Brain4Cars: Car That Knows Before You Do via Sensory-Fusion Deep Learning Architecture

Ashesh Jain[1,2], Hema S Koppula[1,2], Shane Soh[2], Bharad Raghavan[2], Avi Singh[1], and Ashutosh Saxena[3]

Cornell University[1], Stanford University[2], Brain Of Things Inc.[3]
{ashesh,hema}@cs.cornell.edu, avisingh@iitk.ac.in, {shanesoh,bharadr,asaxena}@cs.stanford.edu

***Abstract*—Advanced Driver Assistance Systems (ADAS) have made driving safer over the last decade. They prepare vehicles for unsafe road conditions and alert drivers if they perform a dangerous maneuver. However, many accidents are unavoidable because by the time drivers are alerted, it is already too late. Anticipating maneuvers beforehand can alert drivers before they perform the maneuver and also give ADAS more time to avoid or prepare for the danger.**

In this work we propose a vehicular sensor-rich platform and learning algorithms for maneuver anticipation. For this purpose we equip a car with cameras, Global Positioning System (GPS), and a computing device to capture the driving context from both inside and outside of the car. In order to anticipate maneuvers, we propose a sensory-fusion deep learning architecture which jointly learns to anticipate and fuse multiple sensory streams. Our architecture consists of Recurrent Neural Networks (RNNs) that use Long Short-Term Memory (LSTM) units to capture long temporal dependencies. We propose a novel training procedure which allows the network to predict the future given only a partial temporal context. We introduce a diverse data set with 1180 miles of natural freeway and city driving, and show that we can anticipate maneuvers 3.5 seconds before they occur in real-time with a precision and recall of 90.5% and 87.4% respectively.

I. Introduction

Over the last decade cars have been equipped with various assistive technologies in order to provide a safe driving experience. Technologies such as lane keeping, blind spot check, pre-crash systems etc., are successful in alerting drivers whenever they commit a dangerous maneuver [43]. Still in the US alone more than 33,000 people die in road accidents every year, the majority of which are due to inappropriate maneuvers [2]. We therefore need mechanisms that can alert drivers *before* they perform a dangerous maneuver in order to avert many such accidents [56].

In this work we address the problem of anticipating maneuvers that a driver is likely to perform in the next few seconds. Figure 1 shows our system anticipating a left turn maneuver a few seconds before the car reaches the intersection. Our system also outputs probabilities over the maneuvers the driver can perform. With this prior knowledge of maneuvers, the driver assistance systems can alert drivers about possible dangers before they perform the maneuver, thereby giving them more time to react. Some previous works [22, 41, 50] also predict a driver's future maneuver. However, as we show in the following sections, these methods use limited context and/or do not accurately model the anticipation problem.

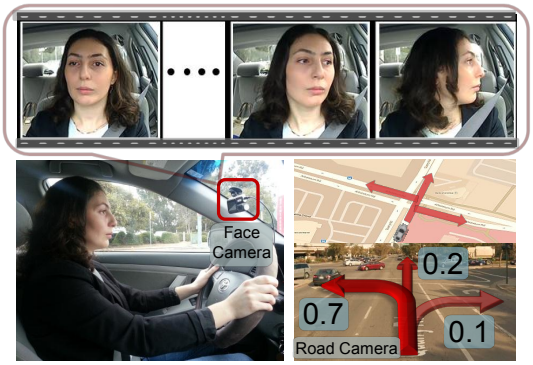


Fig. 1: **Anticipating maneuvers.** Our algorithm anticipates driving maneuvers performed a few seconds in the future. It uses information from multiple sources including videos, vehicle dynamics, GPS, and street maps to anticipate the probability of different future maneuvers.

In order to anticipate maneuvers, we reason with the contextual information from the surrounding events, which we refer to as the *driving context*. We obtain this driving context from multiple sources. We use videos of the driver inside the car and the road in front, the vehicle's dynamics, global position coordinates (GPS), and street maps; from this we extract a time series of multi-modal data from both inside and outside the vehicle. The challenge lies in modeling the temporal aspects of driving and fusing the multiple sensory streams. In this work we propose a specially tailored approach for anticipation in such sensory-rich settings.

Anticipation of the future actions of a human is an important perception task with applications in robotics and computer vision [39, 77, 33, 34, 73]. It requires the prediction of future events from a limited temporal context. This differentiates anticipation from *activity recognition* [73], where the complete temporal context is available for prediction. Furthermore, in sensory-rich robotics settings like ours, the context for anticipation comes from multiple sensors. In such scenarios the end performance of the application largely depends on how the information from different sensors are fused. Previous works on anticipation [33, 34, 39] usually deal with single-data modality and do not address anticipation for sensory-rich robotics applications. Additionally, they learn representations using shallow architectures [30, 33, 34, 39] that cannot handle long temporal dependencies [6].

In order to address the anticipation problem more generally,

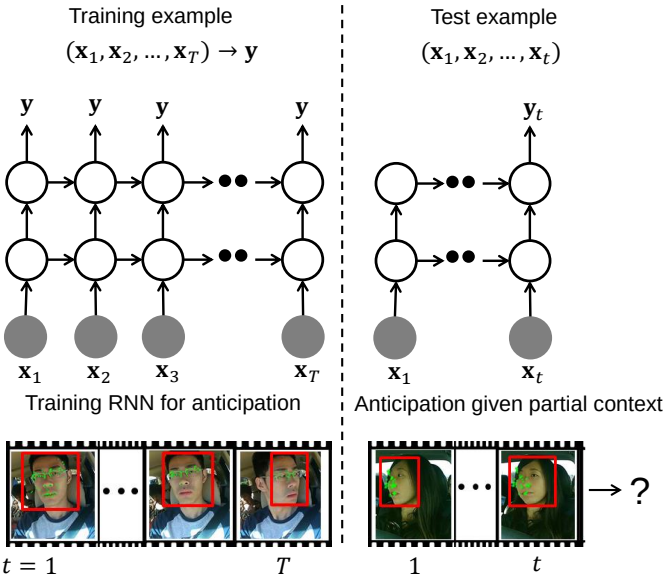


Fig. 2: (**Left**) Shows training RNN for anticipation in a sequence-to-sequence prediction manner. The network explicitly learns to map the partial context $(\mathbf{x}_1, .., \mathbf{x}_t)$ $\forall t$ to the future event $\mathbf{y}$. (**Right**) At test time the network's goal is to anticipate the future event as soon as possible, i.e. by observing only a partial temporal context.

we propose a Recurrent Neural Network (RNN) based architecture which learns rich representations for anticipation. We focus on sensory-rich robotics applications, and our architecture learns how to optimally fuse information from different sensors. Our approach captures temporal dependencies by using Long Short-Term Memory (LSTM) units. We train our architecture in a sequence-to-sequence prediction manner (Figure 2) such that it explicitly learns to anticipate given a partial context, and we introduce a novel loss layer which helps anticipation by preventing over-fitting.

We evaluate our approach on a driving data set with 1180 miles of natural freeway and city driving collected across two states – from 10 drivers and with different kinds of driving maneuvers. The data set is challenging because of the variations in routes and traffic conditions, and the driving styles of the drivers (Figure 3). We demonstrate that our deep learning sensory-fusion approach anticipates maneuvers 3.5 seconds before they occur with 84.5% precision and 77.1% recall while using out-of-the-box face tracker. With more sophesticated 3D pose estimation of the face, our precision and recall increases to **90.5%** and **87.4%** respectively. We believe that our work creates scope for new ADAS features to make roads safer. In summary our key contributions are as follows:

- We propose an approach for anticipating driving maneuvers several seconds in advance.
- We propose a generic sensory-fusion RNN-LSTM architecture for anticipation in robotics applications.
- We release the first data set of natural driving with videos from both inside and outside the car, GPS, and speed information.
- We release an open-source deep learning package `NeuralModels` which is especially designed for robotics applications with multiple sensory streams.

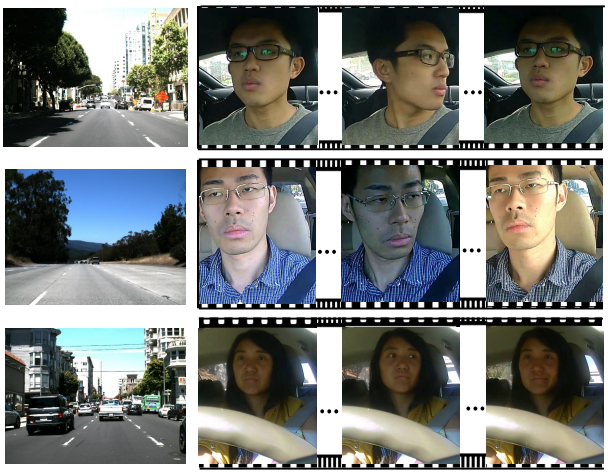


Fig. 3: **Variations in the data set.** Images from the data set [30] for a left lane change. (**Left**) Views from the road facing camera. (**Right**) Driving style of the drivers vary for the same maneuver.

Our data set and deep learning code are publicly available at: http://www.brain4cars.com

II. Related Work

Our work builds upon the previous works on assisitive vehicular technologies, anticipating human activities, learning temporal models, and computer vision methods for analyzing human face.

Assistive features for vehicles. Latest cars available in market comes equipped with cameras and sensors to monitor the surrounding environment. Through multi-sensory fusion they provide assisitive features like lane keeping, forward collision avoidance, adaptive cruise control etc. These systems warn drivers when they perform a potentially dangerous maneuver [59, 68]. Driver monitoring for distraction and drowsiness has also been extensively researched [21, 55]. Techniques like eye-gaze tracking are now commercially available (Seeing Machines Ltd.) and has been effective in detecting distraction. Our work complements existing ADAS and driver monitoring techniques by anticipating maneuvers several seconds before they occur.

Closely related to us are previous works on predicting the driver's intent. Vehicle trajectory has been used to predict the intent for lane change or turn maneuver [9, 22, 41, 44]. Most of these works ignore the rich context available from cameras, GPS, and street maps. Previous works have addressed maneuver anticipation [1, 50, 15, 67] through sensory-fusion from multiple cameras, GPS, and vehicle dynamics. In particular, Morris et al. [50] and Trivedi et al. [67] used Relevance Vector Machine (RVM) for intent prediction and performed sensory fusion by concatenating feature vectors. We will show that such hand designed concatenation of features does not work well. Furthermore, these works do not model the temporal aspect of the problem properly. They assume

that informative contextual cues always appear at a fixed time before the maneuver. We show that this assumption is not true, and in fact the temporal aspect of the problem should be carefully modeled. In contrast to these works, our RNN-LSTM based sensory-fusion architecture captures long temporal dependencies through its memory cell and learns rich representations for anticipation through a hierarchy of non-linear transformations of input data. Our work is also related to works on driver behavior prediction with different sensors [26, 21, 20], and vehicular controllers which act on these predictions [59, 68, 18].

