatrix} \theta_t \\ v_t \end{pmatrix} + (1 - \alpha_s) \begin{pmatrix} \hat{\theta}_{t-1} \\ \hat{v}_{t-1} \end{pmatrix} \quad (3.2)$$

where $\hat{\theta}_t$ and \hat{v}_t are the smoothed time-series of θ_t and v_t , respectively. Note that they are same as the original time-series when $\alpha_s = 1$, while values of α_s closer to zero have a greater smoothing effect and are less responsive to recent changes. The effect of applying smoothing methods is summarized in Section 3.3.4.

3.2.2 Encoder: Convolutional Feature Extraction

We use a convolutional neural network to extract a set of encoded visual feature vector, which we refer to as a convolutional feature cube x_t . Each feature vectors may contain high-level object descriptions that allow the attention model to selectively pay attention to certain parts of an input image by choosing a subset of feature vectors.

As depicted in Figure 3.1, we use a 5-layered convolution network that is utilized by [Bojarski et al.](#) [9] to learn a model for self-driving cars. As discussed by [Lee et al.](#) [45], we

omit max-pooling layers to prevent spatial locational information loss as the strongest activation propagates through the model. We collect a three-dimensional convolutional feature cube x_t from the last layer by pushing the preprocessed image through the model, and the output feature cube will be used as an input of the LSTM layers, which we will explain in Section 3.2.3. Using this convolutional feature cube from the last layer has advantages in generating high-level object descriptions, thus increasing interpretability and reducing computational burdens for a real-time system.

Formally, a convolutional feature cube of size $W \times H \times D$ is created at each timestep t from the last convolutional layer. We then collect x_t , a set of $L = W \times H$ vectors, each of which is a D -dimensional feature slice for different spatial parts of the given input.

$$x_t = \{x_{t,1}, x_{t,2}, \dots, x_{t,L}\} \quad (3.3)$$

where $x_{t,i} \in R^D$ for $i \in \{1, 2, \dots, L\}$. This allows us to focus selectively on different spatial parts of the given image by choosing a subset of these L feature vectors.

3.2.3 Coarse-Grained Decoder: Visual Attention

The goal of soft deterministic attention mechanism $\pi(\{x_{t,1}, x_{t,2}, \dots, x_{t,L}\})$ is to search for a good context vector y_t , which is defined as a combination of convolutional feature vectors $x_{t,i}$, while producing better prediction accuracy. We utilize a deterministic soft attention mechanism that is trainable by standard back-propagation methods, which thus has advantages over a hard stochastic attention mechanism that requires reinforcement learning. Our model feeds α weighted context y_t to the system as discuss by several works [71, 88]:

$$\begin{aligned} y_t &= f_{\text{flatten}}(\pi(\{\alpha_{t,i}, \{x_{t,i}\}\})) \\ &= f_{\text{flatten}}(\{\alpha_{t,i}x_{t,i}\}) \end{aligned} \quad (3.4)$$

where $i = \{1, 2, \dots, L\}$. $\alpha_{t,i}$ is a scalar attention weight value associated with a certain grid of input image in such that $\sum_i \alpha_{t,i} = 1$. These attention weights can be interpreted as the probability over L convolutional feature vectors that the location i is the important part to produce better estimation accuracy. f_{flatten} is a flattening function. y_t is thus $D \times L$ -dimensional vector that contains convolutional feature vectors weighted by attention weights. Note that, our attention mechanism $\pi(\{\alpha_{t,i}, \{x_{t,i}\}\})$ is different from the previous works [71, 88], which use the α weighted average context $y_t = \sum_{i=1}^L \alpha_{t,i}x_{t,i}$. We observed that this change significantly improves overall prediction accuracy. The performance comparison is explained in Section 3.3.5.

3.2.3.1 Long Short-term Memory (LSTM).

As we summarize in Figure 3.1, we use a long short-term memory (LSTM) network [29] that predicts the inverse turning radius \hat{u}_t and generates attention weights $\{\alpha_{t,i}\}$ at each timestep

t conditioned on the previous hidden state h_{t-1} and a current convolutional feature cube x_t . The LSTM is defined as follows:

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \text{tanh} \end{pmatrix} A \begin{pmatrix} h_{t-1} \\ y_t \end{pmatrix} \quad (3.5)$$

where i_t, f_t, o_t , and $c_t \in \mathbb{R}^M$ are the M-dimensional input, forget, output, memory state of the LSTM at time t , respectively. Internal states of the LSTM are computed conditioned on the hidden state $h_t \in \mathbb{R}^M$ and an α -weighted context vector $y_t \in \mathbb{R}^d$. We use an affine transformation $A : \mathbb{R}^{d+M} \rightarrow \mathbb{R}^{4M}$. The logistic sigmoid activation function and the hyperbolic tangent activation function are represented as sigm and tanh , respectively. The hidden state h_t and the cell state c_t of the LSTM are defined as:

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \text{tanh}(c_t) \end{aligned} \quad (3.6)$$

where \odot is element-wise multiplication.

3.2.3.2 Attention.

We use an additional hidden layer, denoted by $f_{\text{attn}}(x_{t,i}, h_{t-1})$, which is conditioned on the previous LSTM state h_{t-1} , and the current feature vectors $x_{t,i}$. Then, we use multinomial logistic regression (*i.e.* softmax regression) function to obtain the attention weight $\{\alpha_{t,i}\}$ as follows:

$$f_{\text{attn}}(x_{t,i}, h_{t-1}) = W_{\mathbf{a}}(W_{\mathbf{x}}x_{t,i} + W_{\mathbf{h}}h_{t-1} + b_{\mathbf{a}}) \quad (3.7)$$

where $W_{\mathbf{a}} \in \mathbb{R}^d$, $W_{\mathbf{x}} \in \mathbb{R}^{d \times d}$, and $W_{\mathbf{h}} \in \mathbb{R}^{d \times M}$, which are learned parameters. The attention weight $\alpha_{t,i}$ for each spatial location i is then computed by multinomial logistic regression (*i.e.* softmax regression) function as follows:

$$\alpha_{t,i} = \frac{\exp(f_{\text{attn}}(x_{t,i}, h_{t-1}))}{\sum_{j=1}^l \exp(f_{\text{attn}}(x_{t,j}, h_{t-1}))} \quad (3.8)$$

3.2.3.3 Initialization.

To initialize memory state c_t and hidden state h_t of the LSTM, we use average of the convolutional feature slices $x_{0,i} \in \mathbb{R}^d$ for $i \in \{0, 1, \dots, l\}$ and feed through two additional hidden layers: $f_{\text{init},c}$ and $f_{\text{init},h}$.

$$c_0 = f_{\text{init},c} \left(\frac{1}{l} \sum_{i=1}^l x_{0,i} \right), \quad h_0 = f_{\text{init},h} \left(\frac{1}{l} \sum_{i=1}^l x_{0,i} \right) \quad (3.9)$$

3.2.3.4 Output.

The output of the vehicle controller is vehicle’s inverse turning radius \hat{u}_t . We use additional hidden layer, denoted by $f_{\text{out}}(y_t, h_t)$, which are conditioned on the current hidden state h_t and the spatially-attended context y_t .

$$\begin{aligned}\hat{u}_t &= f_{\text{out}}(y_t, h_t) \\ &= W_{\text{U}}(W_{\text{Y}}y_t + W_{\text{h}}h_t)\end{aligned}\tag{3.10}$$

where $W_{\text{U}} \in \mathbb{R}^d$, $W_{\text{Y}} \in \mathbb{R}^{d \times d}$, $W_{\text{h}} \in \mathbb{R}^{d \times M}$, which are learned parameters.

3.2.3.5 Loss Function and Regularization.

As discussed by [88], doubly stochastic regularization can encourage the attention model to at different parts of the image. At each timestep t , our attention model predicts a scalar $\beta_t = \text{sigm}(f_{\beta}(h_{t-1}))$ with an additional hidden layer f_{β} conditioned on the previous hidden state h_{t-1} such that

$$y_t = \text{sigm}(f_{\beta}(h_{t-1})) f_{\text{flatten}}(\{\alpha_{t,i}x_{t,i}\})\tag{3.11}$$

Concretely, we use the following penalized loss function \mathcal{L}_1 :

$$\mathcal{L}_1(u_t, \hat{u}_t) = \sum_{t=1}^T |u_t - \hat{u}_t| + \lambda \sum_{i=1}^L \left(1 - \sum_{t=1}^T \alpha_{t,i}\right)\tag{3.12}$$

where T is the length of time steps, and λ is a penalty coefficient that encourages the attention model to see different parts of the image at each time frame. Section 3.3.3 describes the effect of using regularization.

3.2.4 Fine-Grained Decoder: Causality Test

The last step of our pipeline is a fine-grained decoder, in which we refine a map of attention and detect local visual saliencies. Though an attention map from our coarse-grained decoder provides probability of importance over a 2D image space, our model needs to determine specific regions that cause a causal effect on prediction performance. To this end, we assess a decrease in performance when a local visual saliency on an input raw image is masked out.

We first collect a consecutive set of attention weights $\{\alpha_{t,i}\}$ and input raw images $\{\mathcal{I}_t\}$ for a user-specified T timesteps. We then create a map of attention, which we refer \mathcal{M}_t as defined: $\mathcal{M}_t = f_{\text{map}}(\{\alpha_{t,i}\})$. Our 5-layer convolutional neural network uses a stack of 5×5 and 3×3 filters without any pooling layer, and therefore the input image of size 80×160 is processed to produce the output feature cube of size $10 \times 20 \times 64$, while preserving its aspect ratio. Thus, we use $f_{\text{map}}(\{\alpha_{t,i}\})$ as up-sampling function by the factor of eight followed by Gaussian filtering [12] as discussed in [88] (see Figure 3.2 (A,B)).

To extract a local visual saliency, we first randomly sample 2D N particles with replacement over an input raw image conditioned on the attention map \mathcal{M}_t . Note that, we also use

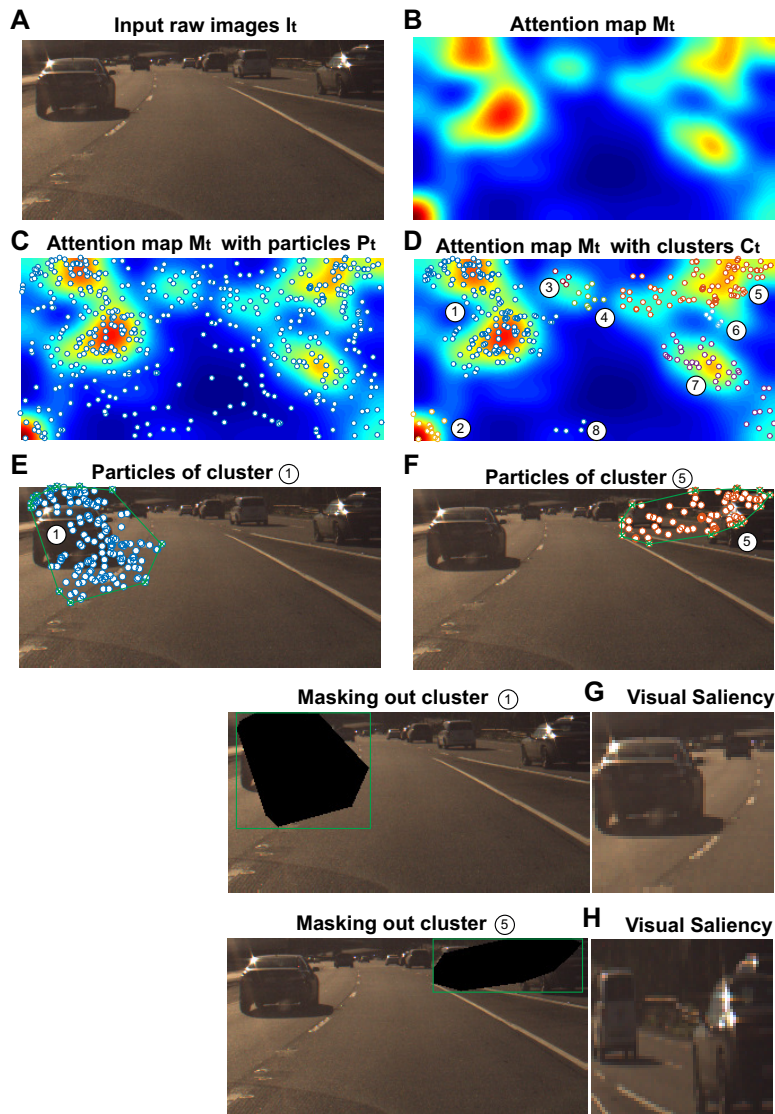


Figure 3.2: Overview of our fine-grained decoder. Given an input raw pixels \mathcal{I}_t (A), we compute an attention map \mathcal{M}_t with a function f_{map} (B). (C) We randomly sample 3D $N = 500$ particles over the attention map, and (D) we apply a density-based clustering algorithm (DBSCAN [21]) to find a local visual saliency by grouping particles into clusters. (E, F) For each cluster $c \in \mathcal{C}$, we compute a convex hull $\mathcal{H}(c)$ to define its region, and mask out the visual saliency to see causal effects on prediction accuracy (see E, F for clusters 1 and 5, respectively). (G, H) Warped visual saliencies for clusters 1 and 5, respectively.

<p>Algorithm 1: Fine-grained Decoder: Causality Check</p> <p>Data: A consecutive set of $\{u_t\}$ and images $\{I_t\}$ and Result: A set of visual saliencies S</p> <p>Get a set of $\{\alpha_{t,i}\}$ and prediction errors $\{\varepsilon_t\}$ by running Encoder and Decoder for all images $\{I_t\}$; $P \leftarrow \phi, S \leftarrow \phi$; for $t = 0$ to $T-1$ do Get a 2D attention map $M_t = f_{\text{map}}(\{\alpha_{t,i}\})$; Get a set P_t of randomly sampled 2D N points conditioned on M_t; Aggregate datasets: $P \leftarrow P \cup \{P_t, t\}$; end Run clustering algorithm on P and get clusters $\{C_t\}$; for $t = 0$ to $T-1$ do for each cluster $c \in C_t$ do Get a convex hull $H(c)$; Masking out pixels on $H(c)$ from I_t; Get a new prediction error $\hat{\varepsilon}_t$; if $\hat{\varepsilon}_t - \varepsilon_t > \delta$ then Aggregate saliency $S \leftarrow S \cup H(c)$; end end end</p>
--

time-axis as the third dimension to consider temporal features of visual saliencies. We thus store spatio-temporal 3D particles $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_t, t\}$ (see Figure 3.2 (C)).

We then apply a clustering algorithm to find a local visual saliency by grouping 3D particles \mathcal{P} into clusters $\{C\}$ (see Figure 3.2 (D)). In our experiment, we use DBSCAN [21], a density-based clustering algorithm that has advantages to deal with a noisy dataset because they group particles together that are closely packed, while marking particles as outliers that lie alone in low-density regions. For points of each cluster c and each time frame t , we compute a convex hull $\mathcal{H}(c)$ to find a local region of each visual saliency detected (see Figure 3.2 (E, F)).

For points of each cluster c and each time frame t , we iteratively measure a decrease of prediction performance with an input image which we mask out a local visual saliency. We compute a convex hull $\mathcal{H}(c)$ to find a local, and mask out each visual saliency by assigning zero values for all pixels lying inside each convex hull. Each causal visual saliency is generated by warping into a fixed spatial resolution ($=64 \times 64$) as shown in Figure 3.2 (G, H). Algorithm 1 explains a pseudo-code for this step.

Feature-level Masking Approach. Along with devising the fine-grained decoder, we may consider using feature-level masking approach. Masking out convolutional features of at-

tended region can provide a computationally efficient way by removing multiple forward passes of the convolutional network. This approach, however, may suffer from low deconvolutional spatial resolution, which makes challenge to view as a unit apart and divide the whole attention map into distinct attended objects, such as cars or lane markings.

3.3 Experiments

3.3.1 Datasets

As explained in Table 3.1, we obtain two large-scale datasets that contain over 1,200,000 frames (≈ 16 hours) collected from Comma.ai [19], Udacity [78], and Hyundai Center of Excellence in Integrated Vehicle Safety Systems and Control (HCE) under a research contract. These three datasets acquired contain video clips captured by a single front-view camera mounted behind the windshield of the vehicle. Alongside the video data, a set of time-stamped sensor measurement is contained, such as vehicle’s velocity, acceleration, steering angle, GPS location and gyroscope angles. Thus, these datasets are ideal for self-driving studies. Note that, for sensor logs unsynced with the time-stamps of video data, we use the estimates of the interpolated measurements. Videos are mostly captured during highway driving in clear weather on daytime, and there included driving on other road types, such as residential roads (with and without lane markings), and contains the whole driver’s activities such as staying in a lane and switching lanes. Note also that, we exclude frames when the vehicle stops which happens when $\hat{v}_t < 1$ m/s.

3.3.2 Training and Evaluation Details

To obtain a convolutional feature cube x_t , we train the 5-layer CNNs explained in Section 3.2.2 by using additional 5-layer fully connected layers (*i.e.* # hidden variables: 1164, 100, 50, and 10, respectively), of which output predicts the measured inverse turning radius u_t . Incidentally, instead of using addition fully-connected layers, we could also obtain a convolutional feature cube x_t by training from scratch with the whole network. In our experiment, we obtain the $10 \times 20 \times 64$ -dimensional convolutional feature cube, which is then flattened to 200×64 and is fed through the coarse-grained decoder. Other recent types of more recent expressive networks may give a performance boost over our CNN configuration. However, exploration of other convolutional architectures would be out of our scope.

We experiment with various numbers of LSTM layers (1 to 5) of the soft deterministic visual attention model but did not observe any significant improvements in model performance. Unless otherwise stated, we use a single LSTM layer in this experiment. For training, we use Adam optimization algorithm [40] and also use dropout [73] of 0.5 at hidden state connections and Xavier initialization [23]. We randomly sample a mini-batch of size 128, each of batch contains a set Consecutive frames of length $T = 20$. Our model took less than

	Dataset		
	Comma.ai [19]	HCE	Udacity [78]
# frame	522,434	80,180	650,690
FPS	20Hz	20Hz/30Hz	20Hz
Hours	≈ 7 hrs	≈ 1 hr	≈ 8 hrs
Condition	Highway/Urban	Highway	Urban
Location	CA, USA	CA, USA	CA, USA
Lighting	Day/Night	Day	Day

Table 3.1: Dataset details. Over 16 hours ($>1,200,000$ video frames) of driving dataset that contains a front-view video frames and corresponding time-stamped measurements of vehicle dynamics. The data is collected from two public data sources, Comma.ai [19] and Udacity [78], and Hyundai Center of Excellence in Vehicle Dynamic Systems and Control (HCE).

24 hours to train on a single NVIDIA Titan X Pascal GPU. Our implementation is based on Tensorflow [1] and code will be publicly available upon publication.

Two datasets (Comma.ai [19] and HCE) we used were available with images captured by a single front-view camera. This makes it hard to use the data augmentation technique proposed by Bojarski *et al.* [9], which generated images with artificial shifts and rotations by using two additional off-center images (left-view and right-view) captured by the same vehicle. Data augmentation may give a performance boost, but we report performance without data augmentation.

3.3.3 Effect of Choosing Penalty Coefficient λ

Our model provides a better way to understand the rationale of the model’s decision by visualizing where and what the model sees to control a vehicle. Figure 3.3 shows a consecutive input raw images (with sampling period of 5 seconds) and their corresponding attention maps (*i.e.* $\mathcal{M}_t = f_{\text{map}}(\{\alpha_{t,i}\})$). We also experiment with three different penalty coefficients $\lambda \in \{0, 10, 20\}$, where the model is encouraged to pay attention to wider parts of the image (see differences between the bottom 3 rows in Figure 3.3) as we have larger λ . For better visualization, an attention map is overlaid by an input raw image and color-coded; for example, red parts represent where the model pays attention. For quantitative analysis, prediction performance in terms of mean absolute error (MAE) is explained on the bottom of each figure. We observe that our model is indeed able to pay attention on road elements, such as lane markings, guardrails, and vehicles ahead, which is essential for human to drive.

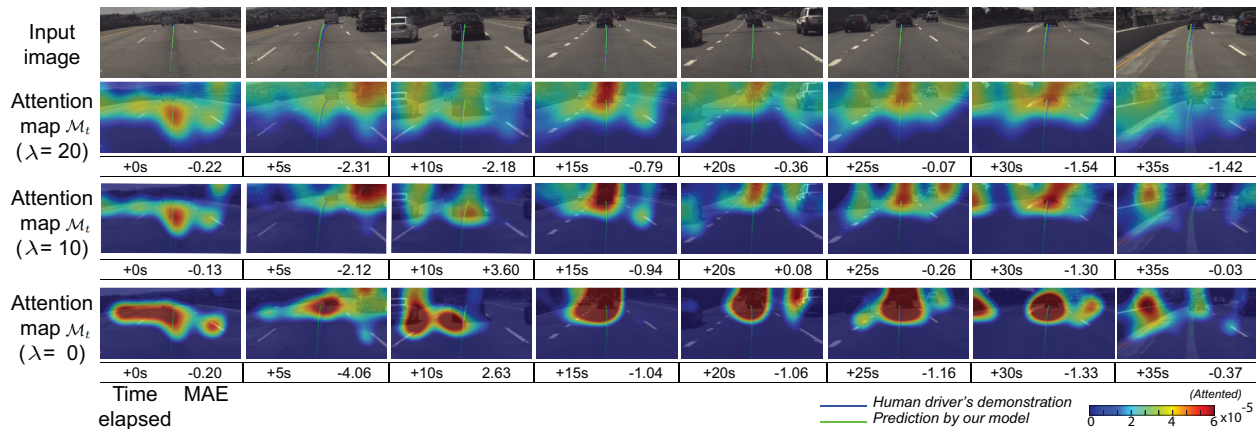


Figure 3.3: Attention maps over time. Unseen consecutive input image frames are sampled at every 5 seconds (see from left to right). (Top) Input raw images with human driver’s demonstrated curvature of path (blue line) and predicted curvature of path (green line). (From the bottom) We illustrate attention maps with three different regularization penalty coefficients $\lambda \in \{0, 10, 20\}$. Each attention map is overlaid by an input raw image and color-coded. Red parts indicate where the model pays attention. *Data*: Comma.ai [19]

3.3.4 Effect of Varying Smoothing Factors

Recall from Section 3.2.1 that the single exponential smoothing method [33] is used to reduce the effect of human factors variation and the effect of measurement noise for two sensor inputs: steering angle and velocity. A robust model for autonomous vehicles would yield consistent performance, even when some measurements are noisy. Figure 3.4 shows the prediction performance in terms of mean absolute error (MAE) on a comma.ai testing data set. Various smoothing factors $\alpha_s \in \{0.01, 0.05, 0.1, 0.3, 0.5, 1.0\}$ are used to assess the effect of using smoothing methods. With setting $\alpha_s=0.05$, our model for the task of steering estimation performs the best. Unless otherwise stated, we will use α_s as 0.05.

3.3.5 Quantitative Analysis

In Table 3.2, we compare the prediction performance with alternatives in terms of MAE. We implement alternatives that include the work by Bojarski *et al.* [9], which used an identical base CNN and a fully-connected network (FCN) without attention. To see the contribution of LSTMs, we also test a CNN and LSTM, which is identical to ours but does not use the attention mechanism. We also compare average pooled soft attention method [71]. Since this model focuses on a classification problem, we modified the output layer as ours. We use a single layer LSTM with $\lambda = 0$. For our model, we test with three different values of penalty coefficients $\lambda \in \{0, 10, 20\}$.

