Recurrent Neural Networks for Driver Activity Anticipation via Sensory-Fusion Architecture

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

***Abstract*— Anticipating the future actions of a human is a widely studied problem in robotics that requires spatio-temporal reasoning. In this work we propose a deep learning approach for anticipation in sensory-rich robotics applications. We introduce a sensory-fusion architecture which jointly learns to anticipate and fuse information from 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 train our architecture in a sequence-to-sequence prediction manner, and it *explicitly* learns to predict the future given only a partial temporal context. We further introduce a novel loss layer for anticipation which prevents over-fitting and encourages early anticipation. We use our architecture to anticipate driving maneuvers several seconds before they happen on a natural driving data set of 1180 miles. The context for maneuver anticipation comes from multiple sensors installed on the vehicle. Our approach shows significant improvement over the state-of-the-art in maneuver anticipation by increasing the precision from 77.4% to 90.5% and recall from 71.2% to 87.4%.**

I. Introduction

Anticipating the future actions of a human is an important perception task and has many applications in robotics. It has enabled robots to navigate in a social manner and perform collaborative tasks with humans while avoiding conflicts [24, 41, 21, 40]. In another application, anticipating driving maneuvers several seconds in advance [19, 27, 38, 36] enables assistive cars to alert drivers before they make a dangerous maneuver. Maneuver anticipation complements existing Advance Driver Assistance Systems (ADAS) by giving drivers more time to react to road situations and thereby can prevent many accidents [30].

Activity anticipation is a challenging problem because it requires the prediction of future events from a limited temporal context. It is different from *activity recognition* [40], where the complete temporal context is available for prediction. Furthermore, in sensory-rich robotics settings, 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 [20, 21, 24] usually deal with single-data modality and do not address anticipation for sensory-rich robotics applications. Additionally, they learn representations using shallow architectures [19, 20, 21, 24] that cannot handle long temporal dependencies [4].

In order to address the anticipation problem more generally, 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 1) 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.

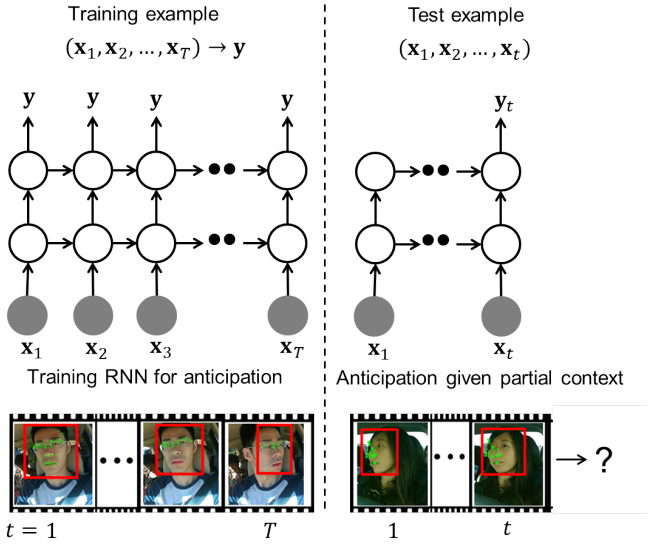


Fig. 1: (**Left**) Shows training RNN for anticipation in a sequence-to-sequence prediction manner. The network explicitly learns to map the partial context $(\mathbf{x}_1, .., \mathbf{x}_t)$ $\forall t$ to the future event $\mathbf{y}$. (**Right**) At test time the network's goal is to anticipate the future event as soon as possible, i.e. by observing only a partial temporal context.

We evaluate our approach on the task of anticipating driving maneuvers several seconds before they happen [19, 27]. The *context* (contextual information) for maneuver anticipation comes from multiple sensors installed on the vehicle such as cameras, GPS, vehicle dynamics, etc. Information from each of these sensory streams provides necessary cues for predicting future maneuvers. Our overall architecture models each sensory stream with an RNN and then non-linearly combines the high-level representations from multiple RNNs to make a final prediction.