Anticipation and Modeling Humans. Modeling of human motion has given rise to many applications, anticipation being one of them. Anticipating human activities has shown to improve human-robot collaboration [73, 36, 46, 38, 16]. Similarly, forecasting human navigation trajectories has enabled robots to plan sociable trajectories around humans [33, 8, 39, 29]. Feature matching techniques have been proposed for anticipating human activities from videos [57]. Modeling human preferences has enabled robots to plan good trajectories [17, 60, 28, 31]. Similar to these works, we anticipate human actions, which are driving maneuvers in our case. However, the algorithms proposed in the previous works do not apply in our setting. In our case, anticipating maneuvers requires modeling the interaction between the driving context and the driver's intention. Such interactions are absent in the previous works, and they use shallow architectures [6] that do not properly model temporal aspects of human activities. They further deal with a single data modality and do not tackle the challenges of sensory-fusion. Our problem setup involves all these challenges, for which we propose a deep learning approach which efficiently handles temporal dependencies and learns to fuse multiple sensory streams.

Analyzing the human face. The vision approaches related to our work are face detection and tracking [69, 76], statistical models of face [10] and pose estimation methods for face [75]. Active Appearance Model (AAM) [10] and its variants [47, 74] statistically model the shape and texture of the face. AAMs have also been used to estimate the 3D-pose of a face from a single image [75] and in design of assistive features for driver monitoring [55, 63]. In our approach we adapt off-the-shelf available face detection [69] and tracking algorithms [58] (see Section VI). Our approach allows us to easily experiment with more advanced face detection and tracking algorithms. We demonstrate this by using the Constrained Local Neural Field (CLNF) model [4] and tracking 68 fixed landmark points on the driver's face and estimating the 3D head-pose.

Learning temporal models. Temporal models are commonly used to model human activities [35, 49, 71, 72]. These models have been used in both discriminative and generative fashions. The discriminative temporal models are mostly inspired by the Conditional Random Field (CRF) [42] which captures the temporal structure of the problem. Wang et al. [72] and Morency et al. [49] propose dynamic extensions of the CRF for image segmentation and gesture recognition respectively. On the other hand, generative approaches for temporal modeling include various filtering methods, such as Kalman and particle filters [64], Hidden Markov Models, and many types of Dynamic Bayesian Networks [51]. Some previous works [9, 40, 53] used HMMs to model different aspects of the driver's behaviour. Most of these generative approaches model how latent (hidden) states influence the observations. However, in our problem both the latent states and the observations influence each other. In the following sections, we will describe the Autoregressive Input-Output HMM (AIO-HMM) for maneuver anticipation [30] and will use it as a baseline to compare our deep learning approach. Unlike AIO-HMM our deep architecture have internal memory which allows it to handle long temporal dependencies [24]. Furthermore, the input features undergo a hierarchy of non-linear transformation through the deep architecture which allows learning rich representations.

Two building blocks of our architecture are Recurrent Neural Networks (RNNs) [54] and Long Short-Term Memory (LSTM) units [25]. Our work draws upon ideas from previous works on RNNs and LSTM from the language [62], speech [23], and vision [14] communities. Our approach to the joint training of multiple RNNs is related to the recent work on hierarchical RNNs [19]. We consider RNNs in multi-modal setting, which is related to the recent use of RNNs in image-captioning [14]. Our contribution lies in formulating activity anticipation in a deep learning framework using RNNs with LSTM units. We focus on sensory-rich robotics applications, and our architecture extends previous works doing sensory-fusion with feed-forward networks [52, 61] to the fusion of temporal streams. Using our architecture we demonstrate state-of-the-art on maneuver anticipation.

III. Overview

We first give an overview of the maneuver anticipation problem and then describe our system.

A. Problem Overview

Our goal is to anticipate driving maneuvers a few seconds before they occur. This includes anticipating a lane change before the wheels touch the lane markings or anticipating if the driver keeps straight or makes a turn when approaching an intersection. This is a challenging problem for multiple reasons. First, it requires the modeling of context from different sources. Information from a single source, such as a camera capturing events outside the car, is not sufficiently rich. Additional visual information from within the car can also be used. For example, the driver's head movements are useful for anticipation – drivers typically check for the side traffic while changing lanes and scan the cross traffic at intersections.

Second, reasoning about maneuvers should take into account the driving context at both local and global levels. Local context requires modeling events in vehicle's vicinity such as the surrounding vision, GPS, and speed information. On the other hand, factors that influence the overall route contributes to the global context, such as the driver's final destination. Third, the informative cues necessary for anticipation appear at

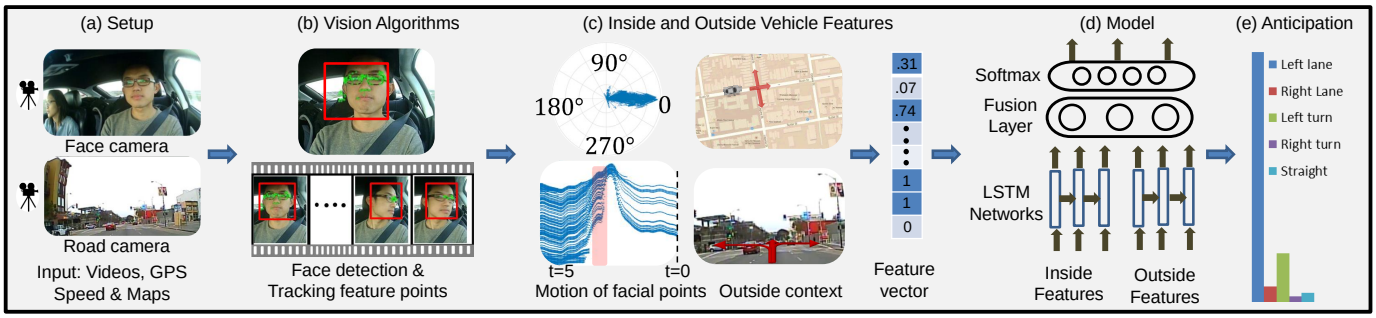


Fig. 4: **System Overview.** Our system anticipating a left lane change maneuver. (a) We process multi-modal data including GPS, speed, street maps, and events inside and outside of the vehicle using video cameras. (b) Vision pipeline extracts visual cues such as driver's head movements. (c) The inside and outside driving context is processed to extract expressive features. (d,e) Using our deep learning architecture we fuse the information from outside and inside the vehicle and anticipate the probability of each maneuver.

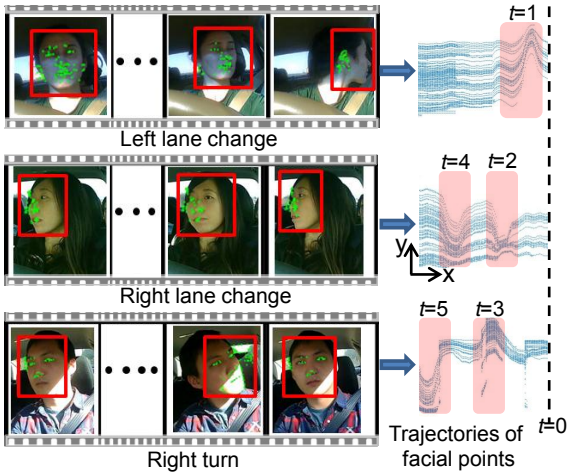


Fig. 5: **Variable time occurrence of events.** *Left*: The events inside the vehicle before the maneuvers. We track the driver's face along with many facial points. *Right*: The trajectories generated by the horizontal motion of facial points (pixels) 't' seconds before the maneuver. X-axis is the time and Y-axis is the pixels' horizontal coordinates. Informative cues appear during the shaded time interval. Such cues occur at variable times before the maneuver, and the order in which the cues appear is also important.

variable times before the maneuver, as illustrated in Figure 5. In particular, the time interval between the driver's head movement and the occurrence of the maneuver depends on many factors such as the speed, traffic conditions, etc.

In addition, appropriately fusing the information from multiple sensors is crucial for anticipation. Simple sensory fusion approaches like concatenation of feature vectors performs poorly, as we demonstrate through experiments. In our proposed approach we learn a neural network layer for fusing the temporal streams of data coming from different sensors. Our resulting architecture is end-to-end trainable via back propagation, and we jointly train it to: (i) model the temporal aspects of the problem; (ii) fuse multiple sensory streams; and (iii) anticipate maneuvers.

B. System Overview

For maneuver anticipation our vehicular sensory platform includes the following (as shown in Figure 4):

1) A driver-facing camera inside the vehicle. We mount this camera on the dashboard and use it to track the driver's head movements. This camera operates at 25 fps.
2) A camera facing the road is mounted on the dashboard to capture the (outside) view in front of the car. This camera operates at 30 fps. The video from this camera enables additional reasoning on maneuvers. For example, when the vehicle is in the left-most lane, the only safe maneuvers are a right-lane change or keeping straight, unless the vehicle is approaching an intersection.
3) A speed logger for vehicle dynamics because maneuvers correlate with the vehicle's speed, e.g., turns usually happen at lower speeds than lane changes.
4) A Global Positioning System (GPS) for localizing the vehicle on the map. This enables us to detect upcoming road artifacts such as intersections, highway exits, etc.

Using this system we collect 1180 miles of natural city and freeway driving data from 10 drivers. We denote the information from sensors with feature vector $\mathbf{x}$. Our vehicular systems gives a temporal sequence of feature vectors $\{(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t, ...)\}$. For now we do not distinguish between the information from different sensors, later in Section V-B we introduce sensory fusion. In Section VI we formally define our feature representations and describe our data set in Section VIII-A. We now formally define anticipation and present our deep learning architecture.