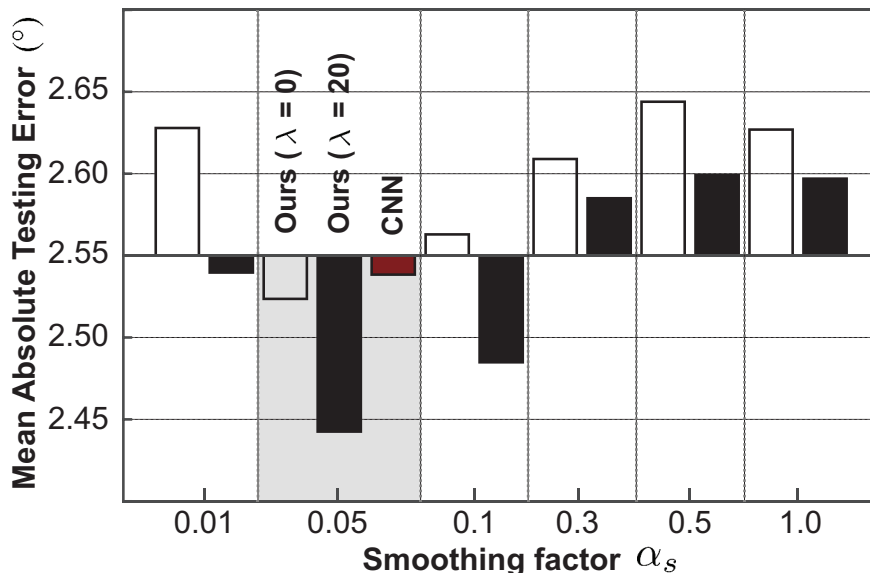


Figure 3.4: Effect of applying a single exponential smoothing method over various smoothing factors from 0.1 to 1.0. We use two different penalty coefficients $\lambda \in \{0, 20\}$. With setting $\alpha_s = 0.05$, our model performs the best. *Data*: Comma.ai [19]

Our model shows competitive prediction performance than alternatives. Our model shows 1.18–4.15 in terms of MAE on testing dataset. This confirms that incorporation of attention does not degrade control accuracy. The average run-time for our model and alternatives took less than a day to train each dataset.

3.3.6 Effect of Causal Visual Saliencies

Recall from Section 3.2.4, we post-process the attention network’s output by clustering it into attention blobs and filtering if they have an causal effect on network output. Figure 3.5 (A) shows typical examples of an input raw image, an attention networks’s output with spurious attention sources, and our refined attention heat map. We observe our model can produce a simpler and more accurate map of visual saliency by filtering out spurious attention blobs. In our experiment, 62% and 58% out of all attention blobs are indeed spurious attention sources on Comma.ai [19] and HCE datasets (see Figure 3.5 (B)). We provide additional example sets in Figure 3.6.

Dataset	Model	MAE in degree [SD]	
		Training	Testing
Comma.ai [19]	CNN+FCN [9]	.421 [0.82]	2.54 [3.19]
	CNN+LSTM	.488 [1.29]	2.58 [3.44]
	Attention ($\lambda=0$)	.497 [1.32]	2.52 [3.25]
	Attention ($\lambda=10$)	.464 [1.29]	2.56 [3.51]
	Attention ($\lambda=20$)	.463 [1.24]	2.44 [3.20]
HCE	CNN+FCN [9]	.246 [.400]	1.27 [1.57]
	CNN+LSTM	.568 [.977]	1.57 [2.27]
	Attention ($\lambda=0$)	.334 [.766]	1.18 [1.66]
	Attention ($\lambda=10$)	.358 [.728]	1.25 [1.79]
	Attention ($\lambda=20$)	.373 [.724]	1.20 [1.66]
Udacity [78]	CNN+FCN [9]	.457 [.870]	4.12 [4.83]
	CNN+LSTM	.481 [1.24]	4.15 [4.93]
	Attention ($\lambda=0$)	.491 [1.20]	4.15 [4.93]
	Attention ($\lambda=10$)	.489 [1.19]	4.17 [4.96]
	Attention ($\lambda=20$)	.489 [1.26]	4.19 [4.93]

Table 3.2: Control performance comparison in terms of mean absolute error (MAE) in degree and its standard deviation. Control accuracy is not degraded by incorporation of attention compared to an identical base CNN without attention. *Abbreviation:* SD (standard deviation)

3.4 Related Work

3.4.1 End-to-End Learning for Self-driving Cars

Self-driving vehicle control has made notable progress in the last several years. These controllers use a variety of approaches, which can mainly be divided in the following two types: (1) a mediated perception-based approach and (2) an end-to-end learning approach. The mediated perception-based approach depends on recognizing human-designated features, such as lane markings, pedestrians, or cars. These approaches mainly use a controller with if-then-else rules, which generally require demanding parameter tuning for achieving a balanced performance. Notable examples may include Urmson *et al.* [79], Buehler *et al.* [11], and Levinson *et al.* [46].

ALVINN (Autonomous Land Vehicle In a Neural Network) [63] was the first attempt to use neural network for directly mapping images to navigate the direction of the vehicle. Recent success [9] suggests that neural networks can be successfully applied to self-driving

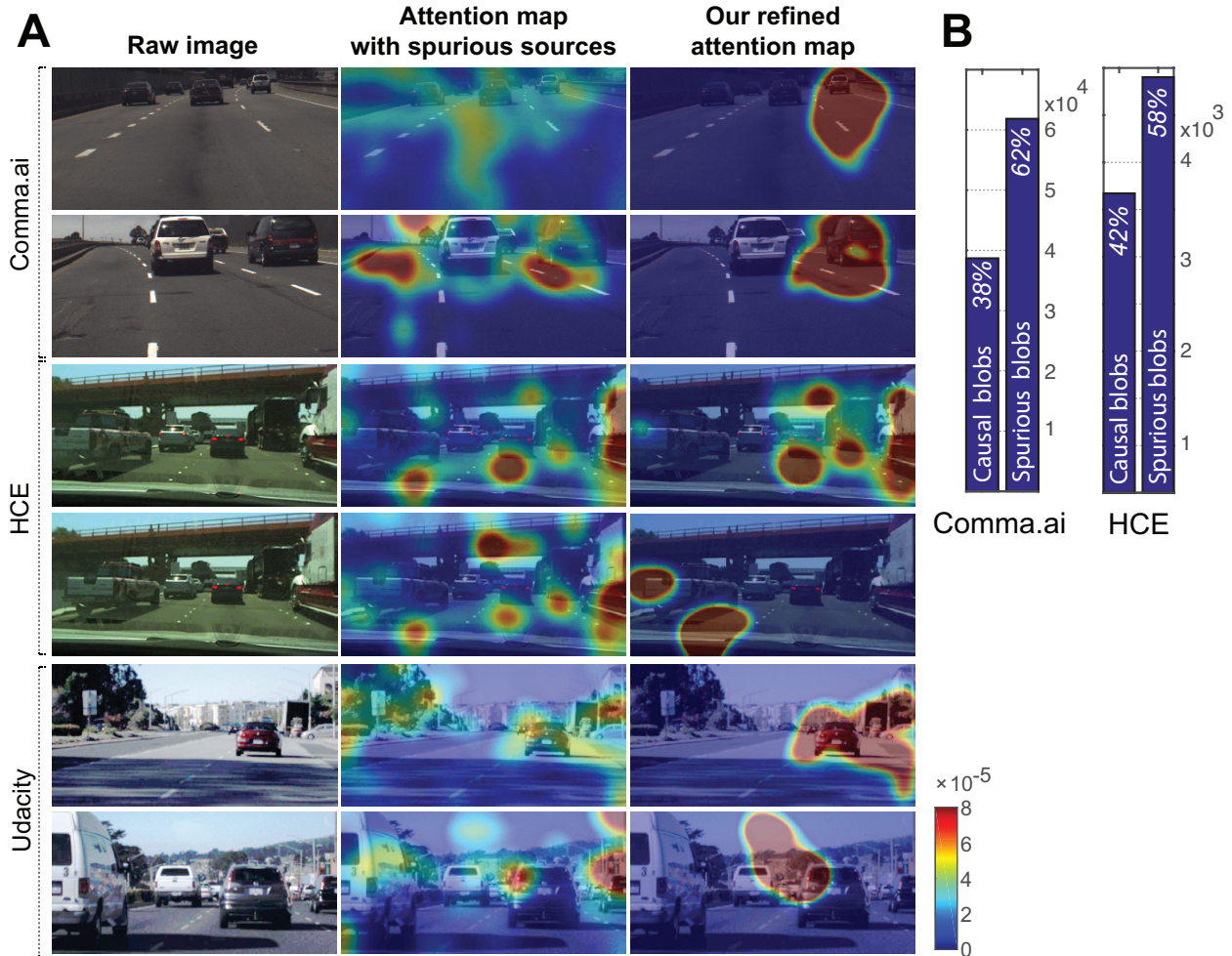


Figure 3.5: (A) We illustrate examples of (left) raw input images, their (middle) visual attention heat maps with spurious attention sources, and (right) our attention heat maps by filtering out spurious blobs to produce simpler and more accurate attention maps. (B) To measure how much the causal filtering is simplifying attention clusters, we quantify the number of attention blobs before and after causal filtering.

vehicle control in an end-to-end manner. Most of these approaches use a behavioral cloning model to learn a vehicle controller by supervised regression to demonstrations by human drivers. The training data comprise a stream of dash-cam images from one or more vehicle cameras, and the control outputs (*i.e.* steering, acceleration, and braking) from the driver.

Bojarski *et al.* [9] used a deep neural network to directly map a stream of front-view dashcam images to steering controls. Xu *et al.* [87] collected a large crowd-sourced driving dataset, and predicted a sequence of discretized vehicle’s future ego-motion (*i.e.* go straight,

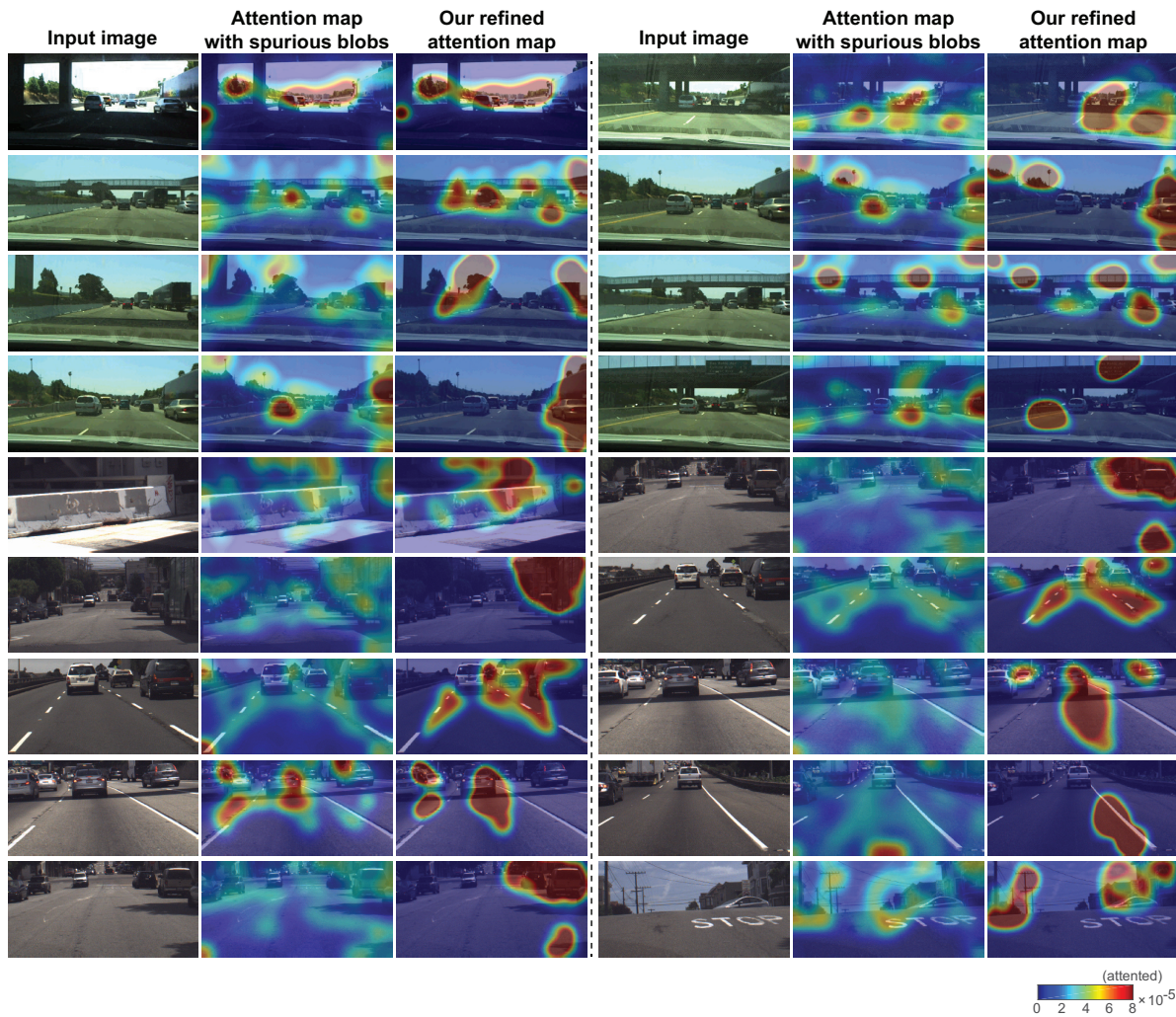


Figure 3.6: We illustrate additional examples of (left) raw input images, their (middle) visual attention heat maps with spurious attention sources, and (right) our attention heat maps by filtering out spurious blobs to produce simpler and more accurate attention maps.

stop, left-turn, and right turn) given a series of dashcam images and prior vehicle states, *i.e.* speed. Fernando *et al.* [22] presented a driving model that uses images and the steering wheel trajectory so that the model gains a long-term planning capacity via neural memory networks. Similarly, Chi *et al.* [17] proposed a Conv-LSTM framework to efficiently utilize the spatio-temporal information to model a stateful process. These models show good performance but their behavior is opaque and uninterpretable.

Chen *et al.* [13] explored an intermediate approach by defining human interpretable features (*i.e.* the curvature of lane, distances to neighboring lane markings, and distance from

the front-located vehicle) and training a CNN to predict these features. A simple vehicle controller is then used to map these features to steering angle commands. They also generated deconvolution maps to show image areas that affected network output. However, there were several difficulties with that work: (i) use of the intermediate layer may cause significant degradation of control accuracy and (ii) the intermediate feature descriptors provide a limited and ad-hoc vocabulary for explanations.

3.4.2 Visual Explanation

In a landmark work, Zeiler and Fergus [93] used “deconvolution” to visualize layer activations of convolutional networks. LeCun *et al.* [44] provides textual explanations of images as automatically-generated captions. Building on this work, Bojarski *et al.* [10] developed a richer notion of “contribution” of a pixel to the output. However a difficulty with deconvolution-style approaches is the lack of formal measures of how the network output is affected by spatially-extended features (rather than pixels). Attention-based approaches like ours directly extract areas of the image that *did not affect* network output (because they were masked out by the attention model), and causal filtering further removes spurious image areas. Hendricks *et al.* [27] trains a deep network to generate species specific explanation without explicitly identifying semantic features. Also, Justin Johnson [36] proposes DenseCap which uses fully convolutional localization networks for dense captioning, their paper achieves both localizing objects and describing salient regions in images using natural language. In reinforcement learning, Zrihem *et al.* [92] proposes a visualization method to interpret the agent’s action by describing Markov Decision Process model as a directed graph on a t-SNE map.

Chapter 4

Textual Explanations for Self-driving Vehicles

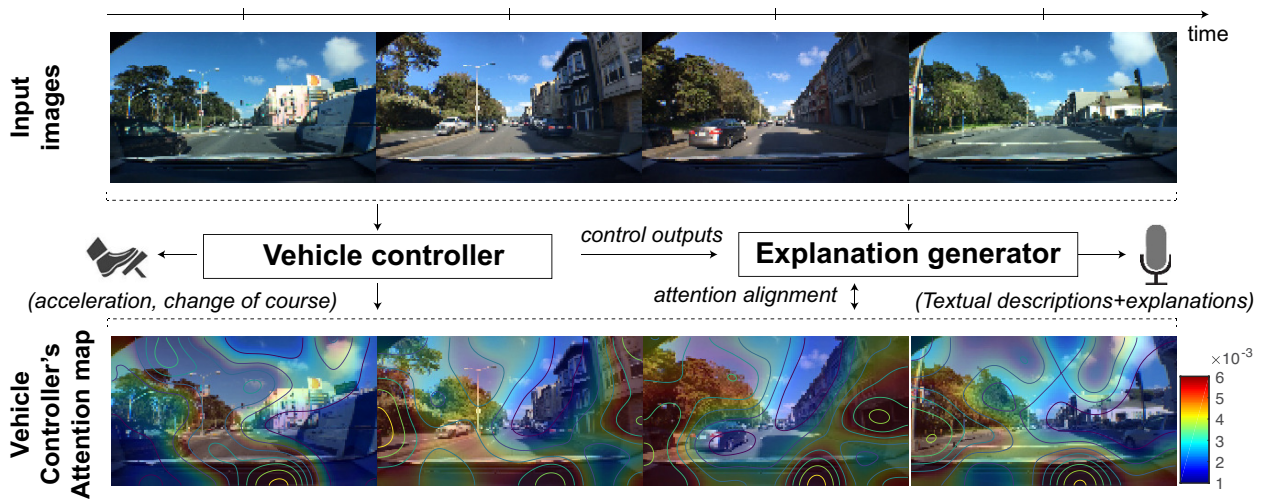
4.1 Problem Statement

Deep neural networks are an effective tool [9, 87] to learn vehicle controllers for self-driving cars in an end-to-end manner. Despite their effectiveness as function estimators, DNNs are typically cryptic black-boxes. There are no explainable states or labels in such a network, and representations are fully distributed as sets of activations. Explainable models that make deep models more transparent are important for a number of reasons: (i) user acceptance – self-driving vehicles are a radical technology for users to accept, and require a very high level of trust, (ii) understanding and extrapolation of vehicle behavior – users ideally should be able to anticipate what the vehicle will do in most situations, (iii) effective communication – they help user communicate preferences to the vehicle and vice versa.

Explanations can be either *rationalizations* – explanations that justify the system’s behavior in a post-hoc manner, or *introspective explanations* – explanations that are based on the system’s internal state. Introspective explanations represent *causal* relationships between the system’s input and its behavior, and address all the goals above. Rationalizations can address acceptance, (i) above, but are less helpful with (ii) understanding the causal behavior of the model or (iii) communication which is grounded in the vehicle’s internal state (known as theory of mind in human communication).

One way of generating introspective explanations is via visual attention [88, 37]. Visual attention filters out non-salient image regions, and image areas inside the attended region have potential causal effect on the output (those outside cannot). As shown in [37], additional salience filtering can be applied so that the attention map shows only regions that causally affect the output. Visual attention constrains the reasons for the controllers actions but does

*This work has been presented at the 15th European Conference on Computer Vision (ECCV), 2018.



Example of textual descriptions + explanations:

Ours: “The car is driving forward + because there are no other cars in its lane”

Human annotator: “The car heads down the street + because the street is clear.”

Figure 4.1: Our model predicts vehicle’s control commands, *i.e.* an acceleration and a change of course, at each timestep, while an explanation model generates a natural language explanation of the rationales, *e.g.* “The car is driving forward because there are no other cars in its lane”, and a visual explanation in the form of attention – attended regions directly influence the textual explanation generation process.

not *e.g.* tie specific actions to specific input regions *e.g.* “the vehicle slowed down because the light controlling the intersection is red”. It is also likely to be less convenient for passengers to replay the attention map vs. a (typically on-demand) speech presentation of a textual explanation.

In this work, we focus on generating textual descriptions and explanations, such as the pair: “vehicle slows down” and “because it is approaching an intersection and the light is red” as in Figure 4.1. Natural language has an advantage of being inherently understandable and does not require familiarity with the design of an intelligent system in order to provide useful information. In order to train such a model, we collect explanations from human annotators. Our explanation dataset is built on top of another large-scale driving dataset [87] collected from dashboard cameras in human driven vehicles. Annotators view the video dataset, compose descriptions of the vehicle’s activity and explanations for the actions that the vehicle driver performed.

Obtaining training data for vehicle explanations is by itself a significant challenge. The ground truth explanations are in fact often rationalizations (generated by an observer rather than the driver), and there are additional challenges with acquiring driver data. But even

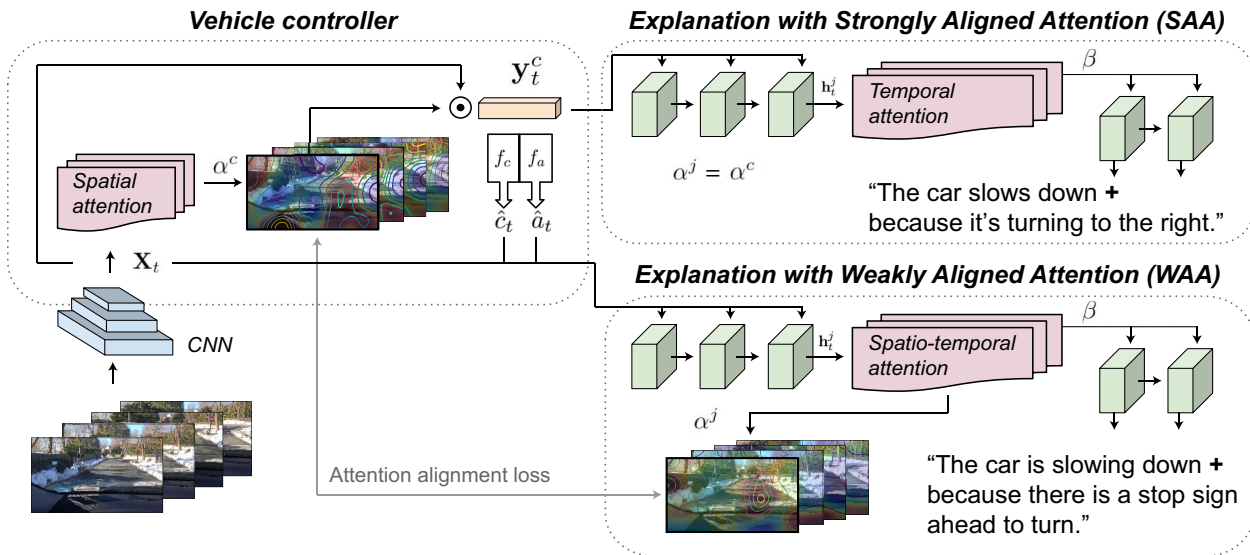


Figure 4.2: Vehicle controller generates spatial attention maps α^c for each frame, predicts acceleration and change of course (\hat{c}_t, \hat{a}_t) that condition the explanation. Explanation generator predicts temporal attention across frames (β) and a spatial attention in each frame (α^j). SAA uses α^c , WAA enforces a loss between α^j and α^c .

more than that, it is currently impossible to obtain human explanations of *what the vehicle controller was thinking*, *i.e.* a real ground truth. Nevertheless our experiments show that using *attention alignment* between controller and explanation models generally improves the quality of explanations, *i.e.* generates explanations which better match the human rationalizations of the driving videos.

Our contributions are as follows. (1) We propose an introspective textual explanation model for self-driving cars to provide easy-to-interpret explanations for the behavior of a deep vehicle control network. (2) We integrate our explanation generator with the vehicle controller by aligning their attentions to ground the explanation, and compare two approaches: attention-aligned explanations and non-aligned rationalizations. (3) We generated a large-scale Berkeley DeepDrive eXplanation (BDD-X) dataset with over 6,984 video clips annotated with driving descriptions, *e.g.* "The car slows down" and explanations, *e.g.* "because it is about to merge with the busy highway". Our dataset provides a new test-bed for measuring progress towards developing explainable models for self-driving cars.

4.2 Explainable Driving Model

In this paper, we propose a driving model that explains how a driving decision was made both (i) by visualizing image regions where the decision maker attends to and (ii) by generating a textual description and explanation of what has triggered a particular driving decision, *e.g.* “the car continues (description) because traffic flows freely (explanation)”. As we summarize in Figure 4.2, our model involves two parts: (1) a *Vehicle controller*, which is trained to learn human-demonstrated vehicle control commands, *e.g.* an acceleration and a change of course; our controller uses a visual (spatial) attention mechanism that identifies potentially influential image regions for the network’s output; (2) a *Textual explanation generator*, which generates textual descriptions and explanations controller behavior. The key to the approach is to align the attention maps.

4.2.1 Preprocessing

Our model is trained to predict two vehicle control commands, *i.e.* an acceleration and a change of course. At each time t , an acceleration, a_t , is measured by taking the derivative of speed measurements, and a change of course, c_t , is computed by taking a difference between a current vehicle’s course and a smoothed value by using simple exponential smoothing method [33]. We provide details in supplemental material. To reduce computational burden, we down-sample to 10Hz and reduce the input dimensionality by resizing raw images to a $90 \times 160 \times 3$ image with nearest-neighbor scaling algorithm. Each image is then normalized by subtracting the mean from the raw input pixels and dividing by its standard deviation. This preprocessing is applied to the latest 4 frames, which are then stacked to produce the final input to the neural network.