We report results on 1180 miles of natural driving data collected from 10 drivers [19]. The data set is challenging because of the variations in routes and traffic conditions, and the driving styles of the drivers (Figure 2). On this data set, our deep learning approach improves the state-of-the-art in maneuver anticipation by increasing the precision from 77.4% to **84.5%** and recall from 71.2% to **77.1%**. We

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

further improved these results by extracting richer features from cameras such as the 3D head pose of the driver's face. Including these features into our architecture increases the precision and recall to **90.5%** and **87.4%** respectively. Key contributions of this paper are:

- A sensory-fusion RNN-LSTM architecture for anticipation in sensory-rich robotics applications.
- A new vision pipeline with rich features (such as 3D head pose) for maneuver anticipation.
- State-of-the-art performance on maneuver anticipation on 1180 miles of driving data [19].

II. Related Work

Our work is related to previous works on anticipating human activities, driver behavior understanding, and Recurrent Neural Networks (RNNs) for sequence prediction.

Several works have studied human activity anticipation for human-robot collaboration and forecasting. Anticipating human activities has been shown to improve human-robot collaboration [40, 21, 25, 10]. Similarly, forecasting human navigation trajectories has enabled robots to plan sociable trajectories around humans [20, 6, 24]. Feature matching techniques have been proposed for anticipating human activities from videos [31]. Approaches used in these works learn shallow architectures [4] that do not properly model temporal aspects of human activities. Furthermore, they deal with a single data modality and do not tackle the challenges of sensory-fusion. We propose a deep learning approach for anticipation which efficiently handles temporal dependencies and learns to fuse multiple sensory streams.

We demonstrate our approach on anticipating driving maneuvers several seconds before they happen. This is a sensor-rich application for alerting drivers several seconds before they make a dangerous maneuvering decision. Previous works have addressed maneuver anticipation [1, 19, 27, 9, 37] through sensory-fusion from multiple cameras, GPS, and vehicle dynamics. In particular, Morris et al. [27] and Trivedi et al. [37] used a Relevance Vector Machine (RVM) for intent prediction and performed sensory fusion by concatenating feature vectors.

More recently, Jain et al. [19] showed that concatenation of sensory streams does not capture the rich context for modeling maneuvers. They proposed an Autoregressive Input-Output Hidden Markov Model (AIO-HMM) which fuses sensory streams through a linear transformation of features and it performs better than feature concatenation [27]. In contrast, we learn an expressive architecture to combine information from multiple sensors. 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 [17, 14, 13, 9], and vehicular controllers which act on these predictions [33, 38, 11].

Two building blocks of our architecture are Recurrent Neural Networks (RNNs) [29] and Long Short-Term Memory (LSTM) units [16]. Our work draws upon ideas from previous works on RNNs and LSTM from the language [35], speech [15], and vision [8] communities. Our approach to the joint training of multiple RNNs is related to the recent work on hierarchical RNNs [12]. We consider RNNs in multi-modal setting, which is related to the recent use of RNNs in image-captioning [8]. 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 on sensory-fusion from feed-forward networks [28, 34] to the fusion of temporal streams. Using our architecture we demonstrate state-of-the-art results on maneuver anticipation.

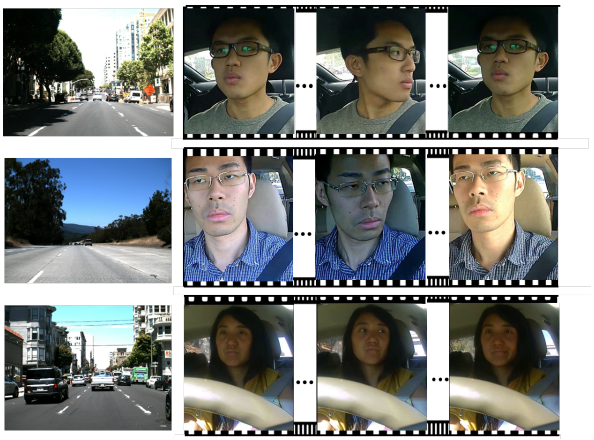


Fig. 2: **Variations in the data set.** Images from the data set [19] for a left lane change. (**Left**) Views from the road facing camera. (**Right**) Driving style of the drivers vary for the same maneuver.

III. 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* [39, 22] 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 [16] for anticipation. Below we give an overview of the standard RNN and LSTM which form the building blocks of our architecture described in Section IV.

A. Recurrent Neural Networks

A standard RNN [29] 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.

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

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

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* [29], 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 [16]. 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 [16]. 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 3 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:

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

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

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

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

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

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 matrices $\mathbf{V}_*$ are daigonal. 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.