IV. Preliminaries

We now formally define anticipation and then present our Recurrent Neural Network architecture. The goal of anticipation is to predict an event several seconds before it happens given the contextual information up to the present time. The future event can be one of multiple possibilities. At training time a set of temporal sequences of observations and events $\{(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)_j, \mathbf{y}_j\}_{j=1}^{N}$ is provided where $\mathbf{x}_t$ is the observation at time t, $\mathbf{y}$ is the representation of the event

(described below) that happens at the end of the sequence at $t = T$, and j is the sequence index. At test time, however, the algorithm receives an observation $\mathbf{x}_t$ at each time step, and its goal is to predict the future event as early as possible, i.e. by observing only a partial sequence of observations $\{(\mathbf{x}_1, ..., \mathbf{x}_t)|t < T\}$. This differentiates anticipation from *activity recognition* [70, 37] where in the latter the complete observation sequence is available at test time. In this paper, $\mathbf{x}_t$ is a real-valued feature vector and $\mathbf{y} = [y^1, ..., y^K]$ is a vector of size K (the number of events), where y^k denotes the probability of the temporal sequence belonging to event the k such that $\sum_{k=1}^{K} y^k = 1$. At the time of training, $\mathbf{y}$ takes the form of a one-hot vector with the entry in $\mathbf{y}$ corresponding to the ground truth event as 1 and the rest 0.

In this work we propose a deep RNN architecture with Long Short-Term Memory (LSTM) units [25] for anticipation. Below we give an overview of the standard RNN and LSTM which form the building blocks of our architecture.

A. Recurrent Neural Networks

A standard RNN [54] takes in a temporal sequence of vectors $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$ as input, and outputs a sequence of vectors $(\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T)$ also known as high-level representations. The representations are generated by non-linear transformation of the input sequence from $t = 1$ to T, as described in the equations below.

$$\mathbf{h}_t = f(\mathbf{W}\mathbf{x}_t + \mathbf{H}\mathbf{h}_{t-1} + \mathbf{b}) \tag{1}$$

$$\mathbf{y}_t = \texttt{softmax}(\mathbf{W}_y\mathbf{h}_t + \mathbf{b}_y) \tag{2}$$

where f is a non-linear function applied element-wise, and $\mathbf{y}_t$ is the `softmax` probabilities of the events having seen the observations up to $\mathbf{x}_t$. $\mathbf{W}$, $\mathbf{H}$, $\mathbf{b}$, $\mathbf{W}_y$, $\mathbf{b}_y$ are the parameters that are learned. Matrices are denoted with bold, capital letters, and vectors are denoted with bold, lower-case letters. In a standard RNN a common choice for f is `tanh` or `sigmoid`. RNNs with this choice of f suffer from a well-studied problem of *vanishing gradients* [54], and hence are poor at capturing long temporal dependencies which are essential for anticipation. A common remedy to vanishing gradients is to replace `tanh` non-linearities by Long Short-Term Memory cells [25]. We now give an overview of LSTM and then describe our model for anticipation.

B. Long-Short Term Memory Cells

LSTM is a network of neurons that implements a memory cell [25]. The central idea behind LSTM is that the memory cell can maintain its state over time. When combined with RNN, LSTM units allow the recurrent network to remember long term context dependencies.

LSTM consists of three gates – input gate $\mathbf{i}$, output gate $\mathbf{o}$, and forget gate $\mathbf{f}$ – and a memory cell $\mathbf{c}$. See Figure 6 for an illustration. At each time step t, LSTM first computes its gates' activations $\{\mathbf{i}_t,\mathbf{f}_t\}$ (3)(4) and updates its memory cell from $\mathbf{c}_{t-1}$ to $\mathbf{c}_t$ (5), it then computes the output gate activation $\mathbf{o}_t$ (6), and finally outputs a hidden representation $\mathbf{h}_t$ (7). The inputs into LSTM are the observations $\mathbf{x}_t$ and the hidden representation from the previous time step $\mathbf{h}_{t-1}$. LSTM applies the following set of update operations:

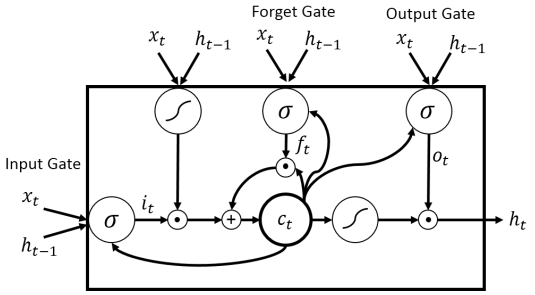


Fig. 6: **Internal working of an LSTM unit.**

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{V}_i\mathbf{c}_{t-1} + \mathbf{b}_i) \tag{3}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{V}_f\mathbf{c}_{t-1} + \mathbf{b}_f) \tag{4}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \texttt{tanh}(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{5}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{V}_o\mathbf{c}_t + \mathbf{b}_o) \tag{6}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \texttt{tanh}(\mathbf{c}_t) \tag{7}$$

where $\odot$ is an element-wise product and σ is the logistic function. σ and `tanh` are applied element-wise. $\mathbf{W}_*$, $\mathbf{V}_*$, $\mathbf{U}_*$, and $\mathbf{b}_*$ are the parameters, further the weight matrices $\mathbf{V}_*$ are diagonal. The input and forget gates of LSTM participate in updating the memory cell (5). More specifically, forget gate controls the part of memory to forget, and the input gate computes new values based on the current observation that are written to the memory cell. The output gate together with the memory cell computes the hidden representation (7). Since LSTM cell activation involves *summation* over time (5) and derivatives distribute over sums, the gradient in LSTM gets propagated over a longer time before vanishing. In the standard RNN, we replace the non-linear f in equation (1) by the LSTM equations given above in order to capture long temporal dependencies. We use the following shorthand notation to denote the recurrent LSTM operation.

$$(\mathbf{h}_t, \mathbf{c}_t) = \text{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}) \tag{8}$$

We now describe our RNN architecture with LSTM units for anticipation. Following which we will describe a particular instantiation of our architecture for maneuver anticipation where the observations $\mathbf{x}$ come from multiple sources.

V. Network Architecture for Anticipation

In order to anticipate, an algorithm must learn to predict the future given only a partial temporal context. This makes anticipation challenging and also differentiates it from activity recognition. Previous works treat anticipation as a recognition problem [34, 50, 57] and train discriminative classifiers (such as SVM or CRF) on the complete temporal context. However, at test time these classifiers only observe a partial temporal context and make predictions within a filtering framework. We model anticipation with a recurrent architecture which unfolds through time. This lets us train a single classifier that learns to handle partial temporal context of varying lengths.

Furthermore, anticipation in robotics applications is challenging because the contextual information can come from multiple sensors with different data modalities. Examples include autonomous vehicles that reason from multiple sensors [3] or robots that jointly reason over perception and language instructions [48]. In such applications the way information from different sensors is fused is critical to the application's final performance. We therefore build an end-to-end deep learning architecture which jointly learns to anticipate and fuse information from different sensors.

A. RNN with LSTM units for anticipation

At the time of training, we observe the complete temporal observation sequence and the event $\{(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T), \mathbf{y}\}$. Our goal is to train a network which predicts the future event given a partial temporal observation sequence $\{(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t) | t < T\}$. We do so by training an RNN in a sequence-to-sequence prediction manner. Given training examples $\{(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)_j, \mathbf{y}_j\}_{j=1}^N$ we train an RNN with LSTM units to map the sequence of observations $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$ to the sequence of events $(\mathbf{y}_1, ..., \mathbf{y}_T)$ such that $\mathbf{y}_t = \mathbf{y}, \forall t$, as shown in Fig. 2. Trained in this manner, our RNN will attempt to map all sequences of partial observations $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t)$ $\forall t \leq T$ to the future event $\mathbf{y}$. This way our model explicitly learns to anticipate. We additionally use LSTM units which prevents the gradients from vanishing and allows our model to capture long temporal dependencies in human activities.[1]

B. Fusion-RNN: Sensory fusion RNN for anticipation

We now present an instantiation of our RNN architecture for fusing two sensory streams: $\{(\mathbf{x}_1, ..., \mathbf{x}_T), (\mathbf{z}_1, ..., \mathbf{z}_T)\}$. In the next section we will describe these streams for maneuver anticipation.

An obvious way to allow sensory fusion in the RNN is by concatenating the streams, i.e. using $([\mathbf{x}_1; \mathbf{z}_1], ..., [\mathbf{x}_T; \mathbf{z}_T])$ as input to the RNN. However, we found that this sort of simple concatenation performs poorly. We instead learn a sensory fusion layer which combines the high-level representations of sensor data. Our proposed architecture first passes the two sensory streams $\{(\mathbf{x}_1, ..., \mathbf{x}_T), (\mathbf{z}_1, ..., \mathbf{z}_T)\}$ independently through separate RNNs (9) and (10). The high level representations from both RNNs $\{(\mathbf{h}_1^x, ..., \mathbf{h}_T^x), (\mathbf{h}_1^z, ..., \mathbf{h}_T^z)$ are then concatenated at each time step t and passed through a fully connected (fusion) layer which fuses the two representations (11), as shown in Figure 7. The output representation from the fusion layer is then passed to the softmax layer for anticipation (12). The following operations are performed from $t = 1$ to T.

$$(\mathbf{h}_t^x, \mathbf{c}_t^x) = \text{LSTM}_x(\mathbf{x}_t, \mathbf{h}_{t-1}^x, \mathbf{c}_{t-1}^x) \quad (9)$$

$$(\mathbf{h}_t^z, \mathbf{c}_t^z) = \text{LSTM}_z(\mathbf{z}_t, \mathbf{h}_{t-1}^z, \mathbf{c}_{t-1}^z) \quad (10)$$

$$\text{Sensory fusion: } \mathbf{e}_t = \texttt{tanh}(\mathbf{W}_f[\mathbf{h}_t^x; \mathbf{h}_t^z] + \mathbf{b}_f) \quad (11)$$

$$\mathbf{y}_t = \texttt{softmax}(\mathbf{W}_y \mathbf{e}_t + \mathbf{b}_y) \quad (12)$$

[1]Driving maneuvers can take up to 6 seconds and the value of T can go up to 150 with a camera frame rate of 25 fps.