4.2.2 Convolutional Feature Encoder

We use a convolutional neural network to encode the visual information into a set of visual feature vectors at time t , *i.e.* convolutional feature cube $X_t = \{x_{t,1}, x_{t,2}, \dots, x_{t,l}\}$ where $x_{t,i} \in R^d$ for $i \in \{1, 2, \dots, l\}$ and l is the number of different spatial regions of the given input. Each feature vector contains a high-level description of objects present in a certain input region. This allows us to focus selectively on different regions of the given image by choosing a subset of these feature vectors. We use a five-layered convolutional network as in [9, 37] and omit max-pooling layers to prevent spatial information loss [45]. The output is a three-dimensional feature cube X_t and the feature block has the size $w \times h \times d$ at each time t .

4.2.3 Vehicle Controller

Our vehicle controller is trained in an end-to-end manner. Given a stream of dashcam images and the vehicle’s (current) sensor measurements, *e.g.* speed, the controller predicts

the acceleration and the change of course at each timestep. We utilize a deterministic soft attention mechanism that is trainable by standard back-propagation methods. The soft attention mechanism applies attention weights multiplicatively to the features and additively pools the results through the maps π . Our model feeds the context vectors y_t^c produced by the controller map π^c to the controller LSTM:

$$y_t^c = \pi^c(\{\alpha_{t,i}^c\}, \{x_{t,i}\}) = \sum_{i=1}^l \alpha_{t,i}^c x_{t,i} \quad (4.1)$$

where $i = \{1, 2, \dots, l\}$. $\alpha_{t,i}^c$ is an attention weight map output by a spatial softmax and satisfies $\sum_i \alpha_{t,i}^c = 1$. These attention weights can be interpreted as the probability over l convolutional feature vectors. A location with a high attention weight is salient for the task (driving). The attention model $f_{\text{attn}}^c(X_t, h_{t-1}^c)$ is conditioned on the previous LSTM state h_{t-1}^c , and the current feature vectors X_t . It comprises a fully-connected layer and a spatial softmax to yield normalized $\{\alpha_{t,i}^c\}$.

The outputs of the vehicle controller are the vehicle’s acceleration \hat{a}_t and the change of course \hat{c}_t . To this end, we use additional multi-layer fully-connected blocks with ReLU nonlinearities, denoted by $f_a(y_t^c, h_t^c)$ and $f_c(y_t^c, h_t^c)$. We also add the entropy H of the attention weight to the objective function:

$$\mathcal{L}_c = \sum_t ((a_t - \hat{a}_t)^2 + (c_t - \hat{c}_t)^2 + \lambda_c H(\alpha_t^c)) \quad (4.2)$$

The entropy is computed on the attention map as though it were a probability distribution. Minimizing loss corresponds to minimizing entropy. Low entropy attention maps are sparse and emphasize relatively few regions. We use a hyperparameter λ_c to control the strength of the entropy regularization term.

4.2.4 Attention Alignments

The controller attention map provides input regions that the network attends to, and these regions have a direct influence on the network’s output. Thus, to yield “introspective” explanation, we argue that the agent must attend to those areas. For example, if a vehicle controller predicts “acceleration” by detecting a green traffic light, the textual justification must mention this evidence, *e.g.* “because the light has turned green”. Here, we explain two approaches to align the vehicle controller and the textual justifier such that they look at the same input regions.

4.2.4.1 Strongly Aligned Attention (SAA)

A consecutive set of spatially attended input regions, each of which is encoded as a context vector y_t^c by the vehicle controller, can be directly used to generate a textual explanation (see Figure 4.2, right-top). Thus, models share a single layer of an attention. As we detail in

Section 4.2.5, our explanation module uses *temporal* attention with weights β to the controller context vectors $\{y_t^j, t = 1, \dots\}$ directly, and thus allows flexibility in output tokens relative to input samples.

4.2.4.2 Weakly Aligned Attention (WAA)

Instead of directly using vehicle controller’s attention, an explanation generator can have its own spatial attention network (see Figure 4.2, right-bottom). A loss, *i.e.* the Kullback-Leibler divergence (D_{KL}), between the two attention maps makes the explanation generator refer to the salient objects:

$$\mathcal{L}_a = \lambda_a \sum_t D_{\text{KL}}(\alpha_t^c || \alpha_t^j) = \lambda_a \sum_t \sum_{i=1}^l \alpha_{t,i}^c (\log \alpha_{t,i}^c - \log \alpha_{t,i}^j) \quad (4.3)$$

where α^c and α^j are the attention maps generated by the vehicle controller and the explanation generator model, respectively. We use a hyperparameter λ_a to control the strength of the regularization term.

4.2.5 Textual Explanation Generator

Our textual explanation generator takes sequence of video frames of variable length and generates a variable-length description/explanation. Descriptions and explanations are typically part of the same sentence in the training data but are annotated with a separator. In training and testing we use a synthetic separator token `<sep>` between description and explanation, but treat them as a single sequence. The explanation LSTM predicts the description/explanation sequence and outputs per-word softmax probabilities.

The source of context vectors for the description generator depends on the type of alignment between attention maps. For weakly aligned attention or rationalizations, the explanation generator creates its own spatial attention map α^j at each time step t . This map includes a loss against the controller attention map for weakly-aligned attention, but has no such loss when generating rationalizations. The attention map α^j is applied to the CNN output yielding context vectors y_t^j .

Our textual explanation generator explains the rationale behind the driving model, and thus we argue that a justifier needs the outputs from the vehicle motion predictor as an input. We concatenate a tuple (\hat{a}_t, \hat{c}_t) with a spatially-attended context vector y_t^j and y_t^c respectively for weakly and strongly aligned attention approaches. This concatenated vector is then used to update the LSTM for a textual explanation generation.

The explanation module applies *temporal* attention with weights β to either the controller context vectors directly $\{y_t^c, t = 1, \dots\}$ (strong alignment), or to the explanation vectors $\{y_t^j, t = 1, \dots\}$ (weak alignment or rationalization). Such input sequence attention is common in sequence-to-sequence models and allows flexibility in output tokens relative

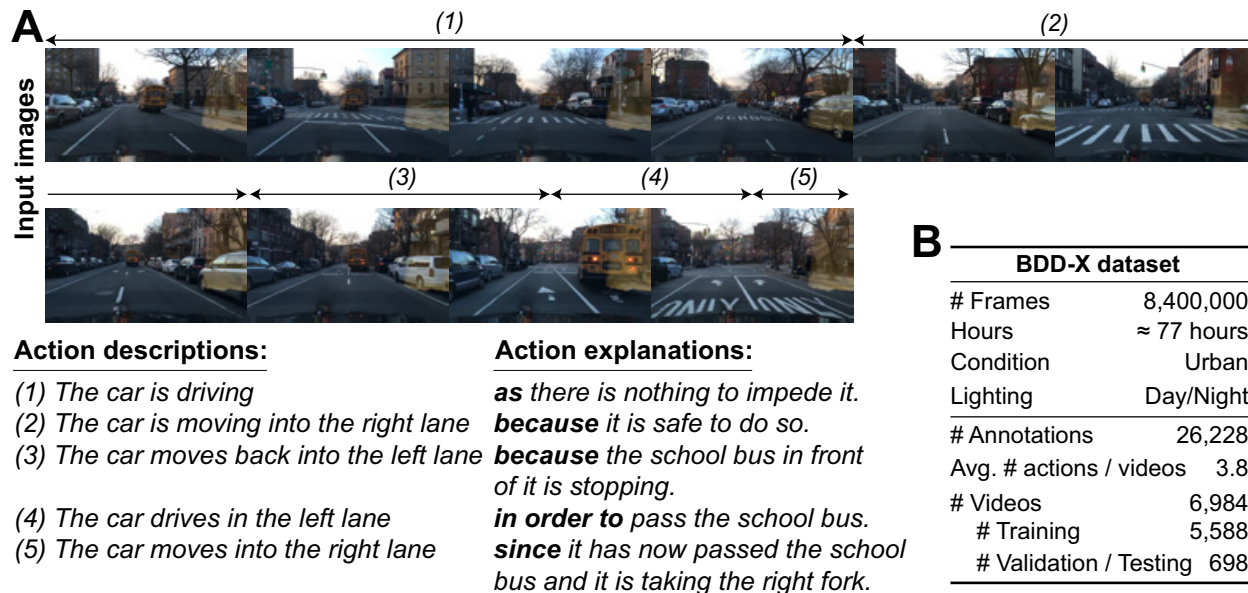


Figure 4.3: (A) Examples of input frames and corresponding human-annotated action description and justification of how a driving decision was made. For visualization, we sample frames at every two seconds. (B) BDD-X dataset details. Over 77 hours of driving with time-stamped human annotations for action descriptions and justifications.

to input samples [6]. The result of temporal attention application is (dropping the c or j superscripts on y):

$$z_k = \pi(\{\beta_{k,t}\}, \{y_t\}) = \sum_{t=1}^T \beta_{k,t} y_t \quad (4.4)$$

where $\sum_t \beta_{k,t} = 1$. The weight $\beta_{k,t}$ at each time k (for sentence generation) is computed by an attention model $f_{\text{attn}}^e(\{y_t\}, h_{k-1}^e)$, which is similar to the spatial attention as we explained in previous section (see supplemental material for details).

To summarize, we minimize the following negative log-likelihood (for training our justifier) as well as vehicle control estimation loss \mathcal{L}_c and attention alignment loss \mathcal{L}_a :

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_a - \sum_k \log p(o_k | o_{k-1}, h_k^e, z_k) \quad (4.5)$$

4.3 Berkeley DeepDrive eXplanation Dataset (BDD-X)

In order to effectively generate and evaluate textual driving rationales we have collected textual justifications for a subset of the Berkeley Deep Drive (BDD) dataset [87]. This dataset contains videos, approximately 40 seconds in length, captured by a dashcam mounted behind the front mirror of the vehicle. Videos are mostly captured during urban driving in various weather conditions, featuring day and nighttime. The dataset also includes driving on other road types, such as residential roads (with and without lane markings), and contains all the typical driver’s activities such as staying in a lane, turning, switching lanes, etc. Alongside the video data, the dataset provides a set of time-stamped sensor measurements, such as vehicle’s velocity, course, and GPS location. For sensor logs unsynchronized with the time-stamps of video data, we use the estimates of the interpolated measurements.

In order to increase trust and reliability, the machine learning system underlying self driving cars should be able to explain why at a certain time they make certain decisions. Moreover, a car that justifies its decision through natural language would also be user friendly. Hence, we populate a subset of the BDD dataset with action description and justification for all the driving events along with their timestamps. We provide examples from our *Berkeley Deep Drive eXplanation (BDD-X)* dataset in Figure 4.3 (A).

4.3.1 Annotation


We provide a driving video and ask a human annotator in Amazon Mechanical Turk to imagine herself being a driving instructor. Note that we specifically select human annotators who are familiar with US driving rules. The annotator has to describe *what* the driver is doing (especially when the behavior changes) and *why*, from a point of view of a driving instructor. Each described action has to be accompanied with a start and end time-stamp. The annotator may stop the video, forward and backward through it while searching for the activities that are interesting and justifiable.

To ensure that the annotators provide us the driving rationales as well as descriptions, we require that they separately enter the *action description* and the *action justification*: e.g. “The car is moving into the left lane” and “because the school bus in front of it is stopping.”. In our preliminary annotation studies, we found that giving separate annotation boxes is helpful for the annotator to understand the task and perform better.

4.3.2 Our Amazon Mechanical Turk annotation interface

Our annotation prompt is demonstrated on Figure 4.4. To ensure that the annotators provide us the driving rationales as well as descriptions, we require that they separately enter the action description and the action justification e.g. : “The car is moving into the left lane” and “because the school bus in front of it is stopping”. In our preliminary annotation studies,

(Click to expand)



0:00 / 0:40

[Link to video](#)

Instructions
 Imagine you are a driving instructor.
 Fill the two text boxes with the following.

(1) Describe **WHAT** the driver is doing, especially when the behavior changes.
 The car is going down the highway
 The car is passing another car while accelerating

(2) **WHY** is the driver doing that / changing behavior
 ... as the lane is free
 ... since the car in front is going slowly and the left lane is empty

- Do **not mention** objects that are **not relevant to the action**.
- Do **not use proper nouns** or names of the places.
- Do not use figures of speech.
- Do **not presume what the driver is thinking**

Please enter the time stamps as 2-digit whole numbers. No punctuation. I.e. 00 09

You'll note the examples always have a conjunction word such as "as, because, since" etc. This is to indicate the *justification* for the action.

Start: End: Action:
 00 00 The car is stopping

Justification:
 because the light is red.

Start: End: Action:
 00 00 The car is stopping

Justification:
 Because the light is red.

Start: End: Action:
 00 00 The car is stopping

Justification:
 Because the light is red.

Start: End: Action:
 00 00 The car is stopping

Justification:
 Because the light is red.

[Click here if you require additional fields.](#)

Figure 4.4: Our Amazon Mechanical Turk annotation interface. Shown is a “dummy” input (The car is stopping + Because the light is red), not an actual annotation.

we found that giving separate annotation boxes are helpful for the annotator to understand the task and perform better.

4.3.3 Dataset Statistics

Our dataset (see Figure 4.3 (B)) is composed of over 77 hours of driving within 6,984 videos. The videos are taken in diverse driving conditions, *e.g.* day/night, highway/city/countryside, summer/winter etc. On an average of 40 seconds, each video contains around 3-4 actions, *e.g.* speeding up, slowing down, turning right etc., all of which are annotated with a description and an explanation. Our dataset contains over 26K activities in over 8.4M frames. We introduce a training, a validation and a test set, containing 5,588, 698 and 698 videos,

respectively.

4.3.4 Inter-human agreement

Although we cannot have access to the internal thought process of the drivers, one can infer the reason behind their actions using the visual evidence of the scene. Besides, it would be challenging to setup the data collection process which enables drivers to report justifications for all their actions, if at all possible. We ensure the high quality of the collected annotations by relying on a pool of qualified workers (*i.e.* they pass a qualification test) and selective manual inspection.

Further, we measure the inter-human agreement on a subset of 998 training videos, each of which has been annotated by two different workers. Our analysis is as follows. In 72% of videos the number of annotated intervals differs by less than 3. The average temporal *IoU* across annotators is 0.63 ($SD = 0.21$). When $IoU > 0.5$ the CIDEr score across action descriptions is 142.60, across action justifications it is 97.49 (random choice: 39.40/28.39, respectively). When $IoU > 0.5$ and action descriptions from two annotators are identical (165 clips¹) the CIDEr score across justifications is 200.72, while a strong baseline, selecting a justification from a different video with the same action description, results in CIDEr score 136.72. These results show an agreement among annotators and relevance of collected action descriptions and justifications.

4.3.5 Coverage of justifications

BDD-X dataset has over 26k annotations (77 hours) collected from a substantial random subset of large-scale crowd-sourced driving video dataset, which consists of all the typical driver’s activities during urban driving. The vocabulary of training action descriptions and justifications is 906 and 1,668 words respectively, suggesting that justifications are more diverse than descriptions. Some of the common actions are (frequency decreasing): moving forward, stopping, accelerating, slowing, turning, merging, veering, pulling [in]. Justifications cover most of the relevant concepts: traffic signs/lights, cars, lanes, crosswalks, passing, parking, pedestrians, waiting, blocking, safety etc.

Table 4.1 includes word counts for the top-30 most frequent words (excluding stop-words) used in the action descriptions and action explanations, respectively. Note, that word counts are obtained but taking all word forms into account (*slow*, *slows*, *slowing*, *slowed*, *slowly*, etc). Most common actions are related to changes in speed, driving forward and turning. However many also include merging, pulling, changing lanes, veering, and less frequent actions like reversing, parking or using wipers. Action explanations cover a diverse list of concepts relevant to driving scenario, such as state of traffic/lanes, traffic lights/signs, pedestrians crossing the street, passing other cars, etc. Although less frequent, many explanations

¹The number of video intervals (not full videos), where the provided action descriptions (not explanations) are identical (common actions *e.g.* “the car slows down”).

Table 4.1: BDD-X dataset details. We provide counts for top-30 words, where words are counted along with all their forms, such as *slow*, *slows*, *slowing*, *slowed*, *slowly*, etc.

BDD-X action descriptions		BDD-X action explanations	
Word	Count	Word	Count
stop	6879	traffic	7486
slow	6122	light	6116
forward	4322	red	3979
drive	3994	move	3915
move	3273	clear	3660
accelerate	2882	ahead	3629
right	2616	road	3528
left	2574	stop	3430
turn	1912	lane	3407
road	1907	turn	3333
lane	1832	front	2715
street	1704	green	1928
speed	1072	park	1523
come	1033	intersection	1513
merge	923	right	1464
pull	723	slow	1395
intersection	717	cars	1390
head	629	left	1272
continue	625	street	1225
slight	554	forward	984
make	539	sign	984
brake	517	make	718
travel	513	approach	702
highway	450	speed	681
maintain	390	down	658
steer	371	pedestrian	649
proceed	359	go	614
steady	329	cross	588
complete	323	get	536
veer	214	pass	490
park	209	enter	488
roll	173	wait	468
speeding	140	way	423

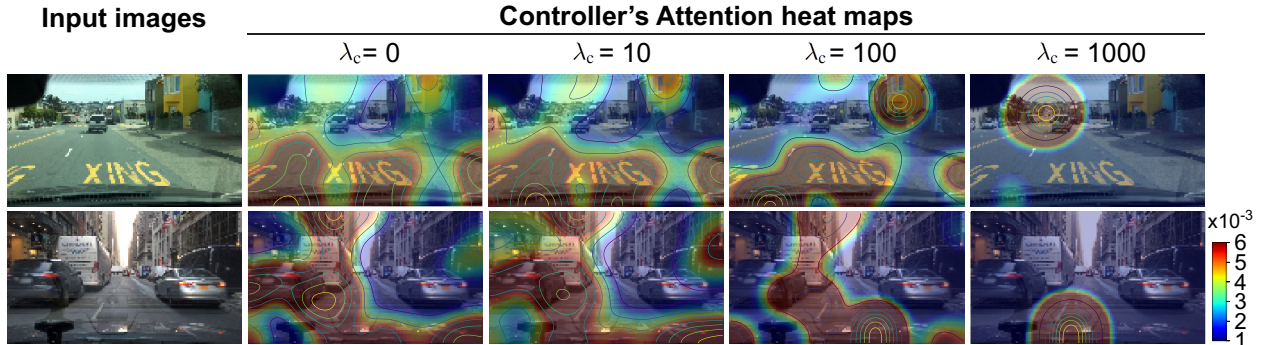


Figure 4.5: Vehicle controller’s attention maps in terms of four different entropy regularization coefficient $\lambda_c = \{0, 10, 100, 1000\}$. Red parts indicate where the model pays more attention. Higher value of λ_c makes the attention maps sparser. We observe that sparser attention maps improves the performance of generating textual explanations, while control performance is slightly degraded.

contain references to weather conditions (rain, snow), different types of vehicles (buses, vans, trucks), road bumps and safety of the performed action.

We provide some additional examples in Table 4.2, showing, in particular, that some explanations require complex reasoning (1,2,3) and illustrate attention to detail (4,5,6).

Description	Explanation
1: The car is making its way carefully through the intersection	because another car has cut it off and pedestrians are approaching the crosswalk.
2: The car stops behind the truck	because it’s waiting for a car in the opposite lane to pass by.
3: The car slows to a stop next to a line of parked cars	since there is no where available to park, but another car up ahead is also double parked.
4: The car slows down	because it’s entering a school crossing zone.
5: The car pulls over to the left side of the street	to avoid a large pothole on the right.
6: The car is moving at a constant slow pace	because it is a single lane road with snow and ice on the road.

Table 4.2: Example annotations where explanations require complex reasoning (1,2,3) and demonstrate attention to detail (4,5,6).

4.4 Experiments

Here, we first provide our training and evaluation details, then make a quantitative and qualitative analysis of our vehicle controller and our textual justifier.

Model	λ_c	Mean of absolute error (MAE)		Mean of distance correlation	
		Acceleration (m/s ²)	Course (degree)	Acceleration (m/s ²)	Course (degree)
CNN+FC [9] [†]	-	6.92 [7.50]	12.1 [19.7]	0.17 [0.15]	0.16 [0.14]
CNN+FC [9]+P	-	6.09 [7.73]	6.74 [14.9]	0.21 [0.18]	0.39 [0.33]
CNN+LSTM+Attention [37] [†]	-	6.87 [7.44]	10.2 [18.4]	0.19 [0.16]	0.22 [0.18]
CNN+LSTM+Attention+P (Ours)	1000	5.02 [6.32]	6.94 [15.4]	0.65 [0.25]	0.43 [0.33]
CNN+LSTM+Attention+P (Ours)	100	2.68 [3.73]	6.17 [14.7]	0.78 [0.28]	0.43 [0.34]
CNN+LSTM+Attention+P (Ours)	10	2.33 [3.38]	6.10 [14.7]	0.81 [0.27]	0.46 [0.35]
CNN+LSTM+Attention+P (Ours)	0	2.29 [3.33]	6.06 [14.7]	0.82 [0.26]	0.47 [0.35]

Table 4.3: Comparing variants of our vehicle controller with different values of the entropy regularization coefficient $\lambda_c=\{0, 10, 100, 1000\}$ and the state-of-the-art. High value of λ_c produces low entropy attention maps that are sparse and emphasize relatively few regions. [†]: Models use a single image frame as an input. The standard deviation is in braces. *Abbreviation*: FC (fully connected layer), P (prior inputs)

4.4.1 Training and Evaluation Details

As the convolutional feature encoder, we use 5-layer CNN [9] that produces a $12 \times 20 \times 64$ -dimensional convolutional feature cube from the last layer. The controller following the CNN has 5 fully connected layers (*i.e.* #hidden dims: 1164, 100, 50, 10, respectively), which predict the acceleration and the change of course, and is trained end-to-end from scratch. Using other more expressive networks may give a performance boost over our base CNN configuration, but these explorations are out of our scope. Given the obtained convolutional feature cube, we first train our vehicle controller, and then the explanation generator (single layer LSTM unless stated otherwise) by freezing the control network. For training, we use Adam optimizer [40] and dropout [73] of 0.5 at hidden state connections and Xavier initialization [23]. The standard dataset is split as 80% (5,588 videos) as the training set, 10% (698 videos) as the test, and 10% (698 videos) as the validation set. Our model takes less than a day to train on a single NVIDIA Titan X GPU.

For evaluating the vehicle controller we use the mean absolute error (lower is better) and the distance correlation (higher is better) and for the justifier we use BLEU [59], METEOR [43], and CIDEr-D [81], as well as human evaluation. The former metrics are widely used for the evaluation of video and image captioning models automatically against ground truth.

4.4.2 Evaluating Vehicle Controller

We start by quantitatively comparing variants of our vehicle controller and the state of the art, which include variants of the work by Bojarski *et al.* [9] and Kim *et al.* [37] in Table 4.3. Note that these works differ from ours in that their output is the curvature of driving, while our model estimates continuous acceleration and the change of course values. Thus, their models have a single output, while ours estimate both control commands. In this experiment, we replaced their output layer with ours. For a fair comparison, we use an identical CNN for all models.

In this experiment, each model estimates vehicle’s acceleration and the change of course. Our vehicle controller predicts acceleration and the change of course, which generally requires prior knowledge of vehicle’s current state, *i.e.* speed and course, and navigational inputs, especially in urban driving. We observe that the use of the latest four consecutive frames and prior inputs (*i.e.* vehicle’s motion measurement and navigational information) improves the control prediction accuracy (see 3rd vs. 7th row), while the use of visual attention also provides improvements (see 1st vs. 3rd row). Specifically, our model without the entropy regularization term (last row) performs the best compared to CNN based approaches [9] and [37]. The improvement is especially pronounced for acceleration estimation.

In Figure 4.5 we compare input images (first column) and corresponding attention maps for different entropy regularization coefficients $\lambda_c = \{0, 10, 100, 1000\}$. Red is high attention, blue is low. As we see, higher λ_c lead to sparser maps. For better visualization, an attention map is overlaid by its contour lines and an input image.

Quantitatively, the controller performance (error and correlation) slightly degrade as λ_c increases and the attention maps become more sparse (see bottom four rows in Table 4.3). So there is some tension between sparse maps (which are more interpretable), and controller performance. An alternative to regularization, [37] use causal filtering over the controller’s attention maps and achieve about 60% reduction in “hot” attention pixels. Causal filtering is desirable for the present work not only to improve sparseness but because after causal filtering, “hot” regions necessarily *do* have a causal effect on controller behavior, whereas unfiltered attention regions may not. We will explore it in future work.