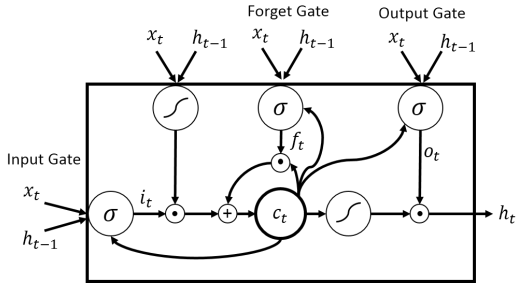


Fig. 3: **Internal working of an LSTM unit.**

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

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.

IV. 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 [21, 27, 31] 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 how 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 [2] or robots that jointly reason over perception and language instructions [26]. In such applications the way information from different sensors is fused is critical to the application's final performance. For example Jain et al. [19] showed that for maneuver anticipation, learning a simple transformation of the sensory streams works better than direct concatenation of those streams. 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. 1. 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

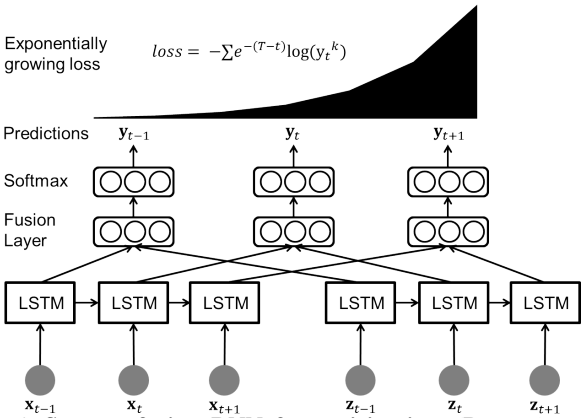


Fig. 4: **Sensory fusion RNN for anticipation.** (**Bottom**) In the Fusion-RNN each sensory stream is passed through their independent RNN. (**Middle**) High-level representations from RNNs are then combined through a fusion layer. (**Top**) In order to prevent over-fitting early in time the loss exponentially increases with time.

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 Sections V and VI, we use the fusion architecture 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 $\{(\mathbf{h}^x_1, ..., \mathbf{h}^x_T),\ (\mathbf{h}^z_1, ..., \mathbf{h}^z_T)$ 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 4. 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.

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

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

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

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

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 $-\log(y^k_t)$ for the mistakes it makes at each time step, where y^k_t 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 4. Under this new scheme, the loss exponentially grows with time as shown below.

$$loss = \sum_{j=1}^{N} \sum_{t=1}^{T} -e^{-(T-t)} \log(y^k_t) \tag{13}$$

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.

D. Model training and data augmentation

Our architecture for maneuver anticipation has more than 25,000 parameters that need to be learned (Section V). With such a large number of parameters on a non-convex manifold, over-fitting becomes a major challenge. We therefore introduce redundancy in the training data which acts like a regularizer and reduces over-fitting [23, 15]. In order to augment training data, we extract sub-sequences of temporal observations. Given a training example with two temporal sensor streams $\{(\mathbf{x}_1, ..., \mathbf{x}_T), (\mathbf{z}_1, ..., \mathbf{z}_T), \mathbf{y}\}$, we uniformly randomly sample multiple sub-sequences $\{(\mathbf{x}_i, ..., \mathbf{x}_j), (\mathbf{z}_i, ..., \mathbf{z}_j), \mathbf{y} | 1 \leq i < j \leq T\}$ as additional training examples. It is important to note that data augmentation only adds redundancy and *does not* rely on any external source of new information.

On the augmented data set, we train the network described in Section IV-B. We use RMSprop gradients which have been shown to work well on training deep networks [7], and we keep the step size fixed at 10^{-4}. We experimented with different variants of softmax loss, and our proposed loss-layer with exponential growth Eq. (13) works best for anticipation (see Section VI for details).

V. Context for maneuver anticipation

In maneuver anticipation the goal is to anticipate the driver's future maneuver several seconds before it happens [19, 27]. The contextual information for anticipation is extracted from sensors installed in the vehicle. Previous work from Jain et al. [19] considers the context from a driver facing camera, a camera facing the road in front, a Global Positioning System (GPS), and an equipment for recording the vehicle's dynamics. The overall contextual information from the sensors is grouped into: (i) the context from inside the vehicle, which comes from the driver facing camera and is represented as temporal sequence of features $(\mathbf{x}_1, ..., \mathbf{x}_T)$; and (ii) the context from outside the vehicle, which comes from remaining sensors and is represented as $(\mathbf{z}_1, ..., \mathbf{z}_T)$.