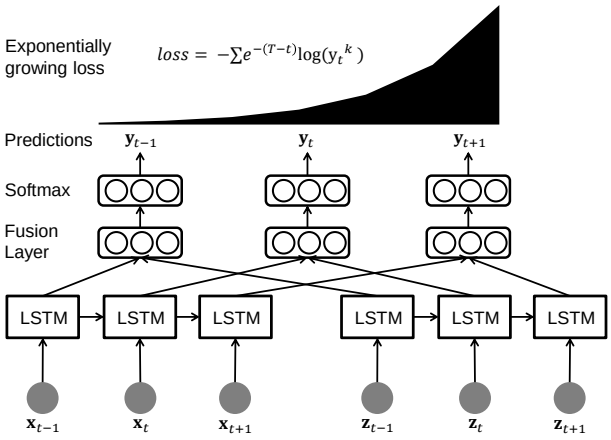


Fig. 7: **Sensory fusion RNN for anticipation.** (**Bottom**) In the Fusion-RNN each sensory stream is passed through their independent RNN. (**Middle**) High-level representations from RNNs are then combined through a fusion layer. (**Top**) In order to prevent over-fitting early in time the loss exponentially increases with time.

where $\mathbf{W}_*$ and $\mathbf{b}_*$ are model parameters, and LSTM_x and LSTM_z process the sensory streams $(\mathbf{x}_1, ..., \mathbf{x}_T)$ and $(\mathbf{z}_1, ..., \mathbf{z}_T)$ respectively. The same framework can be extended to handle more sensory streams.

C. Exponential loss-layer for anticipation.

We propose a new loss layer which encourages the architecture to anticipate early while also ensuring that the architecture does not over-fit the training data early enough in time when there is not enough context for anticipation. When using the standard softmax loss, the architecture suffers a loss of $-\log(y_t^k)$ for the mistakes it makes at each time step, where y_t^k is the probability of the ground truth event k computed by the architecture using Eq. (12). We propose to modify this loss by multiplying it with an exponential term as illustrated in Figure 7. Under this new scheme, the loss exponentially grows with time as shown below.

$$loss = \sum_{j=1}^{N} \sum_{t=1}^{T} -e^{-(T-t)} \log(y_t^k) \quad (13)$$

This loss penalizes the RNN exponentially more for the mistakes it makes as it sees more observations. This encourages the model to fix mistakes as early as it can in time. The loss in equation 13 also penalizes the network less on mistakes made early in time when there is not enough context available. This way it acts like a regularizer and reduces the risk to over-fit very early in time.

VI. Features

We extract features by processing the inside and outside driving contexts. We do this by grouping the overall contextual information from the sensors into: (i) the context from inside the vehicle, which comes from the driver facing camera and is represented as temporal sequence of features $(\mathbf{z}_1, ..., \mathbf{z}_T)$; and (ii) the context from outside the vehicle, which comes from the remaining sensors: GPS, road facing camera, and street

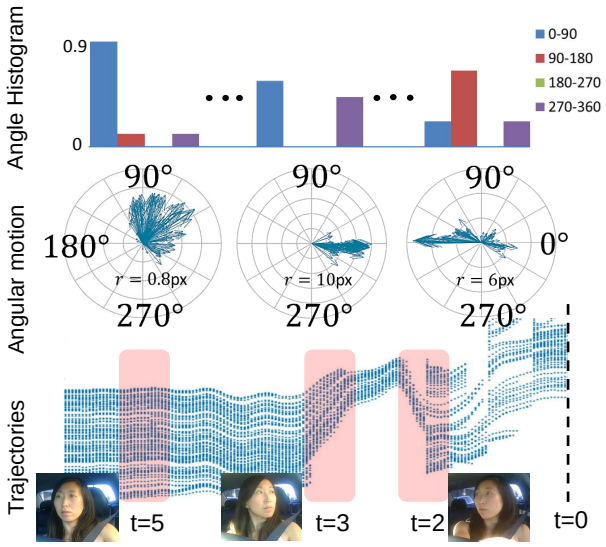


Fig. 8: **Inside vehicle feature extraction.** The angular histogram features extracted at three different time steps for a left turn maneuver. *Bottom*: Trajectories for the horizontal motion of tracked facial pixels 't' seconds before the maneuver. At t=5 seconds before the maneuver the driver is looking straight, at t=3 looks (left) in the direction of maneuver, and at t=2 looks (right) in opposite direction for the crossing traffic. *Middle*: Average motion vector of tracked facial pixels in polar coordinates. r is the average movement of pixels and arrow indicates the direction in which the face moves when looking from the camera. *Top*: Normalized angular histogram features.

maps. We represent the outside context with $(\mathbf{x}_1, ..., \mathbf{x}_T)$. In order to anticipate maneuvers, our RNN architecture (Figure 7) processes the temporal context $\{(\mathbf{x}_1, ..., \mathbf{x}_t), (\mathbf{z}_1, ..., \mathbf{z}_t)\}$ at every time step t, and outputs softmax probabilities $\mathbf{y}_t$ for the following five maneuvers: $\mathcal{M} = \{$*left turn*, *right turn*, *left lane change*, *right lane change*, *straight driving*$\}$.

A. *Inside-vehicle features.*

The inside features $\mathbf{z}_t$ capture the driver's head movements at each time instant t. Our vision pipeline consists of face detection, tracking, and feature extraction modules. We extract head motion features per-frame, denoted by $\phi(\text{face})$. We compute $\mathbf{z}_t$ by aggregating $\phi(\text{face})$ for every 20 frames, i.e., $\mathbf{z}_t = \sum_{i=1}^{20} \phi(\text{face}_i) / \| \sum_{i=1}^{20} \phi(\text{face}_i) \|$.

Face detection and tracking. We detect the driver's face using a trained Viola-Jones face detector [69]. From the detected face, we first extract visually discriminative (facial) points using the Shi-Tomasi corner detector [58] and then track those facial points using the Kanade-Lucas-Tomasi (KLT) tracker [45, 58, 66]. However, the tracking may accumulate errors over time because of changes in illumination due to the shadows of trees, traffic, etc. We therefore constrain the tracked facial points to follow a projective transformation and remove the incorrectly tracked points using the RANSAC algorithm. While tracking the facial points, we lose some of the tracked points with every new frame. To address this problem, we re-initialize the tracker with new discriminative facial points once the number of tracked points falls below a threshold [32].

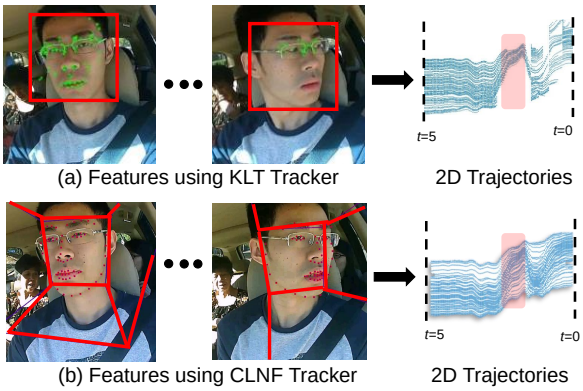


Fig. 9: **Improved features for maneuver anticipation.** We track facial landmark points using the CLNF tracker [4] which results in more consistent 2D trajectories as compared to the KLT tracker [58] used by Jain et al. [30]. Furthermore, the CLNF also gives an estimate of the driver's 3D head pose.

Head motion features. For maneuver anticipation the horizontal movement of the face and its angular rotation (*yaw*) are particularly important. From the face tracking we obtain *face tracks*, which are 2D trajectories of the tracked facial points in the image plane. Figure 8 (bottom) shows how the horizontal coordinates of the tracked facial points vary with time before a left turn maneuver. We represent the driver's face movements and rotations with histogram features. In particular, we take matching facial points between successive frames and create histograms of their corresponding horizontal motions (in pixels) and angular motions in the image plane (Figure 8). We bin the horizontal and angular motions using $[\leq -2,\ -2$ to $0,\ 0$ to $2,\ \geq 2]$ and $[0$ to $\frac{\pi}{2},\ \frac{\pi}{2}$ to $\pi,\ \pi$ to $\frac{3\pi}{2},\ \frac{3\pi}{2}$ to $2\pi]$, respectively. We also calculate the mean movement of the driver's face center. This gives us $\phi(\text{face}) \in \mathbb{R}^9$ facial features per-frame. The driver's eye-gaze is also useful a feature. However, robustly estimating 3D eye-gaze in outside environment is still a topic of research, and orthogonal to this work on anticipation. We therefore do not consider eye-gaze features.

3D head pose and facial landmark features. Our framework is flexible and allows incorporating more advanced face detection and tracking algorithms. For example we replace the KLT tracker described above with the Constrained Local Neural Field (CLNF) model [4] and track 68 fixed landmark points on the driver's face. CLNF is particularly well suited for driving scenarios due its ability to handle a wide range of head pose and illumination variations. As shown in Figure 9, CLNF offers us two distinct benefits over the features from KLT (i) while discriminative facial points may change from situation to situation, tracking fixed landmarks results in consistent optical flow trajectories which adds to robustness; and (ii) CLNF also allows us to estimate the 3D head pose of the driver's face by minimizing error in the projection of a generic 3D mesh model of the face w.r.t. the 2D location of landmarks in the image. The histogram features generated from the optical flow trajectories along with the 3D head pose features (yaw, pitch and row), give us $\phi(\text{face}) \in \mathbb{R}^{12}$ when using the CLNF tracker.

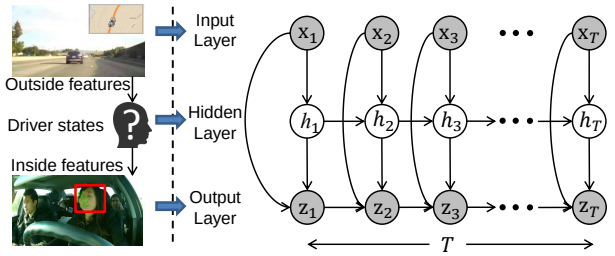


Fig. 10: **AIO-HMM.** The model has three layers: (i) Input (top): this layer represents outside vehicle features $\mathbf{x}$; (ii) Hidden (middle): this layer represents driver's latent states h; and (iii) Output (bottom): this layer represents inside vehicle features $\mathbf{z}$. This layer also captures temporal dependencies of inside vehicle features. T represents time.

Fig. 11: **Our data set** is diverse in drivers and landscape.

In Section VIII we present results with the features from KLT, as well as the results with richer features obtained from the CLNF model.