4.4.3 Evaluating Textual Explanations

In this section, we evaluate textual explanations against the ground truth explanation using automatic evaluation measures, and also provide human evaluation followed by a qualitative analysis.

For state-of-the-art comparison, we implement the S2VT [82] and its variants. Note that in our implementation S2VT uses our CNN and does not use optical flow features. In Table 4.4, we report a summary of our experiment validating the quantitative effectiveness of our approach. Rows 5-10 show that best explanation results are generally obtained with weakly-aligned attention. Comparing with row 4, the introspective models all gave higher scores than the rationalization model for explanation generation. Description scores are more

Type	Model	Control in- puts	λ_a	λ_c	Explanations (<i>e.g.</i> “because the light is red”)			Descriptions (<i>e.g.</i> “the car stops”)		
					BLEU-4	METEOR	CIDEr-D	BLEU-4	METEOR	CIDEr-D
	S2VT [82]	N	-	-	6.332	11.19	53.35	30.21	27.53	179.8
	S2VT [82]+SA	N	-	-	5.668	10.96	51.37	28.94	26.91	171.3
	S2VT [82]+SA+TA	N	-	-	5.847	10.91	52.74	27.11	26.41	157.0
<i>Rationalization</i>	Ours (no constraints)	Y	0	0	6.515	12.04	61.99	31.01	28.64	205.0
<i>Introspective explanation</i>	Ours (with SAA)	Y	-	0	6.998	12.08	62.24	32.44	29.13	213.6
	Ours (with SAA)	Y	-	10	6.760	12.23	63.36	29.99	28.26	203.6
	Ours (with SAA)	Y	-	100	7.074	12.23	66.09	31.84	29.11	214.8
	Ours (with WAA)	Y	10	0	6.967	12.14	64.19	32.24	29.00	219.7
	Ours (with WAA)	Y	10	10	6.951	12.34	68.56	30.40	28.57	206.6
	Ours (with WAA)	Y	10	100	7.281	12.24	69.52	32.34	29.22	215.8

Table 4.4: Comparing generated and ground truth (columns 6-8) descriptions (*e.g.* “the car stops”) and explanations (*e.g.* “because the light is red”). We implement S2VT [82] and variants with spatial attention (SA) and temporal attention (TA) as a baseline. We tested two different attention alignment approaches, *i.e.* WAA (weakly aligned attention) and SAA (strongly aligned attention), with different combinations of two regularization coefficients: $\lambda_a=\{0, 10\}$ for the attention alignment and $\lambda_c=\{0, 10, 100\}$ for the vehicle controller. Rationalization baseline relies on our model (WAA approach) but has no attention alignment. Note that we report all values as a percentage.

mixed, but most of the introspective model scores are higher. As we will see in the next section, our rationalization model focuses on visual saliencies, which is sometimes different from what controller actually “looks at”. For example, in Figure 5 (5th example), our controller sees the front vehicle and our introspective models generate explanations such as “because the car in front is moving slowly”, while our rationalization model does not see the front vehicle and generates “because it’s turning to the right”.

As our training data are human observer annotations of driving videos, and they are not the explanations of drivers, they are post-hoc rationalizations. However, based on the visual evidence, (*e.g.* the existence of a turn right sign explains why the driver has turned right even if we do not have access to the exact thought process of the driver), they reflect typical causes of human driver behavior. The data suggest that grounding the explanations in controller internal state helps produce explanations that better align with human third-party explanations. Biasing the explanations toward controller state (which the WAA and SAA models do) improves their plausibility from a human perspective, which is a good sign.

Type	Model	Control in- puts	λ_a	λ_c	Correctness rate	
					Explanations	Descriptions
<i>Rationalization</i>	Ours (no constraints)	Y	0	0	64.0%	92.8%
<i>Introspective explana- tion</i>	Ours (with SAA)	Y	-	100	62.4%	90.8%
	Ours (with WAA)	Y	10	100	66.0%	93.5%

Table 4.5: Human evaluation of the generated action descriptions and explanations for randomly chosen 250 video intervals. We measure the success rate where at least 2 human judges rate the generated description or explanation with a score 1 (correct and specific/detailed) or 2 (correct).

We further analyze human preference in the evaluation below.

4.4.4 Human Evaluation

In our first human evaluation experiment the human judges are only shown the *descriptions*, while in the second experiment they only see the *explanations* (e.g. “*The car ... because < explanation >*”), to exclude the effect of explanations/descriptions on the ratings, respectively. We randomly select 250 video intervals and compare the Rationalization, WAA ($\lambda_a=10$, $\lambda_c=100$) and SAA ($\lambda_c=100$) predictions. The humans are asked to rate a description/explanation on the scale $\{1..4\}$ (1: correct and specific/detailed, 2: correct, 3: minor error, 4: major error). We collect ratings from 3 human judges for each task. Finally, we compute the majority vote, *i.e.* at least 2 out of 3 judges should rate the description/explanation with a score 1 or 2.

As shown in Table 4.5, our WAA model outperforms the other two, supporting the results above. Interestingly, Rationalization does better than SAA on this subset, according to humans. This is perhaps because the explanation in SAA relies on the exact same visual evidence as the controller, which may include counterfactually important regions (*i.e.* there *could be* a stop sign here), but may confuse the explanation module.

4.4.5 Qualitative Analysis of Textual Justifier

As Figure 4.6 shows, our proposed textual explanation model generates plausible descriptions and explanations, while our model also provides attention visualization of their evidence. In the first example of Figure 4.6, controller sees neighboring vehicles and lane markings, while explanation model generates “the car is driving forward (description)” and “because traffic is moving freely (explanation)”. In Figure 4.6, we also provide other examples that cover common driving situations, such as driving forward (1st example), slowing/stopping (2nd,

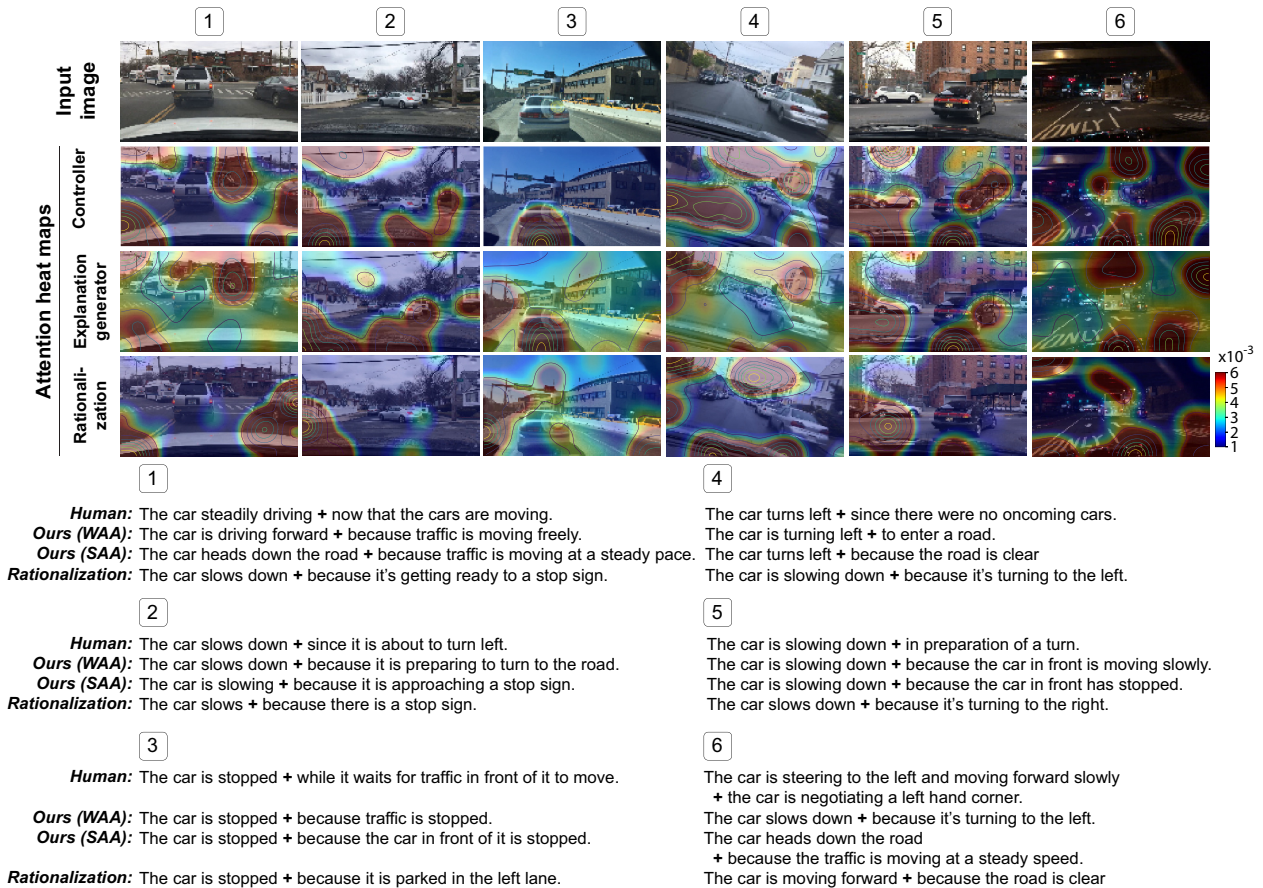


Figure 4.6: Example descriptions and explanations generated by our model compared to human annotations. We provide (top row) input raw images and attention maps by (from the 2nd row) vehicle controller, textual explanation generator, and rationalization model (Note: $(\lambda_c, \lambda_a) = (100, 10)$ and the synthetic separator token is replaced by '+').

3rd, and 5th), and turning (4th and 6th). We also observe that our explanations have significant diversity, *e.g.* they provide various reasons for stopping: red lights, stop signs, and traffic. We provide more diverse examples as supplemental materials.

In Figure 4.7 and 4.8, we provide more examples of explanations of the driving decisions by exploiting both attention visualization and textual justification.

Our model is also able to generate novel explanations not present in the training set. We provide some examples as follows: (1) “because there are no obstructions in the lane ahead”, (2) “because the car ahead is slowing to make a stop”, (3) “because the car is turning onto a different street”, (4) “as it is now making a turn to enter another street”, (5) “the car is stopped at an intersection at a red light”, and (6) “because there are no other cars in the

intersection”.

4.5 Related Work

In this section, we review existing work on end-to-end learning for self-driving cars as well as work on visual explanation and justification.

4.5.1 End-to-End Learning for Self-Driving Cars:

Most of vehicle controllers for self-driving cars can be divided in two types of approaches [13]: (1) a mediated perception-based approach and (2) an end-to-end learning approach. The mediated perception-based approach depends on recognizing human-designated features, such as lane markings, traffic lights, pedestrians or cars, which generally require demanding parameter tuning for a balanced performance [58]. Notable examples include [79], [11], and [46].

As for the end-to-end approaches, recent works [9, 87] suggest that neural networks can be successfully applied to self-driving cars in an end-to-end manner. Most of these approaches use behavioral cloning that learns a driving policy as a supervised learning problem over observation-action pairs from human driving demonstrations. Among these, [9] present a deep neural vehicle controller network that directly maps a stream of dashcam images to steering controls, while [87] use a deep neural network that takes input raw pixels and prior vehicle states and predict vehicle’s future motion. Despite their potential, the effectiveness of these approaches is limited by their inability to explain the rationale for the system’s decisions, which makes their behavior opaque and uninterpretable. In this work, we propose an end-to-end trainable system for self driving cars that is able to justify its predictions visually via attention maps and textually via natural language.

4.5.2 Visual and Textual Explanations

The importance of explanations for an end-user has been studied from the psychological perspective [51, 52], showing that humans use explanations as a guide for learning and understanding by building inferences and seeking propositions or judgments that enrich their prior knowledge. They usually seek for explanations to fill the requested gap depending on prior knowledge and goal in question.

In support of this trend, recently explainability has been growing as a field in computer vision and machine learning. Especially, there is a growing interest in introspective deep neural networks. [93] use deconvolution to visualize inner-layer activations of convolutional networks. [44] propose automatically-generated captions for textual explanations of images. [10] develop a richer notion of contribution of a pixel to the output. However, a difficulty with deconvolution-style approaches is the lack of formal measures of how the network output is affected by spatially-extended features (rather than pixels). Exceptions to this rule are

attention-based approaches. [37] propose attention-based approach with causal filtering that removes spurious attention blobs. However, it is also important to be able to justify the decisions that were made and explain why they are reasonable in a human understandable manner, *i.e.* a natural language. For an image classification problem, [27, 28] used an LSTM [30] caption generation model that generates textual justifications for a CNN model. [61] combine attention-based model and a textual justification system to produce an interpretable model. To our knowledge, ours is the first attempt to justify the decisions of a real-time deep controller through a combination of attention and natural language explanations on a stream of images.

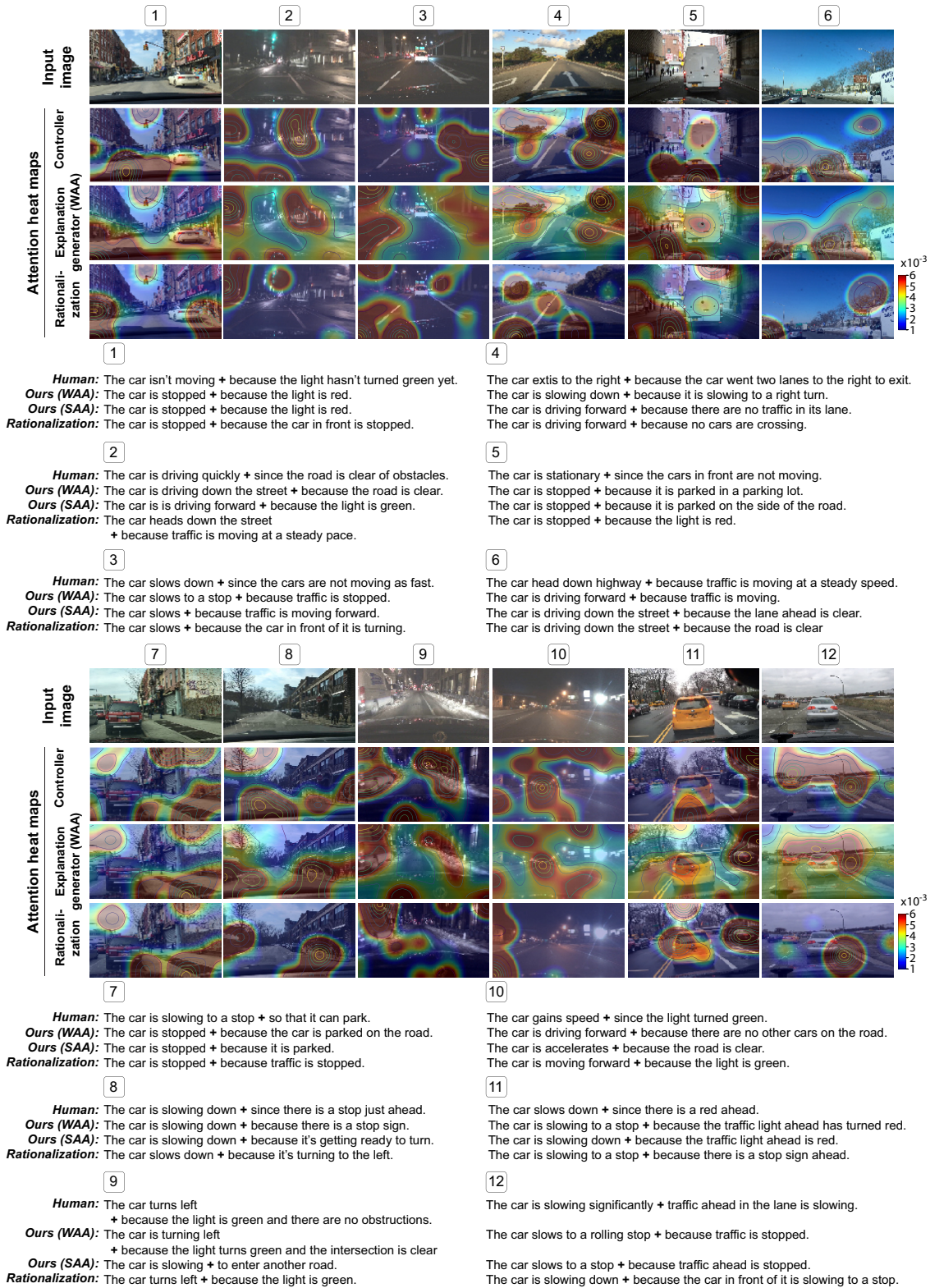
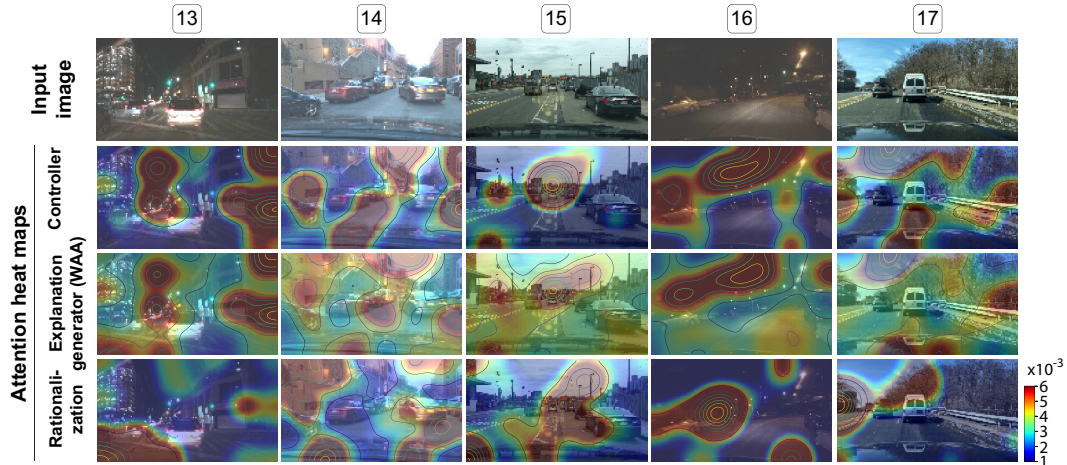


Figure 4.7: Examples of descriptions and explanations. In this experiment, we use (λ_c, λ_a) as (100,10). A synthetic separator token is replaced by '+'.



<p>13</p> <p><i>Human:</i> The car is stopped + as the light is red.</p> <p><i>Ours (WAA):</i> The car is stopped + because the light is red.</p> <p><i>Ours (SAA):</i> The car is moving slowly down the road + because the traffic in front is stopped at a red light.</p> <p><i>Rationalization:</i> The car accelerates + because the car in front is moving forward.</p>	<p>14</p> <p><i>Human:</i> The car backs up + to let another car pull out of a parking space he wants.</p> <p><i>Ours (WAA):</i> The car is accelerating + because traffic is moving forward.</p> <p><i>Ours (SAA):</i> The car is stopped + because the light is red.</p> <p><i>Rationalization:</i> The car is moving slowly forward + to get around a car in the way.</p>	<p>15</p> <p><i>Human:</i> The car is stopped + because it is parked.</p> <p><i>Ours (WAA):</i> The car is stopped + because it is parked.</p> <p><i>Ours (SAA):</i> The car is stopped + because the car is parked.</p> <p><i>Rationalization:</i> The car is stopped at an intersection + because the light is red.</p>	<p>16</p> <p>The car is accelerating and turning right + the car is turning right through the intersection + the car is turning right + to enter another road.</p> <p>The car is turning right + because the light turns green.</p> <p>The car is slowing down + because it's turning to the right.</p>	<p>17</p> <p>The car is turning right + to enter another road.</p> <p>The car is driving forward + because traffic is clear.</p> <p>The car turns right + because the road is clear of traffic.</p> <p>The car is driving down the road + because the road is clear.</p>
--	---	--	--	---

Figure 4.8: Additional examples of descriptions and explanations. In this experiment, we use (λ_c, λ_a) as $(100, 10)$. A synthetic separator token is replaced by '+'.

Chapter 5

Grounding Human-to-Vehicle Advice for Self-driving Vehicles

5.1 Problem Statement

Dramatic progress in self-driving vehicle control has been made in the last several years. The recent achievements [9, 87] suggest that deep neural models can be applied to vehicle controls in an end-to-end manner by effectively learning latent representations from data. Explainability of these deep controllers has increasingly been explored via a visual attention mechanism [37], a deconvolution-style approach [10], and a natural language model [39]. Such explainable models will be an important element of human-vehicle interaction because they allow people and vehicles to understand and anticipate each other’s actions, hence to cooperate effectively.

However, the network’s understanding of a scene is limited by the training data: image areas are only attended to if they are salient to the (training) driver’s subsequent action. We have found that this leads to semantically-shallow models that under-attend to important cues (like pedestrians) that do not predict vehicle behavior as well as other cues, like the presence of a stop light or intersection. We also believe its important for driving models to be able to adapt the “style” of the journey to user input (fast, gentle, scenic route, avoid freeways etc). We use the term “advice” to cover high-level instructions to the vehicle controller about how to drive, including what to attend to. We distinguish advice from explicit commands to the vehicle: which may be problematic if the passenger is not fully attending to the vehicle’s environment.

The goal of this work is to augment imitation learning datasets with long-term advice from humans (*e.g.* driving instructors) and in the shorter term, from passengers in the vehicle.

*This work has been presented at the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

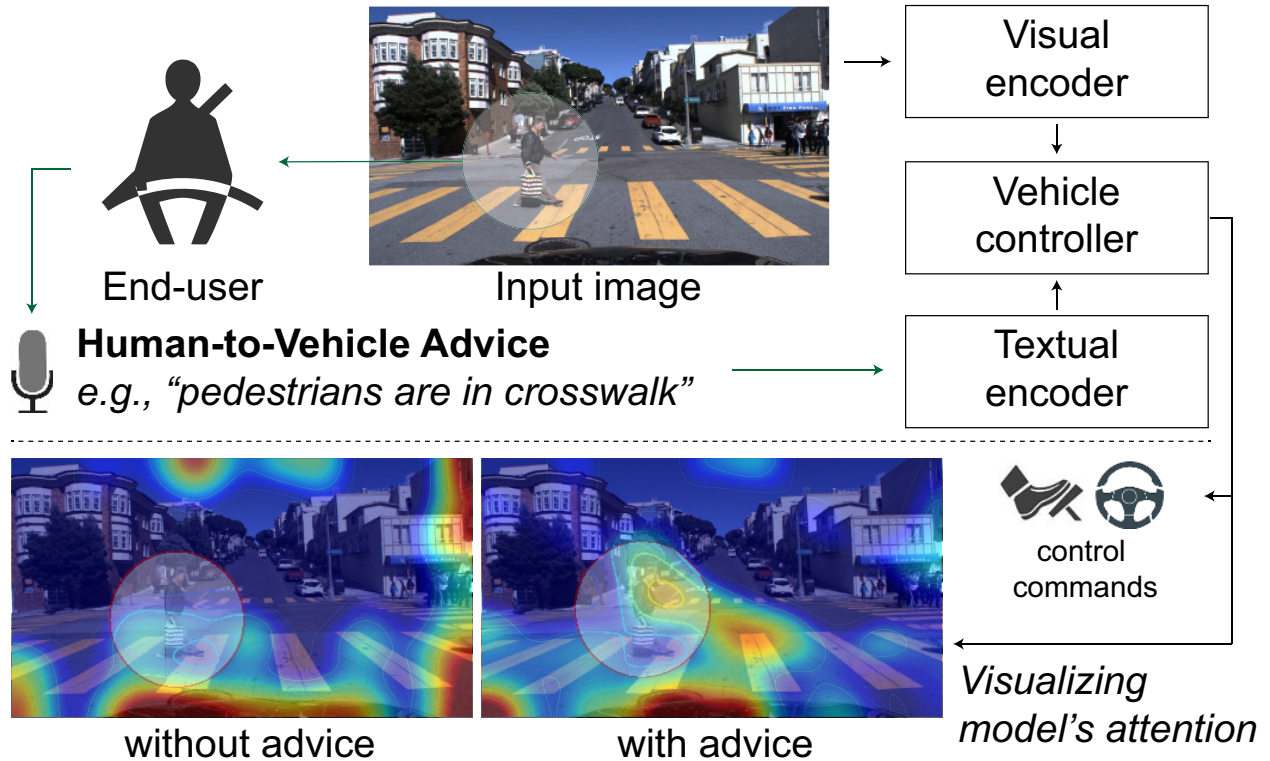


Figure 5.1: Our model takes human-to-vehicle advice as an input, *i.e.* “pedestrians are in crosswalk”, and grounds it into the vehicle controller, which then predicts a sequence of control commands, *i.e.* a steering wheel angle and a vehicle’s speed. Our driving model also provides a visual explanation in the form of attention - highlighted regions have a direct influence on the function being estimated. Visualizing attention maps helps the end-users acknowledge the acceptance of their advice.