B. Outside-vehicle features.

The outside feature vector $\mathbf{x}_t$ encodes the information about the outside environment such as the road conditions, vehicle dynamics, etc. In order to get this information, we use the road-facing camera together with the vehicle's GPS coordinates, its speed, and the street maps. More specifically, we obtain two binary features from the road-facing camera indicating whether a lane exists on the left side and on the right side of the vehicle. We also augment the vehicle's GPS coordinates with the street maps and extract a binary feature indicating if the vehicle is within 15 meters of a road artifact such as intersections, turns, highway exists, etc. We also encode the average, maximum, and minimum speeds of the vehicle over the last 5 seconds as features. This results in a $\mathbf{x}_t \in \mathbb{R}^6$ dimensional feature vector.

VII. Bayesian networks for maneuver anticipation

In this section we propose alternate Bayesian networks [30] based on Hidden Markov Model (HMM) for maneuver anticipation. These models form a strong baseline to compare our sensory-fusion deep learning architecture.

Driving maneuvers are influenced by multiple interactions involving the vehicle, its driver, outside traffic, and occasionally global factors like the driver's destination. These interactions influence the driver's intention, i.e. their state of mind before the maneuver, which is not directly observable. In our Bayesian network formulation, we represent the driver's intention with discrete states that are *latent* (or hidden). In order to anticipate maneuvers, we jointly model the driving context and the *latent* states in a tractable manner. We represent the driving context as a set of features described in Section VI. We now present the motivation for the Bayesian networks and then discuss our key model Autoregressive Input-Output HMM (AIO-HMM).

A. Modeling driving maneuvers

Modeling maneuvers require temporal modeling of the driving context. Discriminative methods, such as the Support Vector Machine and the Relevance Vector Machine [65], which do not model the temporal aspect perform poorly on anticipation tasks, as we show in Section VIII. Therefore, a temporal model such as the Hidden Markov Model (HMM) is better suited to model maneuver anticipation.

An HMM models how the driver's *latent* states generate both the inside driving context ($\mathbf{z}_t$) and the outside driving context ($\mathbf{x}_t$). However, a more accurate model should capture how events *outside* the vehicle (i.e. the outside driving context) affect the driver's state of mind, which then generates the observations *inside* the vehicle (i.e. the inside driving context). Such interactions can be modeled by an Input-Output HMM (IOHMM) [7]. However, modeling the problem with IOHMM does not capture the temporal dependencies of the inside driving context. These dependencies are critical to capture the smooth and temporally correlated behaviours such as the driver's face movements. We therefore present Autoregressive Input-Output HMM (AIO-HMM) which extends IOHMM to model these observation dependencies. Figure 10 shows the AIO-HMM graphical model for modeling maneuvers. We learn separate AIO-HMM model for each maneuver. In order to anticipate maneuvers, during inference we determine which model best explains the past several seconds of the driving context based on the data log-likelihood. In Appendix we describe the training and inference procedure for AIO-HMM.

VIII. Experiments

In this section we first give an overview of our data set and then present the quantitative results. We also demonstrate our system and algorithm on real-world driving scenarios. **Our video demonstrations are available at**: http://www.brain4cars.com.

A. Driving data set

Our data set consists of natural driving videos with both inside and outside views of the car, its speed, and the global position system (GPS) coordinates.[2] The outside car video

[2]The inside and outside cameras operate at 25 and 30 frames/sec.

captures the view of the road ahead. We collected this driving data set under fully natural settings without any intervention.[3] It consists of 1180 miles of freeway and city driving and encloses 21,000 square miles across two states. We collected this data set from 10 drivers over a period of two months. The complete data set has a total of 2 million video frames and includes diverse landscapes. Figure 11 shows a few samples from our data set. We annotated the driving videos with a total of 700 events containing 274 lane changes, 131 turns, and 295 randomly sampled instances of driving straight. Each lane change or turn annotation marks the start time of the maneuver, i.e., before the car touches the lane or yaws, respectively. For all annotated events, we also annotated the lane information, i.e., the number of lanes on the road and the current lane of the car. Our data set is publicly available at http://www.brain4cars.com.

B. Baseline algorithms

We compare the following algorithms:

- *Chance*: Uniformly randomly anticipates a maneuver.
- *SVM* [50]: Support Vector Machine is a discriminative classifier [11]. Morris et al. [50] takes this approach for anticipating maneuvers.[4] We train the SVM on 5 seconds of driving context by concatenating all frame features to get a $\mathbb{R}^{3840}$ dimensional feature vector.
- *Random-Forest* [12]: This is also a discriminative classifier that learns many decision trees from the training data, and at test time it averages the prediction of the individual decision trees. We train it on the same features as SVM with 150 trees of depth ten each.
- *HMM*: This is the Hidden Markov Model. We train the HMM on a temporal sequence of feature vectors that we extract every 0.8 seconds, i.e., every 20 video frames. We consider three versions of the HMM: (i) HMM E: with only outside features from the road camera, the vehicle's speed, GPS and street maps (Section VI-B); (ii) HMM F: with only inside features from the driver's face (Section VI-A); and (ii) HMM $E + F$: with both inside and outside features.
- *IOHMM*: Jain et al. [30] modeled driving maneuvers with this Bayesian network. It is trained on the same features as HMM $E + F$.
- *AIO-HMM*: Jain et al. [30] proposed this Bayesian network for modeling maneuvers. It is trained on the same features as HMM $E + F$.
- *Simple-RNN* (S-RNN): In this architecture sensor streams are fused by simple concatenation and then passed through a single RNN with LSTM units.
- *Fusion-RNN-Uniform-Loss* (F-RNN-UL): In this architecture sensor streams are passed through separate RNNs, and the high-level representations from RNNs are then fused via a fully-connected layer. The loss at each time step takes the form $-\log(y_t^k)$.
- *Fusion-RNN-Exp-Loss* (F-RNN-EL): This architecture is similar to F-RNN-UL, except that the loss exponentially grows with time $-e^{-(T-t)}\log(y_t^k)$.

[3]**Protocol:** We set up cameras, GPS and speed recording device in subject's personal vehicles and left it to record the data. The subjects were asked to ignore our setup and drive as they would normally.

[4]Morries et al. [50] considered binary classification problem (lane change vs driving straight) and used RVM [65].

Algorithm 1 Maneuver anticipation

Initialize $m^* =$ *driving straight*
Input Features $\{(\mathbf{x}_1, ..., \mathbf{x}_T), (\mathbf{z}_1, ..., \mathbf{z}_T)\}$ and prediction threshold p_{th}
Output Predicted maneuver m^*
while $t = 1$ to T **do**
  Observe features $(\mathbf{x}_1, ..., \mathbf{x}_t)$ and $(\mathbf{z}_1, ..., \mathbf{z}_t)$
  Estimate probability $\mathbf{y}_t$ of each maneuver in $\mathcal{M}$
  $m_t^* = \arg\max_{m \in \mathcal{M}} \mathbf{y}_t$
  if $m_t^* \neq$ *driving straight* & $\mathbf{y}_t\{m_t^*\} > p_{th}$ **then**
    $m^* = m_t^*$
    break
  end if
end while
Return m^*

Our RNN and LSTM implementations are open-sourced and available at `NeuralModels` [27]. For the RNNs in our Fusion-RNN architecture we use a single layer LSTM of size 64 with sigmoid gate activations and tanh activation for hidden representation. Our fully connected fusion layer uses tanh activation and outputs a 64 dimensional vector. Our overall architecture (F-RNN-EL and F-RNN-UL) have nearly 25,000 parameters that are learned using RMSprop [13].

C. Evaluation protocol

We evaluate an algorithm based on its correctness in predicting future maneuvers. We anticipate maneuvers every 0.8 seconds where the algorithm processes the recent context and assigns a probability to each of the four maneuvers: {*left lane change, right lane change, left turn, right turn*} and a probability to the event of *driving straight*. These five probabilities together sum to one. After anticipation, i.e. when the algorithm has computed all five probabilities, the algorithm predicts a maneuver if its probability is above a threshold p_{th}. If none of the maneuvers' probabilities are above this threshold, the algorithm does not make a maneuver prediction and predicts *driving straight*. However, when it predicts one of the four maneuvers, it sticks with this prediction and makes no further predictions for next 5 seconds or until a maneuver occurs, whichever happens earlier. After 5 seconds or a maneuver has occurred, it returns to anticipating future maneuvers. Algorithm 1 shows the inference steps for maneuver anticipation.

During this process of anticipation and prediction, the algorithm makes (i) true predictions (tp): when it predicts the correct maneuver; (ii) false predictions (fp): when it predicts a maneuver but the driver performs a different maneuver; (iii) false positive predictions (fpp): when it predicts a maneuver

TABLE I: **Maneuver Anticipation Results.** Average *precision*, *recall* and *time-to-maneuver* are computed from 5-fold cross-validation. Standard error is also shown. Algorithms are compared on the features from Jain et al. [30].

Method	Lane change			Turns			All maneuvers		
	Pr (%)	Re (%)	Time-to-maneuver (s)	Pr (%)	Re (%)	Time-to-maneuver (s)	Pr (%)	Re (%)	Time-to-maneuver (s)
Chance	33.3	33.3	-	33.3	33.3	-	20.0	20.0	-
Morris et al. [50] SVM	73.7 ± 3.4	57.8 ± 2.8	2.40	64.7 ± 6.5	47.2 ± 7.6	2.40	43.7 ± 2.4	37.7 ± 1.8	1.20
Random-Forest	71.2 ± 2.4	53.4 ± 3.2	3.00	68.6 ± 3.5	44.4 ± 3.5	1.20	51.9 ± 1.6	27.7 ± 1.1	1.20
HMM E	75.0 ± 2.2	60.4 ± 5.7	3.46	74.4 ± 0.5	66.6 ± 3.0	4.04	63.9 ± 2.6	60.2 ± 4.2	3.26
HMM F	76.4 ± 1.4	75.2 ± 1.6	3.62	75.6 ± 2.7	60.1 ± 1.7	3.58	64.2 ± 1.5	36.8 ± 1.3	2.61
HMM $E+F$	80.9 ± 0.9	79.6 ± 1.3	3.61	73.5 ± 2.2	75.3 ± 3.1	4.53	67.8 ± 2.0	67.7 ± 2.5	3.72
IOHMM	81.6 ± 1.0	79.6 ± 1.9	3.98	77.6 ± 3.3	75.9 ± 2.5	4.42	74.2 ± 1.7	71.2 ± 1.6	3.83
(*Our final Bayesian network*) AIO-HMM	83.8 ± 1.3	79.2 ± 2.9	3.80	80.8 ± 3.4	75.2 ± 2.4	4.16	77.4 ± 2.3	71.2 ± 1.3	3.53
S-RNN	85.4 ± 0.7	86.0 ± 1.4	3.53	75.2 ± 1.4	75.3 ± 2.1	3.68	78.0 ± 1.5	71.1 ± 1.0	3.15
F-RNN-UL	**92.7** ± 2.1	84.4 ± 2.8	3.46	81.2 ± 3.5	78.6 ± 2.8	3.94	82.2 ± 1.0	75.9 ± 1.5	3.75
(*Our final deep architecture*) F-RNN-EL	88.2 ± 1.4	**86.0** ± 0.7	3.42	**83.8** ± 2.1	**79.9** ± 3.5	3.78	**84.5** ± 1.0	**77.1** ± 1.3	3.58

but the driver does not perform any maneuver (i.e. *driving straight*); and (iv) missed predictions (mp): when it predicts *driving straight* but the driver performs a maneuver. We evaluate the algorithms using their precision and recall scores:

$$Pr = \frac{tp}{\underbrace{tp + fp + fpp}_{\text{Total \# of maneuver predictions}}}; \quad Re = \frac{tp}{\underbrace{tp + fp + mp}_{\text{Total \# of maneuvers}}}$$

The precision measures the fraction of the predicted maneuvers that are correct and recall measures the fraction of the maneuvers that are correctly predicted. For true predictions (tp) we also compute the average *time-to-maneuver*, where time-to-maneuver is the interval between the time of algorithm's prediction and the start of the maneuver.