In full generality, advice might take the form of condition-action rules “if you see a child’s toy on the sidewalk, slow down”. For the present paper, we study the simpler task of accepting short-term textual advice about action or perception.

In this work, we propose a novel driving model that takes natural language inputs (*i.e.* human-to-vehicle advice) from an end-user. Here, we focus on two forms of advice: (1) goal-oriented advice (top-down signal) – to influence the vehicle in a navigation task (*e.g.* “drive slow in a school zone”), (2) stimulus-driven advice (bottom-up signal) – conveys some visual stimuli that the user expects their attention to be actively looked by the vehicle controller (*e.g.* “there is a pedestrian crossing”). As shown in Figure 5.1, the controller needs three main capabilities to handle such advice. (i) Perceptual primitives to evaluate the controller’s behavior. (ii) The ability to understand the user’s utterance and to ground

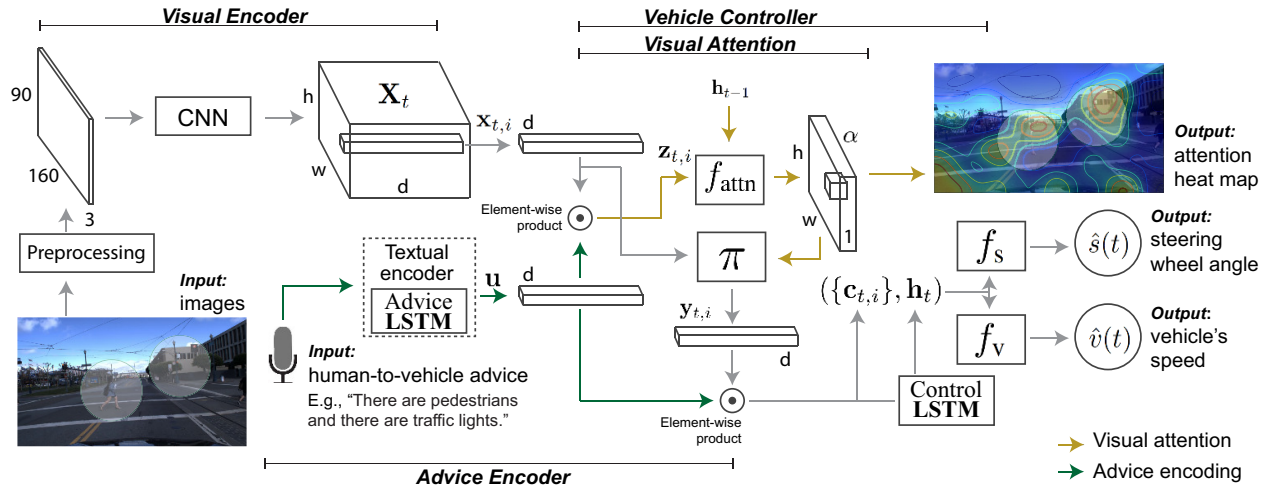


Figure 5.2: Our model consists of three main parts: (1) a visual encoder (CNN here), (2) an advice encoder, which encodes end-user’s utterance (advice) and ground it into the vehicle controller (see green arrows), and (3) an interpretable vehicle controller, which predicts two vehicle control commands (*i.e.* a speed and a steering angle command) from an input raw image stream in an end-to-end manner. Our model also utilizes a (spatial) visual attention mechanism to visualize where and what the model sees (see yellow arrows).

it in the trained perceptual primitives. (iii) Explainability of the controller’s internal state to communicate with the vehicle. We propose that such capabilities can be learned during off-line training.

Our contributions are as follows. (1) We propose a novel advisable driving model that takes human-to-vehicle advice and grounds it into the vehicle controller. (2) We internalize the (stimulus-driven) advice – aligning its attention to make the model refer to the important salient objects even when advice is not available. (3) We generated a large-scale dataset called Honda Research Institute-Advice Dataset (HAD) with over 5,600 video clips (over 32 hours) with human-to-vehicle advice annotations, *e.g.* “there is a pedestrian pushing a stroller through the crosswalk”. The dataset will be available and will provide a new test-bed for measuring progress towards developing advisable models for self-driving cars.

5.2 Advisable Driving Model

As we summarized in Figure 5.2, our model involves three main parts: (1) a *Visual encoder*, which extract high-level visual descriptions by utilizing the convolutional neural network (CNN). (2) An *Advice encoder*, which is a natural language model that encodes end-user’s utterance into a latent vector and ground it into the vehicle controller. (3) An *Interpretable*

vehicle controller, which is trained to predict two control commands (widely used for self-driving vehicle control) in an end-to-end manner, *i.e.* a vehicle’s speed and a steering wheel angle. Our controller uses a visual (spatial) attention mechanism [37], which visualizes controller’s internal state by highlighting image regions where the model fixates on for the network’s output.

5.2.1 Preprocessing

Following [37], we use raw images that are down-sampled to 10Hz and are resized to have input dimensionality as $90 \times 160 \times 3$. For better generalization, each image is then normalized by subtracting its mean from the raw pixels and dividing by its standard deviation. Following Liu *et al.* [50], we marginally change its saturation, hue, and brightness for achieving robustness during a training phase.

5.2.2 Convolutional Feature Encoder

We utilize a convolutional neural network (CNN) to obtain a set of visually-descriptive latent vectors at time t , where each vector contains a high-level visual description in certain input region. In this paper, we refer these latent vectors to as a convolutional feature cube X_t . By feeding an image through the model at each time t , we collect a X_t of size $w \times h \times d$. Note that X_t has l ($=w \times h$) (spatially) different vectors, each of which is a d -dimensional feature slice corresponding to a certain input region. Choosing a subset of these vectors will allow us to focus selectively on different parts of images (*i.e.* attention). Formally, $X_t = \{x_{t,1}, x_{t,2}, \dots, x_{t,l}\}$, where $x_{t,i} \in R^d$ for $i \in \{1, 2, \dots, l\}$.

5.2.3 Advice Encoder

Our advice encoder takes a variable-length advice and yields a latent vector, which then feeds to the controller LSTM (called Control LSTM). Our advice-taking driving model needs to understand the end-users utterance and to ground it into the vehicle controller. We assume that advice will often be given offline, or at the beginning of a ride, *e.g.* “look out for pedestrians” or “drive gently (occupant gets carsick)”. Thus, advice encoding will be prepared ahead of the controller generates control commands.

We train our advice encoder to deal with both types of advice (*i.e.* the goal-oriented and the stimulus-driven advice) without any input-level separation. We use a LSTM (called Advice LSTM, which is different from Control LSTM) to encode an input sentence (*i.e.* human-to-vehicle advice) and to yield a fixed-size latent vector, which is common practice in sequence-to-sequence models. Inspired by the knowledge from the Visual Question Answering (VQA) task, we follow the work by Park *et al.* [60] and use an element-wise multiplication to combine the latent vector from our advice encoder and the visual feature vector from our visual encoder.

Formally, our advice LSTM yields a d -dimensional latent vector $u \in \mathcal{R}^d$. By combining this vector with the visual feature $x_{t,i}$ using element-wise multiplication, we obtain a feature vector $z_{t,i} = x_{t,i} \odot u$, which is then fed into vehicle controller. Note that vehicle controller takes a new image at every time t (thus, update $x_{t,i}$) but the latent vector u remains the same during a period of the event.

Note that we focus on two forms of advice: (i) stimulus-driven and (ii) goal-oriented advice. The former advice (*e.g.* “watch out a pedestrian”) about perception can be grounded into a context $y_{t,i}$ via attention maps. We, however, argue that attention maps may not be sufficient to ground the latter advice (*e.g.* “go straight”), which needs a more direct influence to the controller via an additional element-wise multiplication.

5.2.3.1 Synthetic Token

We use a synthetic token `<none>` to indicate unavailable advice input. Since users will not be aware of the full state of the vehicle (they are not driving), the controller should mainly be in charge. Thus, we augment replicate of the dataset that however has a `<none>` token as the advice input, which exposes the model to events that do not have advice as an input.

5.2.4 Interpretable Vehicle Controller

Providing a controller’s internal state is important for advisable systems since it will be used as a ground or an acknowledgment of their advice taken. To this end, we utilize the attention-based driving model [37] that provides the controller’s internal state by visualizing attention maps – *i.e.* where the model visually fixates on image regions that are relevant to the decision.

5.2.4.1 Visual Attention

Visual attention provides introspective explanations by filtering out non-salient image regions, while image areas inside the attended region have potential causal effect on the output. The goal of visual attention mechanism is to find a context $Y_t = \{y_{t,1}, y_{t,2}, \dots, y_{t,l}\}$ by minimizing a loss function, where $y_{t,i} = \pi(\alpha_{t,i}, x_{t,i}) = \alpha_{t,i}x_{t,i}$ for $i = \{1, 2, \dots, l\}$. Note that a scalar attention weight value $\alpha_{t,i}$ in $[0, 1]$ is associated with a certain grid of input image in such that $\sum_i \alpha_{t,i} = 1$. We use a multi-layer perceptron f_{attn} to generate $\alpha_{t,i}$, *i.e.* $\alpha_{t,i} = f_{\text{attn}}(x_{t,i}, h_{t-1})$ conditioned on the previous hidden state h_{t-1} , and the current feature vector $x_{t,i}$. Softmax regression function is then used to obtain the final attention weight.

5.2.4.2 Outputs

The outputs of our model are two continuous values of a speed $\hat{v}(t)$ and a steering wheel angle $\hat{s}(t)$. We utilize additional hidden layers f_v and f_s , each of which are conditioned on the current hidden state h_t (of the control LSTM) and a context vector c_t . We generate the

context vector by utilizing a function f_{concat} , which concatenates $\{c_{t,i}\}_{i=1}^l = \{y_{t,i} \odot u\}_{i=1}^l$ to output 1-D vector c_t .

5.2.4.3 Internalizing Advice

Stimulus-driven advice provides rich messages about visual salencies (*e.g.* traffic lights, pedestrians, signs, etc) that the vehicle controller should typically see these objects while driving. Thus, to internalize such advice, we argue that the driving model must attend to those areas even when such advice is not available. We add a loss term, *i.e.* the Kullback-Leibler divergence (D_{KL}), between two attention maps (*i.e.* generated with and without advice) to make the driving model refer to the same salient objects:

$$\mathcal{L}_a = \lambda_a \sum_t D_{KL}(\alpha_t^w || \alpha_t^{wo}) = \lambda_a \sum_t \sum_{i=1}^l \alpha_{t,i}^w (\log \frac{\alpha_{t,i}^w}{\alpha_{t,i}^{wo}}) \quad (5.1)$$

where α^w and α^{wo} are the attention maps generated by the vehicle controller with and without advice given, respectively. We use a hyperparameter λ_a to control the strength of the regularization term.

5.2.4.4 Loss function

Existing models have been trained mainly by minimizing the proportional control error term (*i.e.* the difference between human-demonstrated and predicted). However, these systems are prone to suffer from two major issues. (i) Oscillation of control predictions – its prediction has repeated variation against a target value. (ii) Variations in task performance between drivers.

Inspired by proportional-integral-derivative (PID) controller [64], we use the following loss function, which consists of three terms: (i) \mathcal{L}_p , which is proportional to the error (*i.e.* $|e_v(t)| + |e_s(t)|$), where we use the error terms $e_v(t) = v(t) - \hat{v}(t)$ and $e_s(t) = s(t) - \hat{s}(t)$. (ii) \mathcal{L}_d , which is proportional to the derivative of the error (*i.e.* $\frac{d}{dt}e_v(t)$ and $\frac{d}{dt}e_s(t)$), and (iii) \mathcal{L}_i , which is proportional to the integral of the error, which we use the difference in the vehicle’s future course $\theta(t)$ – a cardinal direction in which a vehicle is to be steered. With the bicycle model assumption [64] - which assumes that left and right front wheels are represented by one front wheel, we can approximate a steering wheel angle $s_t \approx L/r$, where L is the length of wheelbase and r is the radius of the vehicle’s path. Then, we can approximate the vehicle’s course $\theta(t) \approx \frac{v(t)\tau}{r} \approx s(t)v(t)$ after the unit time $\tau = 1$. Thus, we use the following loss function \mathcal{L} :

$$\mathcal{L} = \mathcal{L}_a + \frac{1}{T} \sum_{t=0}^{T-1} \left[\underbrace{(|e_v(t)| + |e_s(t)|)}_{\mathcal{L}_p} + \underbrace{\lambda_i |\theta(t) - \hat{\theta}(t)|}_{\mathcal{L}_i} + \underbrace{\lambda_d (|\frac{d}{dt}e_v(t)|^2 + |\frac{d}{dt}e_s(t)|^2)}_{\mathcal{L}_d} \right] \quad (5.2)$$

Table 5.1: Examples of processing annotated descriptions.

Type	Step	Textual Annotation
<i>action</i> desc.	Initial annotation	The driver <i>went</i> straight and <i>stopped</i> at an intersection.
	→ Present tense	The driver <i>goes</i> straight and <i>stops</i> at an intersection.
	→ Imperative	<i>Go</i> straight and <i>stop</i> at an intersection.
<i>attention</i> desc.	Initial Annotation	There <i>was</i> a pedestrian pushing a stroller through the crosswalk.
	→ Present tense	There <i>is</i> a pedestrian pushing a stroller through the crosswalk.

where T is the number of timesteps. We use hyperparameters λ_d and λ_i to control the strength of the terms.

5.3 Honda Research Institute-Advice Dataset (HAD)

In order to evaluate the advisable driving model, we have collected Honda Research Institute-Advice Dataset (HAD). In this section, we describe our dataset in terms of the driving videos used to collect human-annotated textual advice, our annotation process, and analysis of the advice collected.

5.3.1 Driving Videos and Vehicle Control Commands

We use 5,675 video clips (over 32 hours), each of which is on average 20 seconds in length. Each video contains around 1-2 driving activities, *e.g.* passing through an intersection, lane change, stopping, etc. These videos are randomly collected from a large-scale driving video dataset called HDD [65]. This dataset contains camera videos – which are captured by a single front-view camera mounted in a fixed position on the roof top of the vehicle. These videos are mostly captured during urban driving near the San Francisco Bay Area, which contain the typical driver’s activities (*i.e.* turning, merging, lane following, etc) on various road types (*i.e.* highway, residential roads with and without lane markings, etc). Alongside the video data, the dataset provides a set of time-stamped controller area network (CAN) bus records, which contain human driver control inputs (*i.e.* steering wheel angle).

5.3.2 Annotations

We provide a 20 seconds driving video and ask a human annotator to describe, from a point of view of a driving instructor, what the driver is doing (*action* description for goal-oriented advice) and what the driver should pay attention (*attention* description for stimulus-

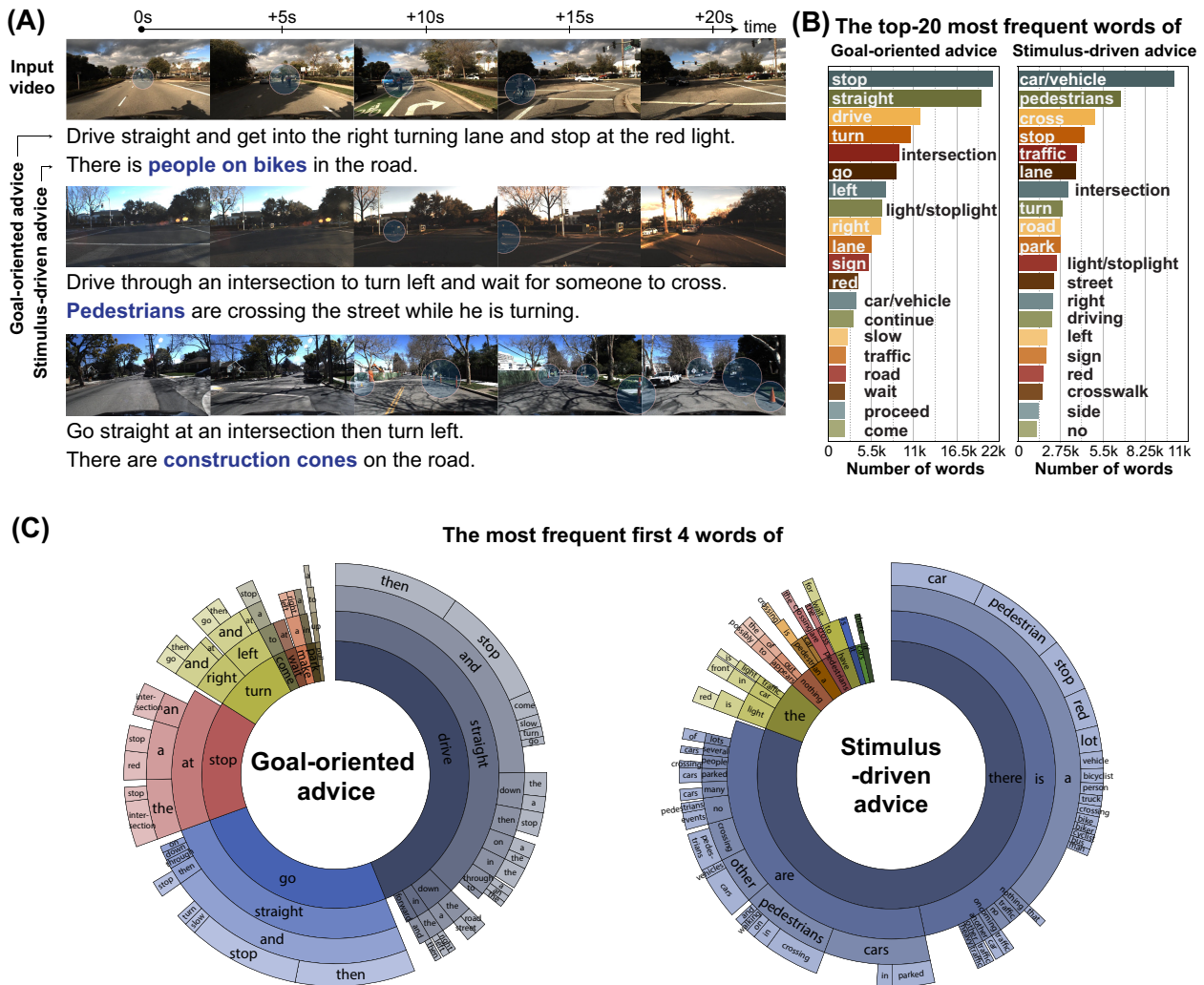


Figure 5.3: (A) Examples of input images, which are sampled at every 5 seconds. We also provide examples of the goal-oriented advice and the stimulus-driven advice, which are collected from human annotators followed by a post-processing. We highlight a visual cue (e.g. pedestrians), which are mentioned in advice, with a blue circle on the images. (B) The counts of top-20 most frequent words used in both types of advice. (C) The distribution of advice by their first four words for both types. The ordering of the words starts from the center, and the length of the arc indicates the proportion of the number of advice containing the word. Note that we remove areas where the number of words is too small (less than 1%) to show.

driven advice). We require that the annotators enter the action description and attention description separately, for example, “The driver crossed lanes from right to left lane” and

“There was construction happening on the road”, respectively. Each video clip has 4-5 action descriptions (25,549 in total) and 3-4 attention descriptions (20,080 in total). We then change the descriptions into the present tense (*e.g.* “The driver crosses lanes from right to left lane”). Especially for action descriptions, we change them to imperative sentences (*e.g.* “Cross lanes from right to left lane”), which are used to offer advice. To ensure the quality of the collected descriptions, we ask another human annotator to proofread the descriptions/advice to correct typographical errors and mistakes in grammar and spelling. In our analysis of annotations, we found that this two-stage annotation is helpful for the annotator to understand the task and perform better. In Figure 5.3 (A), we provide examples of two types of advice collected along with dashboard camera images (sampled at every 5 seconds).

5.3.3 Dataset Characteristics

Figure 5.3 (B) shows word counts of the top-20 most frequent words used in the goal-oriented advice and the stimulus-driven advice, respectively. Note that we exclude prepositions, conjunctions, and definite and indefinite articles. Most common goal-oriented advice is related to changes in speed (*i.e.* stop, slow), driving (*i.e.* drive, straight, go, etc), and turning (*i.e.* left, right, turns). Many also include a list of concepts relevant to a driving, such as traffic light/sign, lane, intersection. The stimulus-driven advice covers a diverse list of concepts relevant to the driving scenario, such as the state of traffic/lane, traffic light/sign, pedestrians crossing the street, passing other parked/crossing cars, etc. Although less frequent, some contain references to different types of vehicle (*i.e.* bus, truck, bike, van, etc), road bumps, and weather conditions.

5.4 Experiments

5.4.1 Training and Evaluation Details

We use a single LSTM layer for all the components of our framework. Our model is trained end-to-end using random initialization (*i.e.* no pre-trained weights). For training, we use Adam optimization algorithm [40] and dropout [73] of 0.5 at hidden state connections and Xavier initialization [23]. Our model takes 1-3 days (depending on types of CNN used) to train and can process over 100 frames on average per second on a single Titan Xp GPU. We use two mathematical criteria (the statistics of absolute errors and the correlation distance) to quantitatively evaluate their performance by comparing with ground-truth human-demonstrated control commands.

5.4.2 Advisable vs. Non-advisable models

As shown in Table 5.2, we first compare the vehicle control prediction performance to see our advice-taking driving model can outperform other existing driving models that do not

Table 5.2: In order to see the effectiveness of our advice-taking model, we compare the vehicle control prediction performance with other two existing models, which do not take advice (the first two rows). For a fair comparison, we use the identical 5-layer base CNN [9]. We also share the same input and output layers trained with the same loss function (*i.e.* the proportional error \mathcal{L}_p alone) used in [37]. We compare the control prediction performance in terms of three different sets of advice (*i.e.* the goal-oriented advice (ADV_G) only, the stimulus-driven advice (ADV_S) only, and both). For evaluation, we use the mean of correlation distances (Corr) and the median of absolute errors as well as the 1st (Q1) and 3rd (Q3) quartiles.

Type	Model	Advice input		Speed (km/h)		Steering Wheel Angle (deg)	
		Training	Testing	Median [Q1, Q3]	Corr	Median [Q1, Q3]	Corr
<i>Non-advisable</i>	ConvNet+FF (feed forward network) [9]	-	-	6.88 [3.13, 13.1]	.597	4.63 [1.80, 12.4]	.366
	ConvNet+LSTM+Attention [37] (baseline)	-	-	3.98 [1.76, 8.10]	.763	3.92 [1.54, 10.1]	.469
<i>Advisable</i>	CNN+LSTM+Attention+Advice (<i>Ours</i>)	ADV_G only	ADV_G	4.25 [1.86, 8.46]	.743	3.53 [1.37, 8.83]	.516
	CNN+LSTM+Attention+Advice (<i>Ours</i>)	ADV_S only	ADV_S	3.28 [1.47, 6.46]	.782	3.78 [1.45, 9.93]	.484
	CNN+LSTM+Attention+Advice (<i>Ours</i>)	$\text{ADV}_G+\text{ADV}_S$	ADV_G	3.78 [1.67, 7.50]	.763	3.54 [1.36, 9.21]	.512
	CNN+LSTM+Attention+Advice (<i>Ours</i>)	$\text{ADV}_G+\text{ADV}_S$	ADV_S	3.78 [1.68, 7.46]	.763	3.78 [1.41, 9.51]	.511

take advice. To this end, we implemented two other existing models, *i.e.* (1) CNN+FF (Feed forward network) [9] and (2) CNN+LSTM+Attention [37]. For a fair comparison, all models used the identical 5-layer CNN [9] as the convolutional (visual) feature encoder trained by minimizing the loss term \mathcal{L}_p only (same as used in [37]. See Equation 5.2). This visual encoder produces a $12 \times 20 \times 64$ -dimensional feature cube from the last convolutional layer. In the later section, we will also explore further potential performance improvements with more expressive neural networks over this base CNN configuration.