We perform cross validation to choose the number of the driver's latent states in the AIO-HMM and the threshold on probabilities for maneuver prediction. For *SVM* we cross-validate for the parameter C and the choice of kernel from Gaussian and polynomial kernels. The parameters are chosen as the ones giving the highest F1-score on a validation set. The F1-score is the harmonic mean of the precision and recall, defined as $F1 = 2 * Pr * Re/(Pr + Re)$.

D. Quantitative results

We evaluate the algorithms on maneuvers that were not seen during training and report the results using 5-fold cross validation. Table I reports the precision and recall scores under three settings: (i) *Lane change*: when the algorithms only predict for the left and right lane changes. This setting is relevant for highway driving where the prior probabilities of turns are low; (ii) *Turns*: when the algorithms only predict for the left and right turns; and (iii) *All maneuvers*: here the algorithms jointly predict all four maneuvers. All three settings include the instances of *driving straight*.

Table I compares the performance of the baseline anticipation algorithms, Bayesian networks, and the variants of our deep learning model. All algorithms in Table I use same feature vectors and KLT face tracker which ensures a fair comparison. As shown in the table, overall the best algorithm for maneuver anticipation is F-RNN-EL, and the best performing Bayesian network is AIO-HMM. F-RNN-EL significantly outperforms AIO-HMM in every setting. This improvement in performance is because RNNs with LSTM units are very expressive models with an internal memory. This allows them to model the much needed long temporal dependencies for anticipation. Additionally, unlike AIO-HMM, F-RNN-EL is a discriminative model that does not make any assumptions about the generative nature of the problem. The results also highlight the importance of modeling the temporal nature in the data. Classifiers like SVM and Random Forest do not model the temporal aspects and hence performs poorly.

The performance of several variants of our deep architecture, reported in Table I, justifies our design decisions to reach the final fusion architecture. When predicting all maneuvers, F-RNN-EL gives 6% higher precision and recall than S-RNN, which performs a simple fusion by concatenating the two sensor streams. On the other hand, F-RNN models each sensor stream with a separate RNN and then uses a fully connected layer to fuse the high-level representations at each time step. This form of sensory fusion is more principled since the sensor streams represent different data modalities. In addition, exponentially growing the loss further improves the performance. Our new loss scheme penalizes the network proportional to the length of context it has seen. When predicting all maneuvers, we observe that F-RNN-EL shows an improvement of 2% in precision and recall over F-RNN-UL. We conjecture that exponentially growing the loss acts like a regularizer. It reduces the risk of our network overfitting early in time when there is not enough context available. Furthermore, the time-to-maneuver remains comparable for F-RNN with and without exponential loss.

The Bayesian networks AIO-HMM and HMM $E+F$ adopt different sensory fusion strategies. AIO-HMM fuses the two sensory streams using an input-output model, on the other hand HMM $E + F$ performs early fusion by concatenation. As a result, AIO-HMM gives 10% higher precision than HMM $E + F$ for jointly predicting all the maneuvers. AIO-HMM further extends IOHMM by modeling the temporal dependencies of events inside the vehicle. This results in better performance: on average AIO-HMM precision is 3% higher than IOHMM, as shown in Table I. Another important aspect of anticipation is the joint modeling of the inside and outside driving contexts. HMM F learns only from the inside driving context, while HMM E learns only from the outside driving context. The performances of both the models is therefore less than HMM $E + F$, which learns jointly both the contexts.

Table II compares the fpp of different algorithms. False positive predictions (fpp) happen when an algorithm predicts

TABLE II: **False positive prediction** (fpp) of different algorithms. The number inside parenthesis is the standard error.

Algorithm	Lane change	Turns	All
Morris et al. [50] SVM	15.3 (0.8)	13.3 (5.6)	24.0 (3.5)
Random-Forest	16.2 (3.3)	12.9 (3.7)	17.5 (4.0)
HMM E	36.2 (6.6)	33.3 (0.0)	63.8 (9.4)
HMM F	23.1 (2.1)	23.3 (3.1)	11.5 (0.1)
HMM $E+F$	30.0 (4.8)	21.2 (3.3)	40.7 (4.9)
IOHMM	28.4 (1.5)	25.0 (0.1)	40.0 (1.5)
AIO-HMM	24.6 (1.5)	20.0 (2.0)	30.7 (3.4)
S-RNN	16.2 (1.3)	16.7 (0.0)	19.2 (0.0)
F-RNN-UL	19.2 (2.4)	25.0 (2.4)	21.5 (2.1)
F-RNN-EL	10.8 (0.7)	23.3 (1.5)	27.7 (3.8)

a maneuver but the driver does not perform any maneuver (i.e. drives straight). Therefore low value of fpp is preferred. HMM F performs best on this metric at 11% as it mostly assigns a high probability to *driving straight*. However, due to this reason, it incorrectly predicts *driving straight* even when maneuvers happen. This results in the low recall of HMM F at 36%, as shown in Table I. AIO-HMM's fpp is 10% less than that of IOHMM and HMM $E+F$, and F-RNN-EL is 3% less than AIO-HMM. The primary reason for false positive predictions is distracted driving. Drivers interactions with fellow passengers or their looking at the surrounding scenes are sometimes wrongly interpreted by the algorithms. Understanding driver distraction is still an open problem, and orthogonal to the objective of this work.

TABLE III: **3D head-pose features.** In this table we study the effect of better features with best performing algorithm from Table I in 'All maneuvers' setting. We use [4] to track 68 facial landmark points and estimate 3D head-pose.

Method	Pr (%)	Re (%)	Time-to-maneuver (s)
F-RNN-EL	84.5 $\pm$ 1.0	77.1 $\pm$ 1.3	3.58
F-RNN-EL w/ 3D head-pose	**90.5** $\pm$ 1.0	**87.4** $\pm$ 0.5	3.16

3D head-pose features. The modularity of our approach allows experimenting with more advanced head tracking algorithms. We replace the pipeline for extracting features from the driver's face [30] by a Constrained Local Neural Field (CLNF) model [4]. The new vision pipeline tracks 68 facial landmark points and estimates the driver's 3D head pose as described in Section VI. As shown in Table III, we see a significant, 6% increase in precision and 10% increase in recall of F-RNN-EL when using features from our new vision pipeline. This increase in performance is attributed to the following reasons: (i) robustness of CLNF model to variations in illumination and head pose; (ii) 3D head-pose features are very informative for understanding the driver's intention; and (iii) optical flow trajectories generated by tracking facial landmark points represent head movements better, as shown in Figure 9. The confusion matrix in Figure 13 shows the precision for each maneuver. F-RNN-EL gives a higher precision than AIO-HMM on every maneuver when both algorithms are trained on same features (Fig. 13c). The new vision pipeline with CLNF tracker further improves the precision of F-RNN-EL on all maneuvers (Fig. 13d).

Effect of prediction threshold. In Figure 12 we study how F1-score varies as we change the prediction threshold p_{th}.

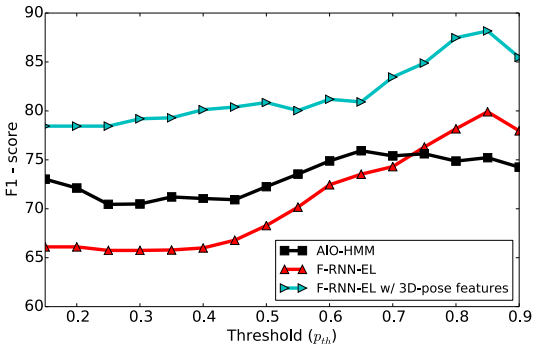


Fig. 12: **Effect of prediction threshold p_{th}.** At test time an algorithm makes a prediction only when it is at least p_{th} confident in its prediction. This plot shows how F1-score vary with change in prediction threshold.

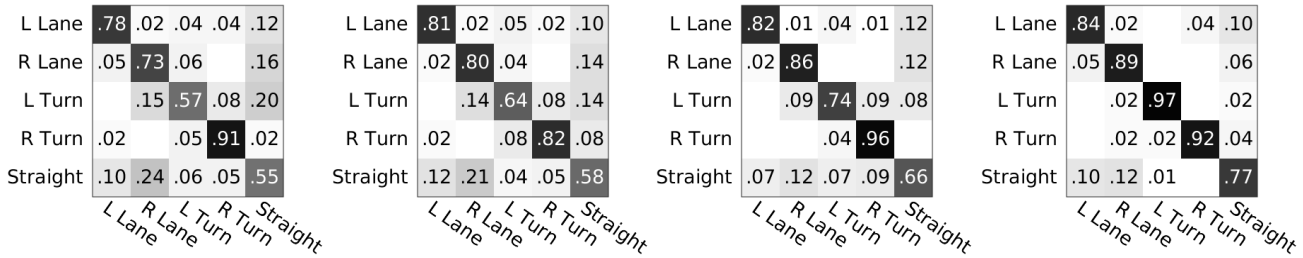
We make the following observations: (i) The F1-score does not undergo large variations with changes to the prediction threshold. Hence, it allows practitioners to fairly trade-off between the precision and recall without hurting the F1-score by much; and (ii) the maximum F1-score attained by F-RNN-EL is 4% more than AIO-HMM when compared on the same features and 13% more with our new vision pipeline. In Tables I, II and III, we used the threshold values which gave the highest F1-score.