In Table 5.2, we report a summary of our experiments validating the quantitative effectiveness of our advice-taking approach. Comparing with the non-advisable models (rows 1-2), our advisable models all gave better scores for vehicle control prediction. As we will see in the next section, we observe that our advisable driving model focuses more on driving-related objects (whether provided as advice or not) than others that do not take advice during training and testing phases. For example, in Figure 5.4 and 5.5, our advisable model pays more attention to pedestrians crossing, a car pulling out, and construction cones. More importantly, advice like “*stop at a stop sign*” or “*there is a person with a stroller crossing the crosswalk*” may reflect typical links between visual causes and actions of human driver behavior. The data suggests that taking advice in controller helps imitate more closely human driver behaviors. Biasing the controller by taking advice improves the plausibility of its output from a human perspective.

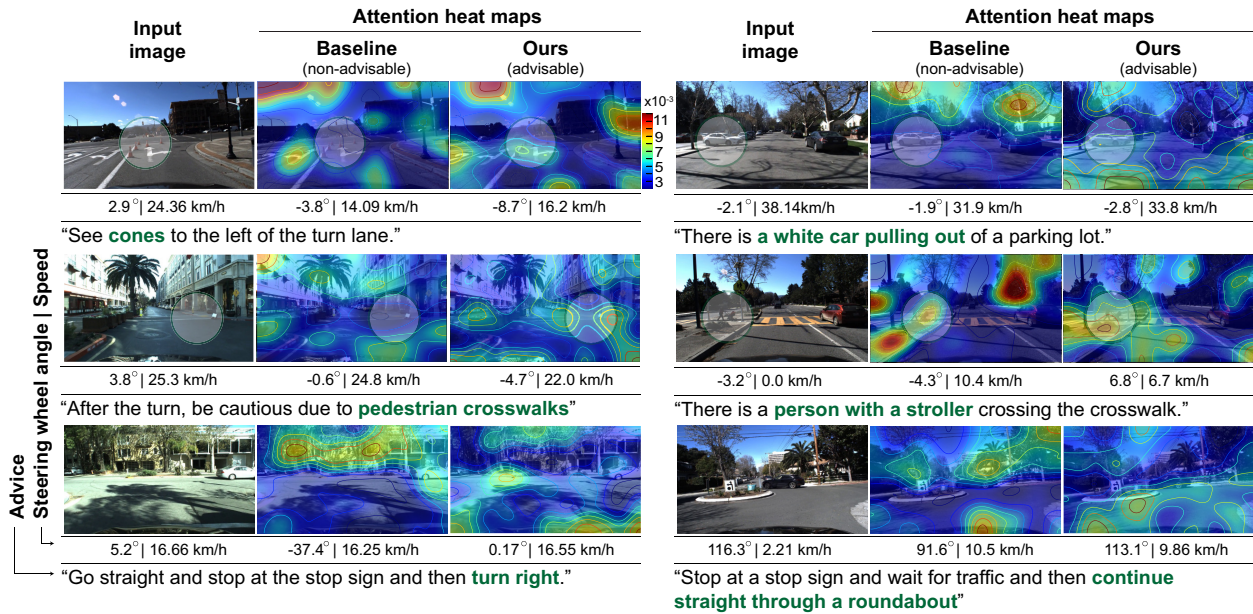


Figure 5.4: Attention heat maps comparison. We provide input raw images and attention heat maps generated by the existing attention-based driving model [37] (Baseline column), and our model trained with all types of advice together (Ours column). We highlight key object-centric words (as appropriate row 1 & 2), *e.g.* cones and a white car pulling out, in green as well as corresponding salient objects in a green circle overlaid on images.

5.4.3 Types of Advice Matter

We further examine the performance comparison with two different types of advice: the goal-oriented advice (*e.g.* “stop at the intersection”) and the stimulus-driven advice (*e.g.* “there is a pedestrian crossing”). In Table 5.2 (rows 3-4), we report vehicle control prediction accuracy when each of which types of advice is given to the model. In our analysis, the goal-oriented advice provides better control accuracy for predicting steering wheel angle commands. This is mainly due to the fact that the goal-oriented advice conveys the more direct messages, which may include navigational command on how the vehicle behaves (*e.g.* go/stop and turn). The stimulus-driven advice, which conveys rich messages about visual saliencies (*e.g.* red light, stop sign, and intersection), provides better predicting accuracy for vehicle’s speed prediction.

5.4.4 Qualitative Analysis of Attention Maps

As shown in Figure 5.4, we qualitatively compared with our baseline by visualizing attention heat maps - the highlighted image region has a potential influence on the network’s outputs.

Table 5.3: We propose an advice internalization technique – which minimizes the difference between two attention maps (generated *with* and *without* advice inputs) and thus makes the driving model refer to the same salient objects. Note that we use a synthetic token `<none>` to indicate when advice inputs are not available. We used λ_a as 50 (by the grid-search method).

Model	Advice input		Speed (km/h)	Steering Wheel Angle (deg)		
	Training	Testing	Median [Q1, Q3] Corr	Median [Q1, Q3]	Corr	
<i>no</i> advice internalization	ADV _S	<None>	3.55 [1.58, 7.12] .777	4.01 [1.59, 10.1]		.479
<i>w/</i> advice internalization	ADV _S	<None>	3.36 [1.51, 6.62] .784	3.96 [1.55, 10.0]		.480

While all models see driving-related common visual cues (*i.e.* lane markings), we observed that our advice-taking model focuses more on both advice-related cues (*i.e.* pedestrian crossing, construction cones, a car pulling out, etc) or visual objects relevant to the certain driving scenario (*i.e.* vehicles, crosswalk, pedestrians, etc).

5.4.5 Internalizing Advice Taken

Users will not usually be aware of the full state of the vehicle (they are not driving), the vehicle controller should mostly be in charge and the human-to-vehicle advice might occasionally be unavailable. As summarized in Table 5.3, we further examine the performance comparison with no advice available (we use a synthetic token `<none>` to indicate unavailable advice input) in a testing time. Interestingly, we observe that (i) the performance of a model trained with the stimulus-driven advice is not degraded much whenever advice inputs are not available in testing (its control performance is still better than other non-advisable approaches), (ii) our advice internalization technique (see Equation 5.1) further improves the control performance toward those having advice inputs.

In Figure 5.5, we further examine the effect of advice internalization by visualizing attention heat maps. We first visualize attention maps generated with no advice provided (*i.e.* using a `<none>` token, see middle row). Then, we visualize the attention map changes when the model takes ground-truth advice as an input (see bottom row). Our result reveals that our model is still able to see driving-related visual cues (*i.e.* traffic lights or lanes), whereas advice inputs can bias the model to refer to objects, which is related to the advice given.

5.4.6 Visual Encoder Comparison

We further examine variants of our proposed model using four different widely-used visual feature encoders. We used the output of intermediate layers from Bojarski *et al.* [9], Inception

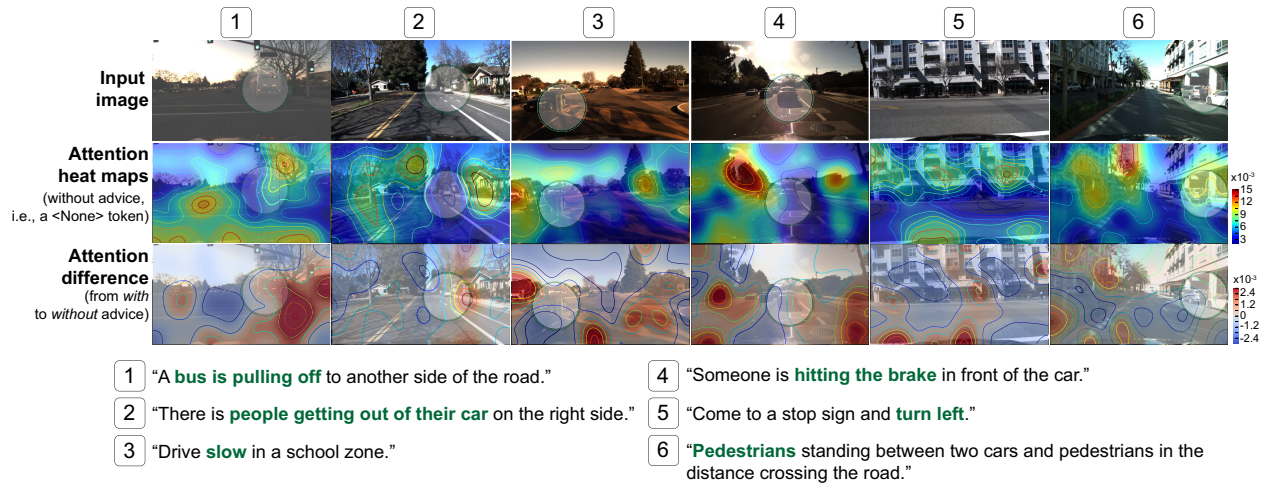


Figure 5.5: We compared attention heat maps generated *with* and *without* advice as an input in testing time. We visualize raw input images with salient objects marked by a green circle, *e.g.* a bus pulling off, which is mentioned by an advice input (1st row). The provided advice (1-6) is provided at the bottom of the figure. We visualize attention heat maps from our trained model but with a synthetic token `<none>` (*i.e.* without advice, 2nd row). Attention map differences between those with and without advice (3rd row), where red parts indicate where the model (with advice) pays more attention.

v3 [75], MobileNet [31], and Inception-ResNet-v2 [74]. We trained all models in an end-to-end manner using random initialization, and we used both types of advice as an input in the training and testing phases (averaged scores are reported). As reported in Table 5.4, the result reveals that control prediction accuracy can be generally expected to improve

Table 5.4: We compared the vehicle control prediction performance with four different visual encoders. Except for the visual encoder part, we use the same training strategy.

CNN base	Speed (km/h)		Steering Wheel Angle (deg)	
	Median [Q1, Q3]	Corr	Median [Q1, Q3]	Corr
MobileNet [31]	3.93 [1.73, 7.80]	.753	4.20 [1.65, 10.7]	.463
Bojarski <i>et al.</i> [9]	3.78 [1.68, 7.49]	.763	3.58 [1.39, 9.34]	.512
Inception v3 [75]	2.89 [1.31, 5.59]	.795	3.47 [1.34, 8.76]	.525
Inception-ResNet-v2 [74]	2.93 [1.33, 5.63]	.796	3.54 [1.36, 9.19]	.491

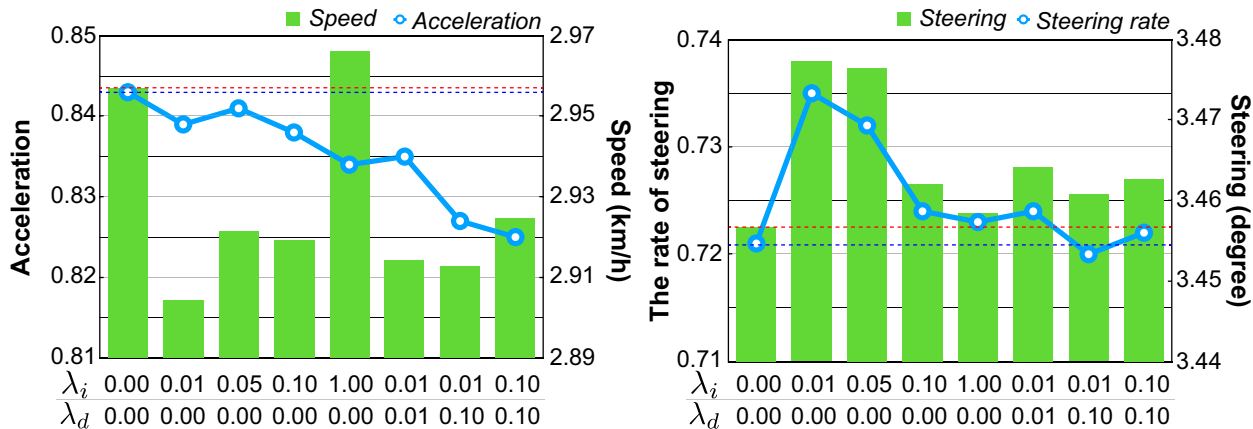


Figure 5.6: Control performance comparison with different sets of hyperparameters (λ_d , λ_i) (see Equation 5.2). Along with proportional prediction errors (green bar), we also visualize derivative errors (blue line). We report the median value of absolute error.

when using a deeper CNN architecture, which learns more expressive visual features. Visual features from the Inception v3-based architecture lead the best performance improvement against other three architectures.

5.4.7 Effect of Regularization

We explored the loss function \mathcal{L} , which contains three terms – \mathcal{L}_p (proportional error), \mathcal{L}_d (derivative error), and \mathcal{L}_i (integral error). We use two hyperparameters λ_d and λ_i to control the strength of the corresponding terms. Figure 5.6 shows control command prediction errors with different combinations of hyperparameters in terms of the median value of absolute errors. We also visualize the error of the acceleration (the derivative of speed) and the steering angle rate (the derivative of steering angle command). The impact of adding these loss terms is dominant in the prediction of speed, whereas the performance in steering is slightly degraded. We obtained marginal improvement by adding integral loss term (λ_i) in speed predictions, while derivative errors are reduced by adding derivative loss term (λ_d).

5.5 Related Work

5.5.1 End-to-End Learning for Self-driving Vehicles

Recent successes [9, 87] suggest that a driving policy can be successfully learned by neural networks as a supervised learner over observation (*i.e.* raw images)-action (*i.e.* steering) pairs collected from human demonstration. Bojarski *et al.* [9] trained a deep neural network to

map a dashcam image to steering controls, while Xu *et al.* [87] explored a stateful model using a dilated deep neural network and recurrent neural network so as to predict a vehicle’s discretized future motion given input images. Other variants of deep neural architecture have been explored [22, 16].

Explainability of deep neural networks has become a growing field in computer vision and machine learning communities. Kim *et al.* [37] utilized a recurrent attention model followed by a causal filtering that removes spurious attention blobs and visualizes causal attention maps. We start our work with this attention-based driving model. Attention model visualizes controller’s internal state by visualizing attention maps, which end-users may use as a ground and an acknowledgment of their advice. Other approaches [10, 39] can also be applied to provide richer explanations, but we leave it for future work.

5.5.2 Advice-taking models

Recognition of the value of advice-taking has a long history in AI community [54], but a few attempts have been made to exploit textual advice. Several approaches have been proposed to translate the natural language advice in formal semantic representations, which is then used to bias actions for simulated soccer task [42], mobile manipulation tasks [56, 55, 76], and a navigation task [4]. These approaches consider high-level action sequences to be given in the task space of the agent. Instead, we consider the visual imitation learning setting, where the model has its own perceptual primitives that are trained by observing third-person demonstration and types of advice. Recent work suggests that incorporation of natural language human feedback can improve a text-based QA agent [47, 85] and image captioning task [49]. Despite their potential, there are various challenges (*e.g.* safety and liability) with collecting human feedback on the actions taken by self-driving cars. Other notable approaches (in the reinforcement learning setting) may include the work by Tung *et al.* [77] that learns a visual reward detector conditioned on natural language action descriptions, which is then used to train agents. To our best knowledge, ours is the first attempt to take human-to-vehicle advice in natural language and ground it in a real-time deep vehicle controller.

Chapter 6

Advisable Learning for Self-driving Vehicles by Internalizing Observation-to-Action Rules

6.1 Problem Statement

Autonomous driving control has made dramatic progress in the last several years. The proposed vehicle controllers use a variety of approaches; recent efforts [9] suggest that deep neural networks can be effectively applied to the controllers in an end-to-end manner. These models, however, are known to be opaque. One way to simplify and expose the underlying reasoning, is via a situation-specific dependence on visible objects in the scene, i.e. by only attending to image areas that are causally linked to the driver’s actions [37]. However, the resulting attention maps are not always compelling or human interpretable. Another option is to verbalize the autonomous vehicle’s behaviour with natural language [39], Figure 6.2 (B). The resulting textual explanations are human understandable, but tend to be rather “shallow”, as they report the more common objects over the less common ones, which may be more important (*e.g.* construction cones). Both approaches fall short of demonstrating causal behaviour akin to a typical human driver.

To address this issue, [38] augment an imitation learning dataset with instantaneous human advice (*e.g.* “there is a pedestrian ahead”, or “turn left”), see Figure 6.2 (A). They show that providing such inputs helps more closely imitate a human driver’s behavior. While promising, this method requires ground-truth human inputs at test time.

Humans learn to drive not only from practice and demonstration, but also from theory, *e.g.* by studying the rules. We advocate for a more principled way of integrating human advice during learning. We assume that at training time, human advice is available in the form of observation-action rules (*e.g.* “if the road is wet, slow down”). Incorporating such rules could help driving models learn more human-like behavior, see Figure 6.1.

A key requirement of an advisable driving model is its explainability – exposing the con-

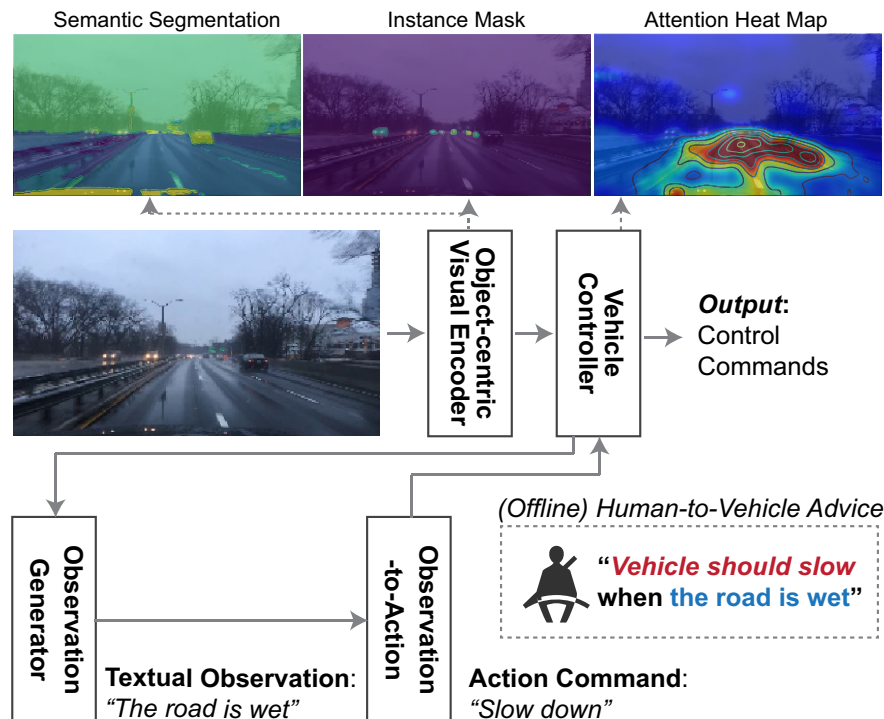


Figure 6.1: Our model consists of four main parts: (1) an object-centric visual encoder built upon a semantic segmentation model, (2) an observation generator, which generates textual observation about the scenes (“The road is wet”), (3) an observation-to-action module, which maps a visual scene description to a (high-level) action command (“Slow down”), and (4) a vehicle controller conditioned on the generated action command.

troller’s internal state is important for a user as an acknowledgement that the system is following advice. As mentioned earlier, visual attention is often used in recent explainable models [37, 39]. These models generate spatial attention maps, which are then displayed over the original images. However, such attention maps are coarse and have limited interpretability. They usually have a low spatial resolution (as the last convolutional layer) and are upsampled with a 2D Gaussian kernel. This blurs out the details and makes it difficult to determine what the model actually attends to. We advocate for using a richer representation, such as semantic segmentation, which provides pixel-wise prediction and delineates object boundaries in images. The output of the last convolutional layer retains information of the corresponding local image regions, which can be advantageous for obtaining more fine-grained attention maps. We thus propose to use semantic segmentation as our input representation, and tie the predicted attention maps to the output of the segmentation model. To further improve the quality of the attention maps, we also use an instance segmentation

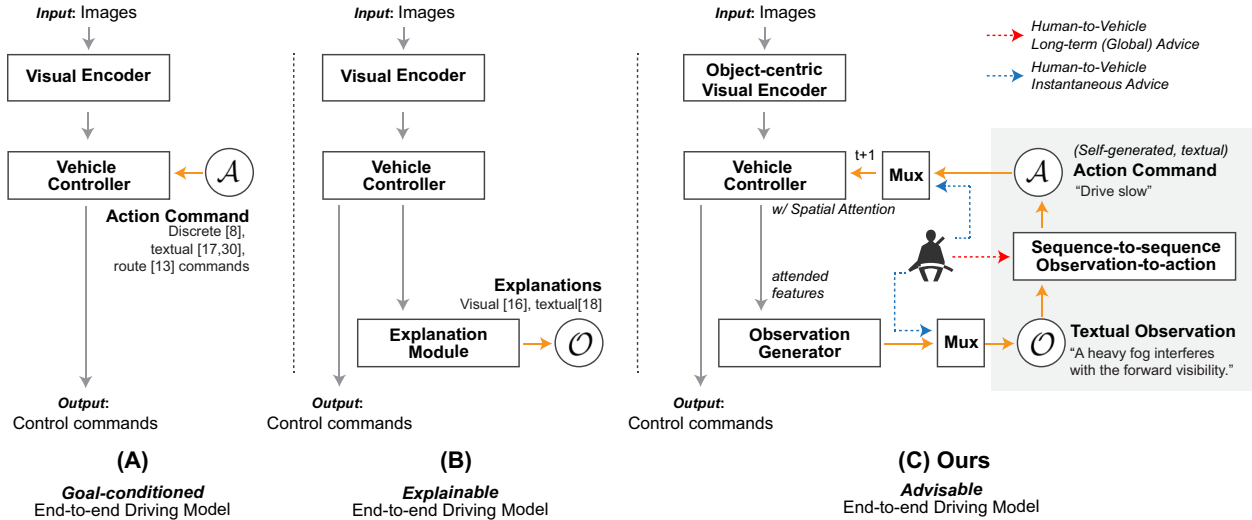


Figure 6.2: (A) Existing goal-conditioned end-to-end driving models that takes (as an input) discrete [18], natural language commands [38, 66], and intended navigational route [26]. (B) Existing explainable end-to-end driving models that transduce DNN states to natural language [39] or visual explanations [37]. (C) Combining two above-mentioned ideas, we can create “Advisable” driving model that takes human-to-vehicle advice in the form of observation-action rules. To incorporate such rules, our model involves a Sequence-to-Sequence Observation-to-Action module, which generates a soft condition-action rule that maps a textual observation to a high-level action command. For details see Section 6.2.

model, which allows us to distribute attention over individual objects.

Overall, we propose a novel self-driving model that is both advisable and explainable, see Figure 6.2 (C). Our model learns advice from human inputs which convey global rules that the user expects the vehicle to follow (*e.g.* “If a heavy fog interferes with your forward visibility, drive slowly”). We can also provide both visual explanations – by producing fine-grained attention maps, and textual explanations – by generating textual utterances (*e.g.* “the traffic light ahead turned red”, thus “the car stopped”). We ground both functionalities in our object-centric visual representation.

We evaluate our approach on the BDD-X dataset [39] and show that our model matches or outperforms prior work in control prediction and textual observation generation. Our attention maps, tied to the semantic segmentation, result in object-centric (and thus more interpretable) visualization of internal states. Our human evaluation in a simulated environment (Carla [20]) further shows that our advisable system can increase user trust.