Anticipation complexity. The F-RNN-EL anticipates maneuvers every 0.8 seconds using the previous 5 seconds of the driving context. The complexity mainly comprises of feature extraction and the model inference in Algorithm 1. Fortunately both these steps can be performed as a dynamic program by storing the computation of the most recent anticipation. Therefore, for every anticipation we only process the incoming 0.8 seconds and not complete 5 seconds of the driving context. On average we predict a maneuver under 0.20 milliseconds using Theano [5] on Nvidia K40 GPU on Ubuntu 12.04.

IX. Conclusion

In this paper we considered the problem of anticipating driving maneuvers a few seconds before the driver performs them. This problem requires the modeling of long temporal dependencies and the fusion of multiple sensory streams. We proposed a novel deep learning architecture based on Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units for anticipation. Our architecture learns to fuse multiple sensory streams, and by training it in a sequence-to-sequence prediction manner, it explicitly learns to anticipate using only a partial temporal context. We also proposed a novel loss layer for anticipation which prevents over-fitting.

We release an open-source data set of 1180 miles of natural driving. We performed an extensive evaluation and showed improvement over many baseline algorithms. Our sensory fusion deep learning approach gives a precision of 84.5% and recall of 77.1%, and anticipates maneuvers 3.5 seconds (on average) before they happen. By incorporating the driver's 3D head-pose our precision and recall improves to 90.5%



(a) IOHMM *(b)* AIO-HMM *(c)* F-RNN-EL *(d)* F-RNN-EL w/ 3D-pose

Fig. 13: **Confusion matrix** of different algorithms when jointly predicting all the maneuvers. Predictions made by algorithms are represented by rows and actual maneuvers are represented by columns. Numbers on the diagonal represent precision.

and 87.4% respectively. Potential application of our work is enabling advanced driver assistance systems (ADAS) to alert drivers before they perform a dangerous maneuver, thereby giving drivers more time to react. We believe that our deep learning architecture is widely applicable to many activity anticipation problems. Our code and data set are publicly available on the project web-page.

Acknowledgement. We thank NVIDIA for the donation of K40 GPUs used in this research. We also thank Silvio Savarese for useful discussions. This work was supported by National Robotics Initiative (NRI) award 1426452, Office of Naval Research (ONR) award N00014-14-1-0156, and by Microsoft Faculty Fellowship and NSF Career Award to Saxena.

APPENDIX A
MODELING MANEUVERS WITH AIO-HMM

Given T seconds long driving context $\mathcal{C}$ before the maneuver $\mathcal{M}$, we learn a generative model for the context $P(\mathcal{C}|\mathcal{M})$. The driving context $\mathcal{C}$ consists of the outside driving context and the inside driving context. The outside and inside contexts are temporal sequences represented by the outside features $\mathbf{x}_1^T = \{\mathbf{x}_1, .., \mathbf{x}_T\}$ and the inside features $\mathbf{z}_1^T = \{\mathbf{z}_1, .., \mathbf{z}_T\}$ respectively. The corresponding sequence of the driver's latent states is $h_1^T = \{h_1, .., h_T\}$. $\mathbf{x}$ and $\mathbf{z}$ are vectors and h is a discrete state.

$$\begin{aligned} P(\mathcal{C}|\mathcal{M}) &= \sum_{h_1^T} P(\mathbf{z}_1^T, \mathbf{x}_1^T, h_1^T|\mathcal{M}) \\ &= P(\mathbf{x}_1^T|\mathcal{M}) \sum_{h_1^T} P(\mathbf{z}_1^T, h_1^T|\mathbf{x}_1^T, \mathcal{M}) \\ &\propto \sum_{h_1^T} P(\mathbf{z}_1^T, h_1^T|\mathbf{x}_1^T, \mathcal{M}) \end{aligned} \tag{14}$$

We model the correlations between $\mathbf{x}$, h and $\mathbf{z}$ with an AIO-HMM as shown in Figure 10. The AIO-HMM models the distribution in equation (14). It does not assume any generative process for the outside features $P(\mathbf{x}_1^T|\mathcal{M})$. It instead models them in a discriminative manner. The top (input) layer of the AIO-HMM consists of outside features $\mathbf{x}_1^T$. The outside features then affect the driver's latent states h_1^T, represented by the middle (hidden) layer, which then generates the inside features $\mathbf{z}_1^T$ at the bottom (output) layer. The events inside the vehicle such as the driver's head movements are temporally correlated because they are generally smooth. The AIO-HMM handles these dependencies with autoregressive connections in the output layer.

Model Parameters. AIO-HMM has two types of parameters: (i) state transition parameters $\mathbf{w}$; and (ii) observation emission parameters ($\boldsymbol{\mu}$,$\boldsymbol{\Sigma}$). We use set $\mathcal{S}$ to denote the possible latent states of the driver. For each state $h = i \in \mathcal{S}$, we parametrize transition probabilities of leaving the state with log-linear functions, and parametrize the output layer feature emissions with normal distributions.

$$\text{Transition: } P(h_t = j|h_{t-1} = i, \mathbf{x}_t; \mathbf{w}_{ij}) = \frac{e^{\mathbf{w}_{ij}\cdot\mathbf{x}_t}}{\sum_{l\in\mathcal{S}} e^{\mathbf{w}_{il}\cdot\mathbf{x}_t}}$$

$$\text{Emission: } P(\mathbf{z}_t|h_t = i, \mathbf{x}_t, \mathbf{z}_{t-1}; \boldsymbol{\mu}_{it}, \boldsymbol{\Sigma}_i) = \mathcal{N}(\mathbf{z}_t|\boldsymbol{\mu}_{it}, \boldsymbol{\Sigma}_i)$$

The inside (vehicle) features represented by the output layer are jointly influenced by all three layers. These interactions are modeled by the mean and variance of the normal distribution. We model the mean of the distribution using the outside and inside features from the vehicle as follows:

$$\boldsymbol{\mu}_{it} = (1 + \mathbf{a}_i \cdot \mathbf{x}_t + \mathbf{b}_i \cdot \mathbf{z}_{t-1})\boldsymbol{\mu}_i$$

In the equation above, $\mathbf{a}_i$ and $\mathbf{b}_i$ are parameters that we learn for every state $i \in \mathcal{S}$. Therefore, the parameters we learn for state $i \in \mathcal{S}$ are $\boldsymbol{\theta}_i = \{\boldsymbol{\mu}_i, \mathbf{a}_i, \mathbf{b}_i, \boldsymbol{\Sigma}_i$ and $\mathbf{w}_{ij}|j \in \mathcal{S}\}$, and the overall model parameters are $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i|i \in \mathcal{S}\}$.

A. Learning AIO-HMM parameters

The training data $\mathcal{D} = \{(\mathbf{x}_{1,n}^{T_n}, \mathbf{z}_{1,n}^{T_n})|n = 1, .., N\}$ consists of N instances of a maneuver $\mathcal{M}$. The goal is to maximize the data log-likelihood.

$$l(\boldsymbol{\Theta}; \mathcal{D}) = \sum_{n=1}^{N} \log P(\mathbf{z}_{1,n}^{T_n}|\mathbf{x}_{1,n}^{T_n}; \boldsymbol{\Theta}) \tag{15}$$

Directly optimizing equation (15) is challenging because parameters h representing the driver's states are *latent*. We therefore use the iterative EM procedure to learn the model parameters. In EM, instead of directly maximizing equation (15), we maximize its simpler lower bound. We estimate the lower bound in the E-step and then maximize that estimate in the M-step. These two steps are repeated iteratively.

E-step. In the E-step we get the lower bound of equation (15) by calculating the expected value of the *complete* data log-likelihood using the current estimate of the parameter $\hat{\boldsymbol{\Theta}}$.

$$\text{E-step: } Q(\boldsymbol{\Theta}; \hat{\boldsymbol{\Theta}}) = E[l_c(\boldsymbol{\Theta}; \mathcal{D}_c)|\hat{\boldsymbol{\Theta}}, \mathcal{D}] \tag{16}$$

where $l_c(\mathbf{\Theta}; \mathcal{D}_c)$ is the log-likelihood of the *complete* data $\mathcal{D}_c$ defined as:

$$\mathcal{D}_c = \{(\mathbf{x}_{1,n}^{T_n}, \mathbf{z}_{1,n}^{T_n}, h_{1,n}^{T_n}) | n = 1, .., N\} \tag{17}$$

$$l_c(\mathbf{\Theta}; \mathcal{D}_c) = \sum_{n=1}^{N} \log P(\mathbf{z}_{1,n}^{T_n}, h_{1,n}^{T_n} | \mathbf{x}_{1,n}^{T_n}; \mathbf{\Theta}) \tag{18}$$

We should note that the occurrences of hidden variables h in $l_c(\mathbf{\Theta}; \mathcal{D}_c)$ are marginalized in equation (16), and hence h need not be known. We efficiently estimate $Q(\mathbf{\Theta}; \hat{\mathbf{\Theta}})$ using the forward-backward algorithm [51].

M-step. In the M-step we maximize the expected value of the complete data log-likelihood $Q(\mathbf{\Theta}; \hat{\mathbf{\Theta}})$ and update the model parameter as follows:

$$\text{M-step: } \mathbf{\Theta} = \arg\max_{\mathbf{\Theta}} Q(\mathbf{\Theta}; \hat{\mathbf{\Theta}}) \tag{19}$$

Solving equation (19) requires us to optimize for the parameters $\boldsymbol{\mu}$, $\mathbf{a}$, $\mathbf{b}$, $\mathbf{\Sigma}$ and $\mathbf{w}$. We optimize all parameters expect $\mathbf{w}$ exactly by deriving their closed form update expressions. We optimize $\mathbf{w}$ using the gradient descent.

B. Inference of Maneuvers

Our learning algorithm trains separate AIO-HMM models for each maneuver. The goal during inference is to determine which model best explains the past T seconds of the driving context not seen during training. We evaluate the likelihood of the inside and outside feature sequences ($\mathbf{z}_1^T$ and $\mathbf{x}_1^T$) for each maneuver, and anticipate the probability $P_{\mathcal{M}}$ of each maneuver $\mathcal{M}$ as follows:

$$P_{\mathcal{M}} = P(\mathcal{M} | \mathbf{z}_1^T, \mathbf{x}_1^T) \propto P(\mathbf{z}_1^T, \mathbf{x}_1^T | \mathcal{M}) P(\mathcal{M}) \tag{20}$$

Algorithm 2 shows the complete inference procedure. The inference in equation (20) simply requires a forward-pass [51] of the AIO-HMM, the complexity of which is $\mathcal{O}(T(|\mathcal{S}|^2 + |\mathcal{S}||\mathbf{z}|^3 + |\mathcal{S}||\mathbf{x}|))$. However, in practice it is only $\mathcal{O}(T|\mathcal{S}||\mathbf{z}|^3)$ because $|\mathbf{z}|^3 \gg |S|$ and $|\mathbf{z}|^3 \gg |\mathbf{x}|$. Here $|\mathcal{S}|$ is the number of discrete states representing the driver's intention, while $|\mathbf{z}|$ and $|\mathbf{x}|$ are the dimensions of the inside and outside feature vectors respectively. In equation (20) $P(\mathcal{M})$ is the prior probability of maneuver $\mathcal{M}$. We assume an uninformative uniform prior over the maneuvers.

Algorithm 2 Anticipating maneuvers

input Driving videos, GPS, Maps and Vehicle Dynamics
output Probability of each maneuver
  Initialize the face tracker with the driver's face
  while $driving$ **do**
    Track the driver's face [69]
    Extract features $\mathbf{z}_1^T$ and $\mathbf{x}_1^T$ (Sec. VI)
    Inference $P_{\mathcal{M}} = P(\mathcal{M} | \mathbf{z}_1^T, \mathbf{x}_1^T)$ (Eq. (20))
    Send the inferred probability of each maneuver to ADAS
  end while

REFERENCES

[1] Bosch urban. http://bit.ly/1feM3JM. Accessed: 2015-04-23.

[2] 2012 motor vehicle crashes: overview. *N. Highway Traffic Safety Administration, Washington, D.C., Tech. Rep.*, 2013.

[3] A. Andreas, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[4] T. Baltrusaitis, P. Robinson, and L-P. Morency. Constrained local neural fields for robust facial landmark detection in the wild. In *ICCV Workshop*, 2013.

[5] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[6] Y. Bengio and O. Delalleau. On the expressive power of deep architectures. In *Algorithmic Learning Theory*, pages 18–36, 2011.

[7] Y. Bengio and O. Frasconi. An input output hmm architecture. *Advances in Neural Information Processing Systems*, 1995.

[8] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun. Learning motion patterns of people for compliant robot motion. *International Journal of Robotics Research*, 2005.

[9] H. Berndt, J. Emmert, and K. Dietmayer. Continuous driver intention recognition with hidden markov models. In *IEEE Intelligent Transportation Systems Conference*, 2008.

[10] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6), 2001.

[11] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3), 1995.

[12] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. *MSR TR*, 5(6), 2011.

[13] Y. N. Dauphin, H. de Vries, J. Chung, and Y. Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *arXiv:1502.04390*, 2015.

[14] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[15] A. Doshi, B. Morris, and M. M. Trivedi. On-road prediction of driver's intent with multimodal sensory cues. *IEEE Pervasive Computing*, 2011.

[16] A. Dragan and S. Srinivasa. Formalizing assistive teleoperation. In *Proceedings of Robotics: Science and Systems*, 2012.

[17] A. Dragan and S. Srinivasa. Generating legible motion. In *Proceedings of Robotics: Science and Systems*, 2013.

[18] K. Driggs-Campbell, V. Shia, and R. Bajcsy. Improved driver modeling for human-in-the-loop vehicular con-

trol. In *Proceedings of the International Conference on Robotics and Automation*, 2015.

[19] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[20] L. Fletcher, N. Apostoloff, L. Petersson, and A. Zelinsky. Vision in and out of vehicles. *IEEE IS*, 18(3), 2003.

[21] L. Fletcher, G. Loy, N. Barnes, and A. Zelinsky. Correlating driver gaze with the road scene for driver assistance systems. *Robotics and Autonomous Systems*, 52(1), 2005.

[22] B. Frohlich, M. Enzweiler, and U. Franke. Will this car change the lane?turn signal recognition in the frequency domain. In *IEEE International Vehicle Symposium Proceedings*, 2014.

[23] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, and A. Coates. Deepspeech: Scaling up end-to-end speech recognition. *arXiv:1412.5567*, 2014.

[24] S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *NIPS*, 1995.

[25] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8), 1997.

[26] M. E. Jabon, J. N. Bailenson, E. Pontikakis, L. Takayama, and C. Nass. Facial expression analysis for predicting unsafe driving behavior. *IEEE Pervasive Computing*, (4), 2010.

[27] A. Jain. Neuralmodels. https://github.com/asheshjain399/NeuralModels, 2015.

[28] A. Jain, S. Sharma, and A. Saxena. Beyond geometric path planning: Learning context-driven user preferences via sub-optimal feedback. In *Proceedings of the International Symposium on Robotics Research*, 2013.

[29] A. Jain, D. Das, J. Gupta, and A. Saxena. Planit: A crowdsourcing approach for learning to plan paths from large scale preference feedback. In *Proceedings of the International Conference on Robotics and Automation*, 2015.

[30] A. Jain, H. S. Koppula, B. Raghavan, S. Soh, and A. Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *ICCV*, 2015.

[31] A. Jain, S. Sharma, T. Joachims, and A. Saxena. Learning preferences for manipulation tasks from online coactive feedback. *International Journal of Robotics Research*, 2015.

[32] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Proceedings of the International Conference on Pattern Recognition*, 2010.

[33] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *Proceedings of the European Conference on Computer Vision*. 2012.

[34] H. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *Proceedings of Robotics: Science and Systems*, 2013.

[35] H. Koppula and A. Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *Proceedings of the International Conference on Machine Learning*, 2013.

[36] H. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[37] H. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from rgb-d videos. *International Journal of Robotics Research*, 32(8), 2013.

[38] H. Koppula, A. Jain, and A. Saxena. Anticipatory planning for humanrobot teams. In *ISER*, 2014.

[39] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Proceedings of Robotics: Science and Systems*, 2012.

[40] N. Kuge, T. Yamamura, O. Shimoyama, and A. Liu. A driver behavior recognition method based on a driver model framework. Technical report, SAE Technical Paper, 2000.

[41] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier. Learning-based approach for online lane change intention prediction. In *IEEE International Vehicle Symposium Proceedings*, 2013.

[42] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, 2001.

[43] C. Laugier, I. E. Paromtchik, M. Perrollaz, MY. Yong, J-D. Yoder, C. Tay, K. Mekhnacha, and A. Negre. Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety. *ITS Magazine, IEEE*, 3(4), 2011.

[44] M. Liebner, M. Baumann, F. Klanner, and C. Stiller. Driver intent inference at urban intersections using the intelligent driver model. In *IEEE International Vehicle Symposium Proceedings*, 2012.

[45] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1981.

[46] J. Mainprice and D. Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 2013.

[47] I. Matthews and S. Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60 (2), 2004.

[48] D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *Proceedings of Robotics: Science and Systems*, 2014.

[49] L. Morency, A. Quattoni, and T. Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[50] B. Morris, A. Doshi, and M. Trivedi. Lane change

intent prediction for driver assistance: On-road design and evaluation. In *IEEE International Vehicle Symposium Proceedings*, 2011.

[51] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[52] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of the International Conference on Machine Learning*, 2011.

[53] N. Oliver and A. P. Pentland. Graphical models for driver behavior recognition in a smartcar. In *IEEE International Vehicle Symposium Proceedings*, 2000.

[54] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *arXiv:1211.5063*, 2012.

[55] M. Rezaei and R. Klette. Look at the driver, look at the road: No distraction! no accident! In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[56] T. Rueda-Domingo, P. Lardelli-Claret, J. Luna del Castillo, J. Jimenez-Moleon, M. Garcia-Martin, and A. Bueno-Cavanillas. The influence of passengers on the risk of the driver causing a car collision in spain: Analysis of collisions from 1990 to 1999. *Accident Analysis & Prevention*, 2004.

[57] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Proceedings of the International Conference on Computer Vision*, 2011.

[58] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[59] V. Shia, Y. Gao, R. Vasudevan, K. D. Campbell, T. Lin, F. Borrelli, and R. Bajcsy. Semiautonomous vehicular control using driver modeling. *IEEE Transactions on Intelligent Transportation Systems*, 15(6), 2014.

[60] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 2007.

[61] J. Sung, S. H. Jin, and A. Saxena. Robobarista: Object part-based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *Proceedings of the International Symposium on Robotics Research*, 2015.

[62] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014.

[63] A. Tawari, S. Sivaraman, M. Trivedi, T. Shannon, and M. Tippelhofer. Looking-in and looking-out vision for urban intelligent assistance: Estimation of driver attentive state and dynamic surround for safe merging and braking. In *IEEE IVS*, 2014.

[64] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT press, 2005.

[65] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1, 2001.

[66] C. Tomasi and T. Kanade. Detection and tracking of point features. *International Journal of Computer Vision*, 1991.

[67] M. Trivedi, T. Gandhi, and J. McCall. Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety. *IEEE Transactions on Intelligent Transportation Systems*, 8(1), 2007.

[68] R. Vasudevan, V. Shia, Y. Gao, R. Cervera-Navarro, R. Bajcsy, and F. Borrelli. Safe semi-autonomous control with enhanced driver modeling. In *American Control Conference*, 2012.

[69] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2), 2004.

[70] H. Wang and C. Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[71] S. B. Wang, A. Quattoni, L. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[72] Y. Wang and Q. Ji. A dynamic conditional random field model for object segmentation in image sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[73] Z. Wang, K. Mülling, M. Deisenroth, H. Amor, D. Vogt, B. Schölkopf, and J. Peters. Probabilistic movement modeling for intention inference in human-robot interaction. *International Journal of Robotics Research*, 2013.

[74] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[75] X. Xiong and F. De la Torre. Supervised descent method for solving nonlinear least squares problems in computer vision. *arXiv preprint arXiv:1405.0601*, 2014.

[76] C. Zhang and Z. Zhang. A survey of recent advances in face detection. Technical report, Microsoft Research, 2010.

[77] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 2009.