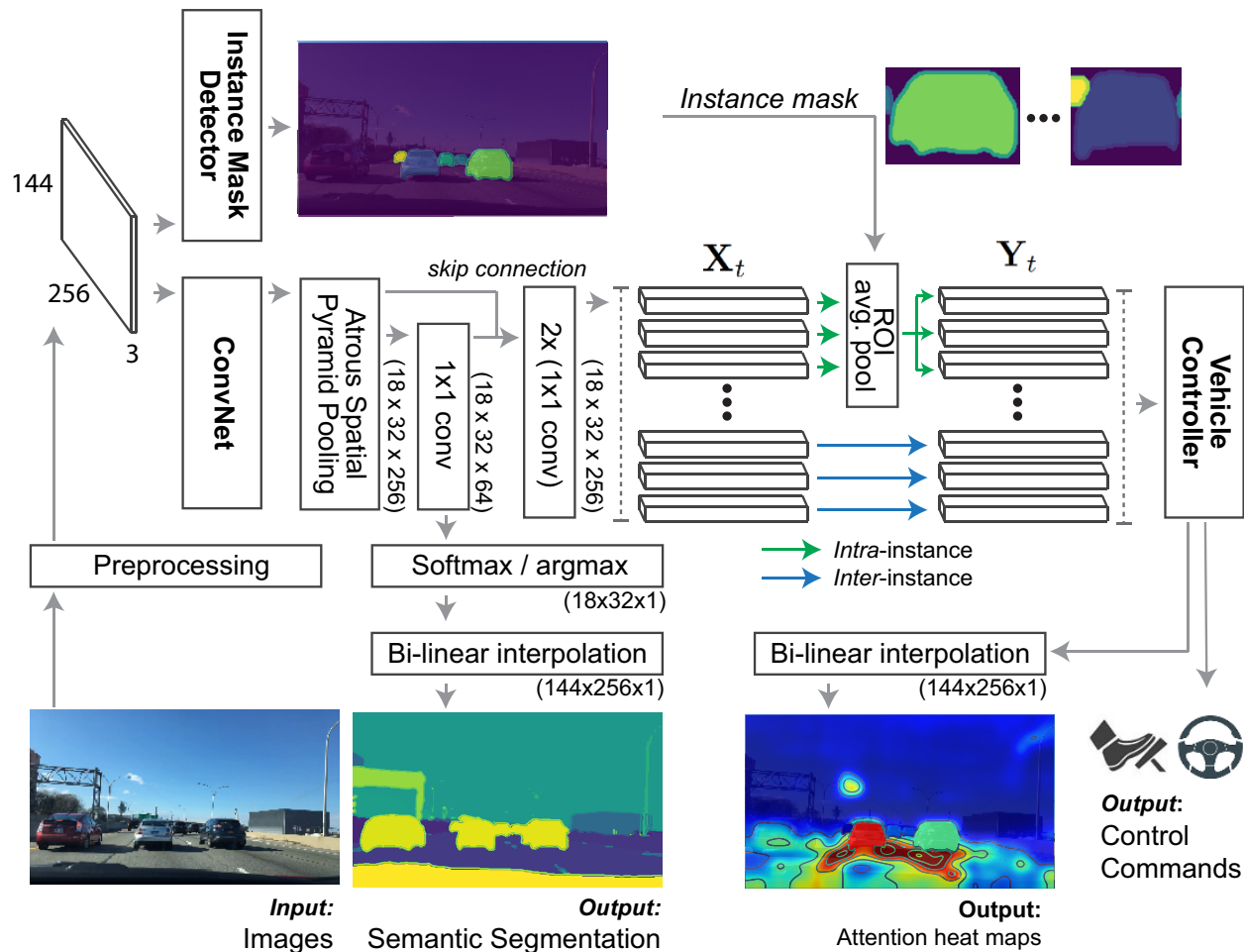


Figure 6.3: The detailed overview of our *Object-centric Visual Encoder* that is built upon an instance mask detector and a semantic segmentation model, both of which provide pixel-wise category predictions from images along with delineating the boundaries of object.

6.2 Advisable Learning

In this paper, we propose a novel driving model that is both explainable and advisable. Our model can provide the basis of its decision both by visualizing image regions that it attends to and by verbalizing the observations of what it sees (*e.g.* “it is snowing”). Our model is also advisable by incorporating general observation-action rules, which it is expected to follow. To this end, our model needs four main capabilities: (i) perceptual primitives to manipulate the vehicle’s behavior, (ii) the ability to control a vehicle conditioned on the determined actions, (iii) the ability to verbalize what is happening while driving, and (iv) the ability to

understand and respond to the observations with the corresponding high-level actions.

As shown in Figure 6.2 (C), our model includes four main components. Our *Object-centric Visual Encoder* extracts visual (semantic) representations through a ConvNet that is pretrained on the task of semantic segmentation (Section 6.2.1). The *Vehicle Controller* is trained to predict control commands conditioned on the high-level action commands (e.g. “stop at the crosswalk”) (Section 6.2.2). The *Observation Generator* produces variable-length textual observations about the scenes (e.g. “pedestrians are waiting to cross”) (Section 6.2.3). Finally, our *Sequence-to-Sequence Observation-to-Action* module generates soft condition-action rules that map visual scene descriptions (e.g. “it is snowing”) to high-level action commands (e.g. “maintain a slow speed”) (Section 6.2.4). Note that, our *Vehicle Controller* utilizes a visual (spatial) attention mechanism, which can highlight image regions the model fixates on for the network’s output. This attended feature is then fed into the *Observation Generator* for the final prediction.

6.2.1 Object-centric Visual Encoder

We use images that are down-sampled to 10Hz and are resized to have dimensionality $144 \times 256 \times 3$ by applying bi-linear interpolation. Each image is normalized by subtracting the global mean from the raw pixels and dividing by the global standard deviation [68], see Figure 6.3.

6.2.1.1 Segmentation as an Input Representation

Instead of training a ConvNet from scratch, we use a semantic segmentation model that is pre-trained on the Mapillary Vistas street-view scene understanding dataset [57]. Our front-end vision module is therefore trained to recognize pixel-wise category predictions from images along with delineating the boundaries of each object. Here, we use the DeepLab v3 model [14], a state-of-the-art network that uses atrous spatial pyramid pooling to robustly segment objects at multiple scales with various filters of different sampling rates and fields-of-view. We obtain a high-level visual representation of an input image at each time step t . This representation X_t (of size $18 \times 32 \times 256$) contains a set of 256-dimensional latent vectors over the spatial dimension, *i.e.* $X_t = \{x_{t,1}, x_{t,2}, \dots, x_{t,l}\}$, where $l (= w \times h)$ is the spatial dimension. Note, that the use of semantic segmentation as the internal representation of visual scenes is generally transferable between real-world and simulated setting.

6.2.1.2 Object-centric RoI Pooling

To further provide object-centric attention heat maps, which highlight more precise object regions, we use an instance detection model, MaskRCNN model [25], and tie the predicted instance masks to the feature X_t . Here, we use the MaskRCNN model [25] as a region proposal to obtain instance-level masks. Given the instance regions (RoIs), a position-sensitive RoI pooling layer is used to aggregate the latent vectors $x_{t,i}$ for $i = \{1, 2, \dots, l\}$

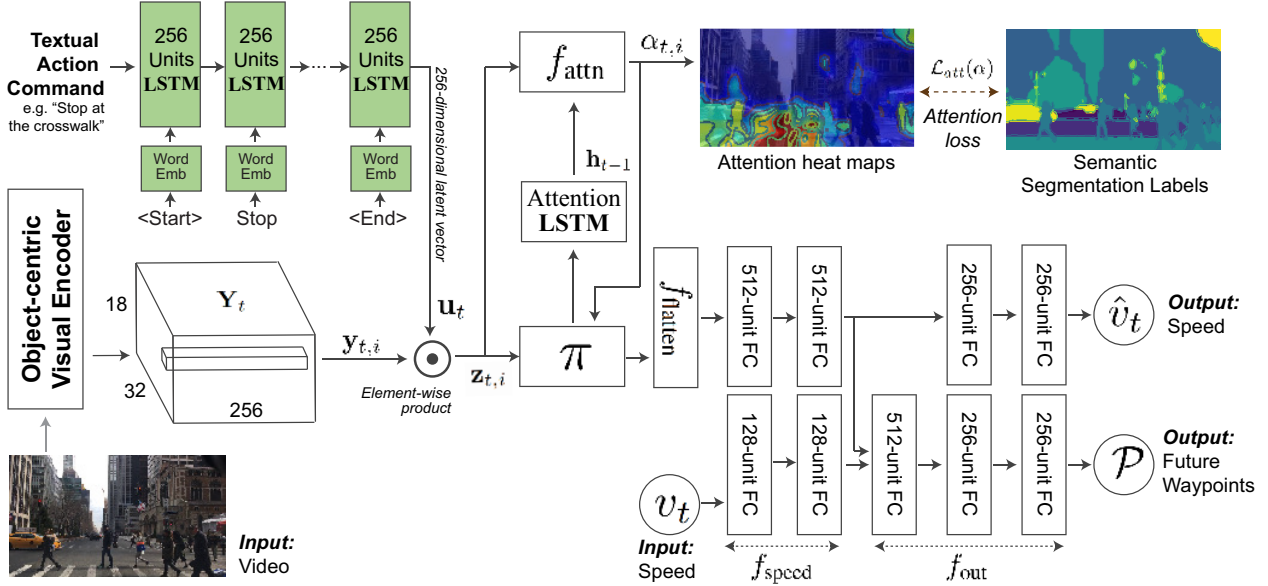


Figure 6.4: The detailed overview of our goal-conditioned *Vehicle Controller*. We take an action command in natural language as an input and ground it into the controller. Our model adopts spatial attention mechanism π , which guides where the controller looks. Conditioned on the attended feature and the current speed v_t , our model outputs future trajectory \mathcal{P} and speed \hat{v}_t . For details, see Section 6.2.2.

for each RoI. Note, that the pooled latent vector is then distributed equally to replace the original representations. This provides a subset of feature slices that share the same latent representation, and thus allows the model to equally attend to parts of RoI.

6.2.2 Goal-conditioned Vehicle Controller

6.2.2.1 Grounding Natural Language Action Command

Our vehicle controller is trained to predict control commands conditioned on the high-level action command (e.g. “maintains a slow speed”). We use a textual encoder that takes a variable-length textual command and grounds it into the vehicle controller. Following [38], we use an LSTM to encode an input word sequence and yield a 256-dimensional latent vector u_t . We combine this vector with the visual feature $y_{t,i}$ by an element-wise multiplication and obtain a feature vector $z_{t,i} = y_{t,i} \odot u_t$ for $i = \{1, 2, \dots, l\}$, which is then fed into visual attention module to generate attention maps. We provide detailed model architecture in Figure 6.4.

6.2.2.2 Visual Attention

Visual attention provides introspective (visual) explanations by filtering out non-salient image regions, while the attended regions have a potential causal effect on the output. The goal of visual attention mechanism is to find a context $C_t = \{c_{t,1}, c_{t,2}, \dots, c_{t,l}\}$ by minimizing a loss function, where $c_{t,i} = \pi(\alpha_{t,i}, z_{t,i}) = \alpha_{t,i} z_{t,i}$ for $i = \{1, 2, \dots, l\}$. Note that a scalar attention weight value $\alpha_{t,i}$ is in $[0, 1]$ such that $\sum_i \alpha_{t,i} = 1$. We use a multi-layer perceptron to compute these attention weights, *i.e.* $\alpha_{t,i} = f_{\text{attn}}(z_{t,i}, h_{t-1})$, conditioned on the previous hidden state h_{t-1} (of the *Attention LSTM*), and the current advice-grounded feature vector $z_{t,i}$. Softmax regression function is used to obtain the final normalized attention weight.

6.2.2.3 Output

Inspired by the prior work [7, 94], our vehicle controller predicts a future trajectory $\mathcal{P} = [p_{t,\Delta}, p_{t,2\Delta}, \dots, p_{t,N\Delta}]$ along with speed \hat{v}_t . Each point $p_{t,j\Delta}$ for $j = \{1, 2, \dots, N\}$ is characterized by its future longitudinal and latitudinal location after the time $j\Delta$. This trajectory can be converted into low-level driving control commands (*i.e.* steering, braking, and acceleration) by an optimizer within the constraints of the vehicle’s dynamics. Different types of vehicles may utilize different control outputs to achieve the same driving trajectory, which argues against training a network to directly output low-level steering and acceleration control.

To predict the future trajectory, we use additional hidden layers f_{out} conditioned on the latent representation C_t (from our *Advice-grounded Visual Attention*) and the current speed v_t , *i.e.* $\mathcal{P} = f_{\text{out}}([f_{\text{flatten}}(C_t), f_{\text{speed}}(v_t)])$, where f_{speed} denotes additional hidden layers to encode the speed in a high-dimensional latent space. f_{flatten} is a flattening function. We use Δ as 0.5 seconds and N as 6 (thus, we predict the future trajectory in the next 3 seconds).

6.2.2.4 Loss Function

We minimize the proportional control error (*i.e.* the difference between human-demonstrated and predicted) to train our future trajectory predictor.

$$\mathcal{L}_{ctl} = \frac{1}{NT} \sum_{t=1}^T \sum_{j=1}^N \lambda_j \|p_{t,j\Delta} - \hat{p}_{t,j\Delta}\|_2^2 + \lambda_0 \|v_t - \hat{v}_t\|_2^2 \quad (6.1)$$

where λ_j and λ_0 control the strength of each term, chosen to be inversely proportional to the global variance.

6.2.3 Textual Observation Generator

The main goal of our textual observation generator is to summarize visual observations, which need to be considered while driving, *e.g.* “there is a school bus with lights flashing” (this usually means the vehicle should pull over and remain stopped). Here, we use the term

“observation” to convey the notion of the model’s ability to actively perceive and register visual cues as being important for the vehicle controller. These observations can take a variety of forms with different levels of urgency and will be provided to the vehicle controller at every time step.

To generate such observations, our model involves a video-to-text module that takes a sequence of video frames and generates variable-length textual observations. In order to implement such a model, we start from the work of [39] that is originally designed to generate textual descriptions/explanations such as a pair “vehicle slows down” (description) and “because it is approaching an intersection and the light is red” (explanation). Unlike [39], where descriptions/explanations are predicted jointly as a single sequence (separated by a token), we focus on generating the later part (*i.e.* explanations) and treat them as observations. These observations are then used to predict the corresponding textual action commands, directing the vehicle to behave in a certain way (*e.g.* go, pass, turn), in Section 6.2.4.

We collect the latent vector \bar{c}_t over the past T timesteps by summing over the attended feature vectors $\{c_{t,i}\}$, *i.e.* $\bar{c}_t = \sum_{i=1}^l c_{t,i}$. We then apply a temporal attention mechanism with weights $\beta_{k,t}$ to those vectors at each time step k (of sentence generation), *i.e.* $g_k = \sum_{t=t_0-T+1}^{t_0} \beta_{k,t} \bar{c}_t$ where t_0 is the current timestep and $\sum_t \beta_{k,t} = 1$ with $\beta_{k,t}$ is in $[0, 1]$. The weight $\beta_{k,t}$ is computed by an attention model, which is similar to the spatial attention. This is common practice in sequence-to-sequence models and allows flexibility in output tokens relative to input samples [6].

Our decoder outputs per-word softmax probabilities. We minimize the following negative log-likelihood \mathcal{L}_{obs} :

$$\mathcal{L}_{obs} = - \sum_k \log p(o_k | o_{k-1}, g_k), \quad (6.2)$$

6.2.4 Sequence-to-Sequence Observation-to-Action

We want our model to incorporate natural language human-to-vehicle advice. Such advice is typically high-level, rather than low-level (where the vehicle controller operates). Recent work [38] proposed a model that allows short-term (or local) textual advice from passengers (*e.g.* “there are construction cones” or “slow down”). More generally, advice might take the form of condition-action rules. In this work, we focus on such long-term (or global) advice from humans (*e.g.* driving instructors).

We use a general encoder-decoder framework to incorporate the observation-action rules. Our LSTM encoder takes a *generated* variable-length textual observation (“there is a sharp turn ahead”) and yields a representative latent vector, while the decoder (another LSTM) outputs an action command sequence (“slow down”). The model is trained by minimizing the negative log-likelihood (similar to the observation generator). Our model is supervised by human inputs in the form of observation-action rules that the user expects the vehicle to follow. The predicted action commands are given as input to the vehicle controller.

6.2.4.1 Human-to-Vehicle Instantaneous Advice

We currently assume that advice is given offline, rather than during online human-vehicle interaction. Note, however, that our model can also take instantaneous human-to-vehicle advice. As shown in Figure 6.2 (C), we use two multiplexers to accept observational and navigational advice. The observational advice is mapped to an action command by our model.

6.2.4.2 Loss function

Our *Observation-to-Action* module outputs per-word softmax probabilities and we minimize the following negative log-likelihood $\mathcal{L}_{obs2act}$:

$$\mathcal{L}_{obs2act} = - \sum_m \log p(a_m | a_{m-1}, \{o_1, o_2, \dots, o_K\}) \quad (6.3)$$

We minimize the following loss function \mathcal{L} to train our entire driving model end-to-end, $\mathcal{L} = \mathcal{L}_{obs} + \mathcal{L}_{traj} + \mathcal{L}_{obs2act}$.

6.3 Experiments

6.3.1 Dataset

We use the Berkeley DeepDrive-eXplanation (BDD-X) dataset [39] to train and evaluate our proposed model. BDD-X contains front-view dashcam videos (≈ 40 seconds) collected during urban driving in the United States, covering all the typical driving events (lane following, intersection passing, turning, etc). Alongside the video data, the dataset provides corresponding time-stamped sensor measurements (*i.e.* IMU sensor measurements), which we use as a ground-truth control signal. For sensor logs that are not synchronized with the time-stamps of video data, we use the (linearly) interpolated measurements.

Moreover, the dataset provides textual (i) descriptions of the vehicle’s actions (*what* the driver is doing), and (ii) explanations for the driver’s actions (*why* the driver took that action from the point of view of a driving instructor), such as the pair: “the car slows down” and “because it is approaching an intersection”. This dataset is collected from human annotators in Amazon Mechanical Turk. We supervise our Textual Observation Generator with the textual explanations, while our Sequence-to-Sequence Observation-to-Action module is supervised with action descriptions (*i.e.* as navigational commands).

6.3.2 Training and Evaluation Details

Except for our object-centric visual encoder, we train other parts end-to-end using random initialization (*i.e.* no pre-trained weights). Unless otherwise stated, we use a single LSTM layer for all the components of our framework. For training, we use Adam optimization

Table 6.1: We report the vehicle control prediction performance for our approach and existing baselines. We compare the performance in terms of the median of average displacement errors (ADEs) as well as the 1st (Q1) and 3rd (Q3) quartiles (lower is better), *i.e.* Median [Q1, Q3]. [†]We used the following four discrete commands: lane following, turning, merging, and parking.

Model	ADE (in meters) ↓	
	<i>without</i> speed inputs	<i>with</i> speed inputs
A. CNN+FC [9]	2.36 [1.18, 4.61]	-
B. A + LSTM [87]	3.29 [1.49, 6.93]	-
C. B + Attention [37]	2.22 [1.17, 4.61]	-
D. A + Discrete commands (w/ branched output) [18] [†]	2.28 [0.89, 4.56]	1.35 [0.66, 2.76]
E. C + (natural language) commands [38]	2.11 [0.84, 4.86]	1.35 [0.42, 2.94]
F. D + Long-term (global) Advice	2.14 [0.93, 4.57]	0.81 [0.45, 1.61]
G. F + Object-centric Visual Encoder (ours)	1.97 [0.95, 4.39]	0.65 [0.46, 1.43]

algorithm [40] and Xavier initialization [23]. Our model takes 2 days to train on two NVIDIA Titan Xp GPUs. Our implementation is based on Tensorflow [1] and our code will be available upon publication. For evaluation, we use the average displacement error (ADE) to quantitatively evaluate control prediction performance by comparing to ground-truth human-demonstrated control commands. To evaluate the textual utterances generated by our model, we use popular automatic metrics: BLEU [59], METEOR [43], CIDEr-D [81], and SPICE [3].

6.3.3 Driving Performance Evaluation

We report the vehicle control prediction performance for our model and a number of baselines to evaluate the ability to control a vehicle conditioned on the determined actions. We compare to end-to-end driving models, CNN + FC [9], CNN + FC + LSTM [87], and CNN + FC + LSTM + Attention [37] and goal-conditioned driving models that ground different types of goal: discrete commands [18], top-down view intended route [26], and natural language commands [38]. For a fair comparison, we use the same base CNN [18] in all cases except the model G, which uses our object-centric front-end visual encoder. All models have the same output layer and are trained by minimizing the same loss function. As explained in Section 6.2.2, our model uses the current vehicle’s speed as an input, but we

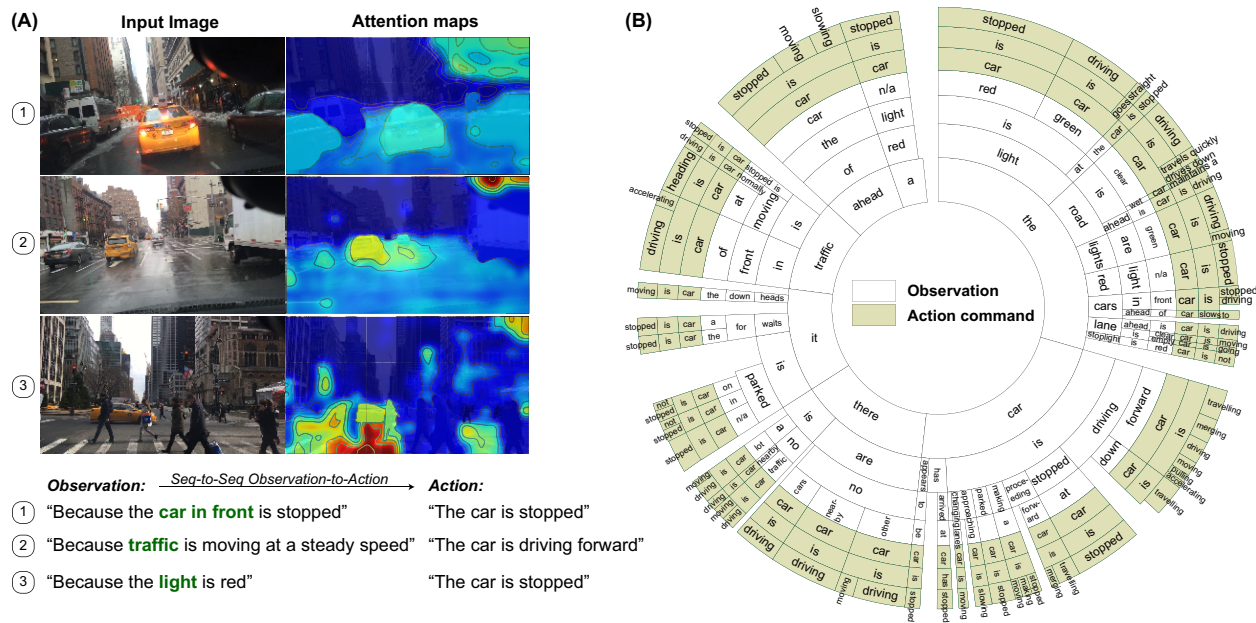


Figure 6.5: (A) Example observations and action commands generated by our model. We provide input raw images and attention maps of the vehicle controller. Our observation generator predicts a textual observation (*i.e.* “Because the car in front is stopped”), while our seq-to-seq observation-to-action module generates a textual action command (*i.e.* “The car is stopped”). Such a (high-level) action command is then grounded into the vehicle controller, which outputs control commands, *i.e.* future waypoints. (B) The distribution of our top-100 generated observation/action pairs by their first four or three words, respectively. The ordering of the words starts from the center, and the length of the arc indicates the proportion of the number of words. Note that we remove areas where the number of words is too small to show.

also evaluate models without speed inputs.

We report performance of the aforementioned models in Table 6.1 (lower is better). Consistent with the prior work, goal-conditioned models [18, 38] (D and E) generally provide better control prediction performance against the non-goal-conditioned models (top three rows). We observe that our model is further improved by adding long-term (or global) advising module (compare F vs. D). Our controller shares the attended feature with the Observation Generator, and thus encourages the model to attend to important visual cues (*e.g.* stop sign, traffic lights, pedestrians). Using our Object-centric Visual Encoder (instead of training a ConvNet from scratch) further improves control prediction performance (compare G vs. F).

Table 6.2: We report the quality of the generated textual observations (top) and action commands (bottom). We rely on standard automatic metrics: BLEU-4 [59], METEOR [43], CIDEr-D [81], and SPICE [3]. †: reported by [39]

Model	Textual Observation Generation			
	BLEU-4	METEOR	CIDEr-D	SPICE
S2VT [82]+SA+TA†	5.84	10.9	52.7	14.3
S2VT+SA+TA+WAA [39]†	7.28	12.2	69.5	17.5
Transformer-based Decoder [80]	9.90	13.6	70.1	17.5
Ours	12.2	16.0	104.4	21.7
Model	Textual Action Commands Generation			
	BLEU-4	METEOR	CIDEr-D	SPICE
S2VT [82]+SA+TA†	27.1	26.4	157.0	55.1
S2VT+SA+TA+WAA [39]†	32.3	29.2	215.8	59.6
Ours	40.0	33.3	310.7	60.9

6.3.4 Analysis of Observation-to-Action Module

In Figure 6.5 (A), we provide qualitative examples of the textual observations (*e.g.* “because the car in front is stopped”) and corresponding high-level action commands (“the car is stopped”) generated by our model. We also show the generated attention maps, which highlight image regions that have influenced the network’s outputs (*i.e.* both textual observations and control commands). Our model attends to relevant visual cues and generates corresponding textual sequences. The vehicle controller also looks at other driving-related objects, *e.g.* lane markings. Importantly, our model is able to learn observation-action rules, which are provided by humans at training time, and correctly reflect typical links between visual causes and actions of human driving behavior.

To see the distribution of the learned observation-to-action rules, we cluster observation/action pairs based on the first few words (*e.g.* the-light-is-red-car-is-stopped from the pair: “because the light is red” and “the car is stopped”) as shown in Figure 6.5 (B). Our model generates a variety of observation-to-action pairs, which are compatible with the human driver’s general knowledge. For example, the observation starts with “the road is wet” produces an action command starting with “the car maintains slow speed”.

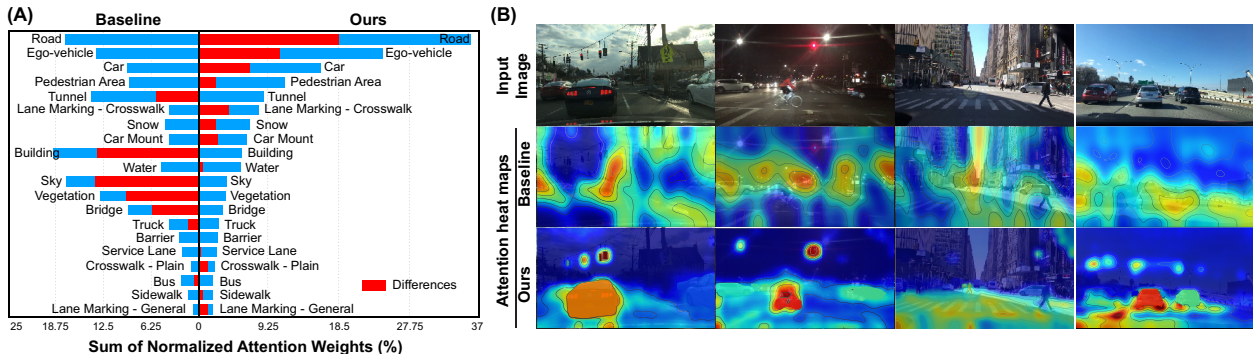


Figure 6.6: (A) The sum of normalized attention weights (blue) over the individual semantic regions for the baseline [37] and our model; differences shown in red. Our model attends more to road, car, pedestrian area, lane markings, and less to buildings, sky, vegetation. (We chose the top-20 most frequently attended regions of our model.) (B) We provide input images and compare attention maps from the baseline and our model. Attention maps are overlaid by their contour lines and shown over the input images. Higher value (red) of attention weight shows what the driving model attends to.

6.3.5 Towards Semantically Rich Driving Model

Analyzing the generated attention maps confirms that our model focuses more on important object-related visual cues (*e.g.* vehicles, lane markings, etc). In contrast, a baseline model [37] often attends to background (*e.g.* sky, trees, buildings, etc) but under-attends to important visual cues. In Figure 6.6 (A), we provide the top 20 semantic segmentation labels where our model attends to. Blue bars represent the sum of normalized attention weights for each label. The top 5 attended regions for our model are *road*, *ego-vehicle*, *pedestrian area*, *tunnel*, *lane marking*, while the baseline focuses on *building*, *road*, *sky*, *tunnel*, *ego-vehicle*. To see the difference between those models, we also visualize the differences as a red bar. Ours clearly focuses more on driving-related features, *e.g.* road, car, pedestrian area, lane markings, snow, and less on buildings, sky, vegetation, etc. In Figure 6.6 (B), we further compare the attention maps between ours and a baseline model [37]. We provide input video frames (1st row), attention maps generated by the baseline model (2nd row), and our attention maps (3rd row). Attention maps show that our model attends to important object-related visual cues (*e.g.* vehicles, lane markings, etc) with delineated object boundaries (more interpretable).

Note that, to see the effect of our front-end visual encoder, we use (i) DeepLab v3 [14] architecture and (ii) MaskRCNN [25] architecture, which are state-of-the-art approaches in the task of semantic segmentation and instance object detection, respectively. These networks are pretrained on the large-scale Mapillary dataset [57] and Microsoft COCO dataset [48].

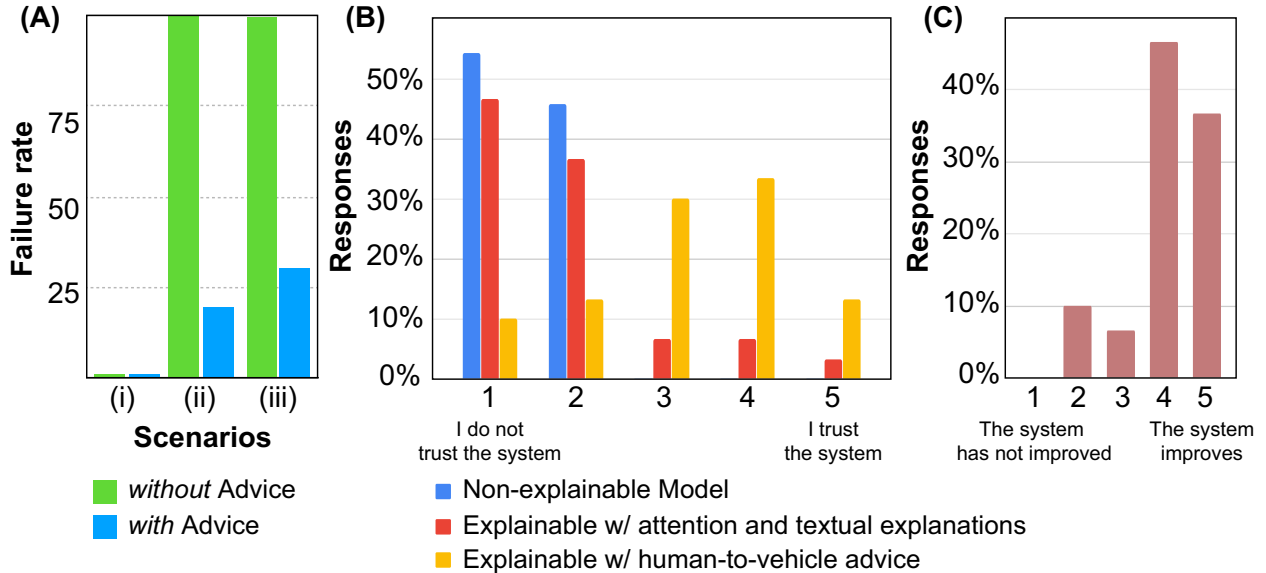


Figure 6.7: (A) We report the failure rate with and without advice inputs in the following three scenarios on a Carla simulator. (B-C) We also report the responses from our human study for the questions: (B) “How much do you trust this system?”, and (C) “To what level has the system improved with the human-to-vehicle advice?”. Answers were measured on a 1-5 Likert scale.

6.3.6 Generated Observation/Action Quality

Next we evaluate the quality of our generated observations and action commands, see Table 6.2 (higher is better). Our Textual Observation Generator predicts natural language observations based on the visual inputs. Some of our baselines are video captioning approaches, which do not take the vehicle control into account (S2VT [82]+SA (spatial attention)+TA (temporal attention) and Transformer-based approach [80]). At the same time, our full system is trained end-to-end, including the loss on the predicted controls, thus our textual observations are encouraged to be relevant to driving behavior. Therefore, we also compare to the best version of [39], the WAA model (weakly-aligned attention). This model generates action descriptions and explanations conditioned on predicted vehicle control, and we interpret the latter as observations. This is unlike our approach, where, conversely, vehicle control is predicted based on observations/action commands. Nevertheless, these are meaningful reference numbers for our approach. As we see, our model obtains the highest scores in all metrics both for generated observations and action commands.

6.3.7 Simulation and Human Evaluation

Explainable and advisable driving models can increase user trust by providing effective communication, which helps users convey their preferences/guidance to the vehicle and vice versa. To verify this, we run a human evaluation. We first migrate our driving model from the offline setting to a simulated environment, Carla [20], *i.e.* our model is trained on the BDD-X dataset and tested in the Carla simulator. We choose three different driving scenarios: (i) stopping at red lights, (ii) stopping at red lights in heavy rain, and (iii) stopping at a stop marking. In these experiments our driving model fails to stop for (ii) and (iii) scenarios. We then test the model with the following advice: “the light is red” and “there is a stop sign” for respective scenarios. We observe that the failure rate drops (see Figure 6.7 (A)). Further, we recruit 20 human judges and study the following three cases: (i) user only observes the car’s behavior, (ii) user observes the model’s behavior along with the attention and textual explanations, and (iii) user observes the model’s behavior, attention, and textual explanations, *before and after* providing advice. As shown in Figure 6.7 (B), our explainable and advisable system shows better responses for user-trust. Specifically, providing visual and textual explanations slightly improves the user trust (blue vs. red). Further, showing users an example where the driving model accepts human-to-vehicle advice significantly improves the user-trust (red vs. yellow). In addition, we obtain feedback from the users by asking “To what level has the system improved with the human-to-vehicle advice?”. Our evaluators acknowledge that advice improves the driving system, see Figure 6.7 (C). We provide the details of our evaluation in the Carla simulator in the supplemental material.

6.4 Related Work

6.4.1 End-to-End Learning for Self-driving Vehicles

Recent works [7, 26] suggest that a driving policy can be successfully learned by neural networks through supervised learning over observation (*e.g.* video) and action (*e.g.* steering) pairs, that are collected from human demonstration. Bojarski *et al.* [9] trained a 5-layer ConvNet to predict steering controls from a dashcam image, while Xu *et al.* [87] utilized a dilated ConvNet combined with an LSTM so as to predict vehicle’s discretized future motions. Recently, Hecker *et al.* [26] explored the extended model that takes a surround-view multi-camera system, a route planner, and a CAN bus reader. Codevilla *et al.* [18] explored a conditional end-to-end driving model that takes high-level command input (*i.e.* left-/right-turn, lane following, and intersection passing) at test time, see Figure 6.2 (A). To reduce the complexity, there is growing interest in end-to-mid [94] and mid-to-mid [7] driving models that produce a mid-level output representation in the form of a drivable trajectory by consuming either raw sensor or an intermediate scene representation as input. These models show good performance in simple driving scenarios (*e.g.* lane following). Their behavior, however, is opaque and learning to drive in urban areas remains challenging. These driving models are also known to be “black boxes” and thus lack of transparency may be a major

drawback in self-driving applications where a high level of user trust is required to accept such a radical technology.

6.4.2 Visual and Textual Explanations

Explainability of deep neural networks has become a growing field in computer vision and machine learning communities [24]. In landmark work, [93] utilized deconvolution layers to visualize the internal representation of a ConvNet. Other approaches [95, 70] have explored synthesizing an image that highly activates a neuron. However, they lack formal measures of how the function estimated by the network is affected by spatially-extended features.

Attention-based approaches may be exceptions to this rule. Kim *et al.*[37] utilized an attention model followed by additional salience filtering to show regions that causally affect the output. Wang *et al.* [83] and Wu *et al.* [86] introduced an instance-level attention model that finds objects (*e.g.* , cars, pedestrians) that the network needs to pay attention to. However, such attention may be less convenient (especially in the driving domain) for users to “replay”. It is also important to be able to justify the decisions that were made and explain why they are reasonable in a human understandable manner, *i.e.* in natural language. For an image classification problem, [27, 28] used an LSTM caption generation model that generates textual justifications of a CNN model. [61, 86] combine an attention-based model and a textual justification system to produce an interpretable model. Kim *et al.* [39] proposed a textual explanation model to explain the rationales behind the vehicle controller, see Figure 6.2 (B). Explainable models can help reveal what the model is doing and show the basis for its decisions, which makes it easier to expose weaknesses and further improve. We propose a model that is both explainable and advisable. Human-to-vehicle advice can take a variety of forms, while natural language is an intuitive form of communication for humans. Our approach is inspired by [39], but we incorporate advice through learning to generate observations and corresponding actions in natural language.

6.4.3 Advice-taking Models

Recognition of the value of advice-taking has a long history in AI community [54], but few attempts have been made to exploit textual advice. Several approaches have been proposed to translate natural language advice to formal semantic representations, which are then used to bias actions for simulated soccer [42], mobile manipulation [56, 55, 76], and navigation [4]. Recent work suggests that incorporating natural language human feedback can improve text-based QA agents [47, 85] and image captioning performance [49]. Despite its potential, there are various challenges with collecting human feedback on the actions taken by self-driving cars (*e.g.* safety and liability). Other notable approaches (in the reinforcement learning setting) include the work by Tung *et al.* [77] that learns a visual reward detector conditioned on natural language action descriptions, which is then used to train agents. Kim *et al.* [38] introduced an approach to ground instantaneous human-to-vehicle advice w.r.t. perception and action and showed that accepting such advice improves overall control prediction ac-

curacy, while Roh *et al.* [66] focused on conditioning natural language instructions to the driving model, see Figure 6.2 (A). Inspired by these work, we incorporate observation-action rules at training time, and learn to recognize when to follow advice at test time, rather than expecting such advice to be given by a “passenger” at test time.

6.4.4 Semantic Segmentation

Semantic segmentation is one of the fundamental building blocks of perception modules for autonomous driving. Most of the recent ConvNet-based models are inspired by the fully convolutional networks [53] and encoder-decoder architecture [5]. The recent models further explore the idea of encoder-decoder with fully convolutional skip connections to generate high-resolution outputs, such as U-Net [67], Feature Pyramid Network (FPN) [41], and Deep Layer Aggregation (DLA) [91]. Meanwhile, dilation [89, 90], also known as atrous convolution [14], is another widely used building block to generate high resolution intermediate feature maps, while enlarging the receptive field of a ConvNet. In order to obtain fine-grained visual attention and better ground natural language inputs, we utilize the state-of-the-art DeepLab v3 [15] network, which utilizes atrous convolutional filter with different dilation rates in parallel, to handle the objects at different scales.

Bibliography

- [1] Martin Abadi et al. “TensorFlow: Large-scale machine learning on heterogeneous systems”. In: (2015).
- [2] Zeynep Akata et al. “Generating Post-Hoc Rationales of Deep Visual Classification Decisions”. In: *Chapter in Explainable and Interpretable Models in Computer Vision and Machine Learning. The Springer Series on Challenges in Machine Learning* (2018).
- [3] Peter Anderson et al. “Spice: Semantic propositional image caption evaluation”. In: *ECCV*. Springer. 2016, pp. 382–398.
- [4] Yoav Artzi and Luke Zettlemoyer. “Weakly supervised learning of semantic parsers for mapping instructions to actions”. In: *TACL* (2013).
- [5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *TPAMI* (2017).
- [6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *ICLR* (2014).
- [7] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. “Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst”. In: *RSS* (2019).
- [8] Karsten Behrendt, Libor Novak, and Rami Botros. “A deep learning approach to traffic lights: Detection, tracking, and classification”. In: *ICRA*. IEEE. 2017, pp. 1370–1377.
- [9] Mariusz Bojarski et al. “End to end learning for self-driving cars”. In: *CoRR abs/1604.07316* (2016).
- [10] Mariusz Bojarski et al. “Visualbackprop: visualizing cnns for autonomous driving”. In: *arXiv preprint* (2016).
- [11] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*. Vol. 56. springer, 2009.
- [12] Peter Burt and Edward Adelson. “The Laplacian pyramid as a compact image code”. In: *IEEE Transactions on communications* 31.4 (1983), pp. 532–540.
- [13] Chenyi Chen et al. “Deepdriving: Learning affordance for direct perception in autonomous driving”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2722–2730.

- [14] Liang-Chieh Chen et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *TPAMI* (2018).
- [15] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).
- [16] Lu Chi and Yadong Mu. “Learning End-to-End Autonomous Steering Model from Spatial and Temporal Visual Cues”. In: *Proceedings of the Workshop on Visual Analysis in Smart and Connected Communities*. ACM. 2017, pp. 9–16.
- [17] Lu Chi and Yadong Mu. “Learning End-to-End Driving Model from Spatial and Temporal Visual Cues”. In: *Proceedings of the Workshop on Visual Analysis in Smart and Connected Communities* (2017).
- [18] Felipe Codevilla et al. “End-to-end driving via conditional imitation learning”. In: *ICRA*. IEEE. 2018, pp. 1–9.
- [19] Comma.ai. *Public driving dataset*. <https://github.com/commaai/research>. [Online; accessed 07-Mar-2017]. 2017.
- [20] Alexey Dosovitskiy et al. “CARLA: An open urban driving simulator”. In: *CoRL* (2017).
- [21] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.
- [22] Tharindu Fernando et al. “Going deeper: Autonomous steering with neural memory networks”. In: *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on*. IEEE. 2017, pp. 214–221.
- [23] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feed-forward neural networks.” In: *AISTATS*. 2010.
- [24] David Gunning. “Explainable Artificial Intelligence (XAI)”. In: *Defense Advanced Research Projects Agency (DARPA)* (2017).
- [25] Kaiming He et al. “Mask r-cnn”. In: *ICCV*. 2017, pp. 2961–2969.
- [26] Simon Hecker, Dengxin Dai, and Luc Van Gool. “End-to-end learning of driving models with surround-view cameras and route planners”. In: *ECCV*. 2018.
- [27] Lisa Anne Hendricks et al. “Generating visual explanations”. In: *ECCV*. 2016.
- [28] Lisa Anne Hendricks et al. “Grounding visual explanations”. In: *ECCV*. 2018.
- [29] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [30] Sepp Hochreiter and Jürgen Schmidhuber. “LSTM can solve hard long time lag problems”. In: *NeurIPS*. 1997, pp. 473–479.
- [31] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv preprint arXiv:1704.04861* (2017).

- [32] Jonathan Huang et al. “Speed/accuracy trade-offs for modern convolutional object detectors”. In: *CVPR*. 2017.
- [33] Rob Hyndman et al. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.
- [34] Morten Bornø Jensen et al. “Vision for looking at traffic lights: Issues, survey, and perspectives”. In: *ITSC 17.7* (2016), pp. 1800–1815.
- [35] Yangqing Jia et al. “Caffe: Convolutional Architecture for Fast Feature Embedding”. In: *arXiv preprint arXiv:1408.5093* (2014).
- [36] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. “Densecap: Fully convolutional localization networks for dense captioning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4565–4574.
- [37] Jinkyu Kim and John Canny. “Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention”. In: *ICCV* (2017).
- [38] Jinkyu Kim et al. “Grounding Human-to-Vehicle Advice for Self-driving Vehicles”. In: *CVPR* (2019).
- [39] Jinkyu Kim et al. “Textual Explanations for Self-Driving Vehicles”. In: *ECCV*. 2018.
- [40] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *ICLR* (2015).
- [41] Alexander Kirillov et al. “Panoptic Feature Pyramid Networks”. In: *arXiv preprint arXiv:1901.02446* (2019).
- [42] Gregory Kuhlmann et al. “Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer”. In: *AAAI Workshop*. 2004.
- [43] Alon Lavie and Abhaya Agarwal. “Meteor: An automatic metric for mt evaluation with improved correlation with human judgments”. In: *EMNLP*. 2005.
- [44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.
- [45] Honglak Lee et al. “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations”. In: *Proceedings of the 26th annual international conference on machine learning*. ACM. 2009, pp. 609–616.
- [46] Jesse Levinson et al. “Towards fully autonomous driving: Systems and algorithms”. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE. 2011, pp. 163–168.
- [47] Jiwei Li et al. “Dialogue learning with human-in-the-loop”. In: *arXiv preprint arXiv:1611.09823* (2016).
- [48] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *ECCV*. Springer. 2014, pp. 740–755.
- [49] Huan Ling and Sanja Fidler. “Teaching machines to describe images via natural language feedback”. In: *arXiv preprint arXiv:1706.00130* (2017).

- [50] Wei Liu et al. “SSD: Single shot multibox detector”. In: *ECCV*. Springer. 2016, pp. 21–37.
- [51] T. Lombrozo. *Explanation and abductive inference*. The Oxford handbook of thinking and reasoning, 2012.
- [52] Tania Lombrozo. “The structure and function of explanations”. In: *Trends in Cognitive Science* 10.10 (2006).
- [53] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *CVPR*. 2015.
- [54] John McCarthy. *Programs with common sense*. RLE and MIT computation center, 1960.
- [55] Dipendra Kumar Misra et al. “Environment-driven lexicon induction for high-level instructions”. In: *ACL*. 2015.
- [56] Dipendra K Misra et al. “Tell me dave: Context-sensitive grounding of natural language to manipulation instructions”. In: *IJRR* (2016).
- [57] Gerhard Neuhold et al. “The mapillary vistas dataset for semantic understanding of street scenes”. In: *ICCV*. 2017.
- [58] Brian Paden et al. “A survey of motion planning and control techniques for self-driving urban vehicles”. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 33–55.
- [59] Kishore Papineni et al. “BLEU: a method for automatic evaluation of machine translation”. In: *ACL*. 2002.
- [60] Dong Huk Park et al. “Multimodal Explanations: Justifying Decisions and Pointing to the Evidence”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.
- [61] Dong Huk Park et al. “Multimodal explanations: Justifying decisions and pointing to the evidence”. In: *CVPR*. 2018.
- [62] Mark Philip Philipson et al. “Learning based traffic light detection: Evaluation on challenging dataset”. In: *ITSC*. 2015.
- [63] Dean A Pomerleau. *ALVINN, an autonomous land vehicle in a neural network*. Tech. rep. Carnegie Mellon University, Computer Science Department, 1989.
- [64] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.
- [65] Vasili Ramanishka et al. “Toward Driving Scene Understanding: A Dataset for Learning Driver Behavior and Causal Reasoning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7699–7707.
- [66] Junha Roh et al. “Conditional Driving from Natural Language Instructions”. In: *CoRL* (2019).

- [67] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *MICCAI*. 2015.
- [68] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kotschieder. “In-Place Activated Batch-Norm for Memory-Optimized Training of DNNs”. In: *CVPR*. 2018.
- [69] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *IJCV* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.
- [70] Ramprasaath R Selvaraju et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *ICCV*. 2017, pp. 618–626.
- [71] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. “Action recognition using visual attention”. In: *arXiv preprint arXiv:1511.04119* (2015).
- [72] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *ICLR* (2015).
- [73] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting.” In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.
- [74] Christian Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning.” In: *AAAI*. Vol. 4. 2017, p. 12.
- [75] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826.
- [76] Stefanie Tellex et al. “Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation”. In: *AAAI*. 2011.
- [77] Hsiao-Yu Fish Tung et al. “Reward learning from narrated demonstrations”. In: *CVPR* (2018).
- [78] Udacity. *Public driving dataset*. <https://www.udacity.com/self-driving-car>. [Online; accessed 07-Mar-2017]. 2017.
- [79] Chris Urmson et al. “Autonomous driving in urban environments: Boss and the urban challenge”. In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466.
- [80] Ashish Vaswani et al. “Attention is all you need”. In: *NeurIPS*. 2017.
- [81] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. “Cider: Consensus-based image description evaluation”. In: *ICCV*. 2015.
- [82] Subhashini Venugopalan et al. “Sequence to sequence-video to text”. In: *ICCV*. 2015, pp. 4534–4542.
- [83] Dequan Wang et al. “Deep Object Centric Policies for Autonomous Driving”. In: *ICRA* (2019).
- [84] Michael Weber, Peter Wolf, and J Marius Zöllner. “Deeptlr: A single deep convolutional network for detection and classification of traffic lights”. In: *IV*. IEEE. 2016, pp. 342–348.

- [85] Jason E Weston. “Dialog-based language learning”. In: *NeurIPS*. 2016.
- [86] Jialin Wu and Raymond J Mooney. “Faithful Multimodal Explanation for Visual Question Answering”. In: *arXiv preprint arXiv:1809.02805* (2018).
- [87] Huazhe Xu et al. “End-to-end Learning of Driving Models from Large-scale Video Datasets”. In: *arXiv preprint arXiv:1612.01079* (2016).
- [88] Kelvin Xu et al. “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.” In: *ICML*. Vol. 14. 2015, pp. 77–81.
- [89] Fisher Yu and Vladlen Koltun. “Multi-scale context aggregation by dilated convolutions”. In: *ICLR*. 2016.
- [90] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. “Dilated residual networks”. In: *CVPR*. 2017.
- [91] Fisher Yu et al. “Deep Layer Aggregation”. In: *CVPR*. 2018.
- [92] Tom Zahavy, Nir Ben Zrihem, and Shie Mannor. “Graying the black box: Understanding DQNs”. In: *arXiv preprint arXiv:1602.02658* (2016).
- [93] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 818–833.
- [94] Wenyuan Zeng et al. “End-to-end Interpretable Neural Motion Planner”. In: *CVPR*. 2019, pp. 8660–8669.
- [95] Bolei Zhou et al. “Learning deep features for discriminative localization”. In: *CVPR*. 2016, pp. 2921–2929.