We improve the pipeline from Jain et al. [19] with our deep learning architecture and new features for maneuver

[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.

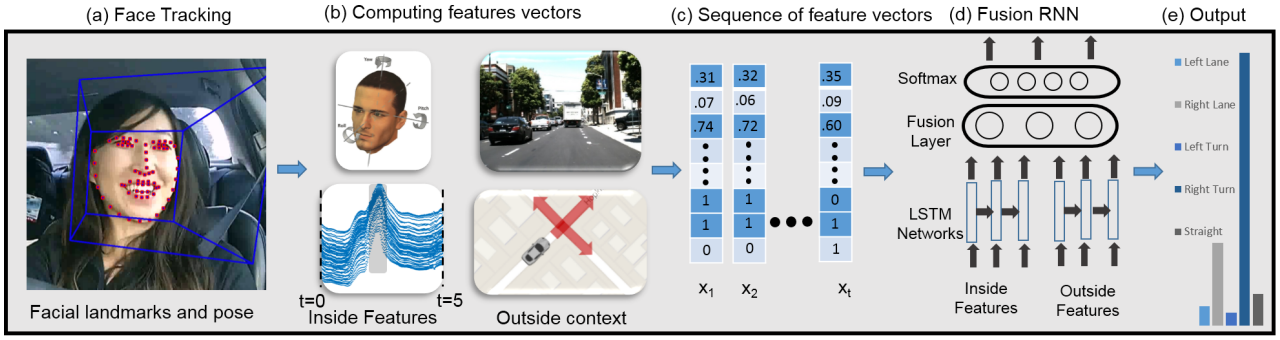


Fig. 5: **Maneuver anticipation pipeline.** Temporal context in maneuver anticipation comes from cameras facing the driver and the road, the GPS, and the vehicle's dynamics. (**a**, **b** and **c**) We improve upon the vision pipeline from Jain et al. [19] by tracking 68 landmark points on the driver's face and including the 3D head-pose features. (**d**) Using the Fusion-RNN we combine the sensory streams of features from inside the vehicle (driver facing camera), with the features from outside the vehicle (road camera, GPS, vehicle dynamics). (**e**) The model outputs the predicted probabilities of future maneuvers.

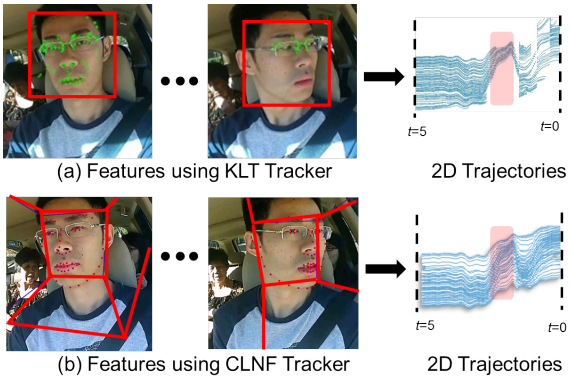


Fig. 6: **Improved features for maneuver anticipation.** We track facial landmark points using the CLNF tracker [3] which results in more consistent 2D trajectories as compared to the KLT tracker [32] used by Jain et al. [19]. Furthermore, the CLNF also gives an estimate of the driver's 3D head pose.

anticipation. Figure 5 shows our complete pipeline. In order to anticipate maneuvers, our RNN architecture (Figure 4) 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*$\}$. We now give an overview of the feature representation used by Jain et al. [19] and then describe our features which significantly improve the performance.

A. Features for maneuver anticipation

In the vision pipeline of Jain et al. [19], the driver facing camera detects discriminative points on the driver's face and tracks the detected points across frames using the KLT tracker [32]. The tracking generates 2D optical flow trajectories in the image plane. From these trajectories the horizontal and angular movements of the face are extracted, and these movements are binned into histogram features for every frame. These histogram features are aggregated over 20 frames (i.e. 0.8 seconds of driving) and constitute the feature vector $\mathbf{x}_t \in \mathbb{R}^9$. Further context for anticipation comes from the camera facing the road, the GPS, and the vehicle's dynamics. This is denoted by the feature vector $\mathbf{z}_t \in \mathbb{R}^6$, and includes the lane information, the road artifacts such as intersections, and the vehicle's speed. We refer the reader to Jain et al. [19] for more information on their features.

B. 3D head pose and facial landmark features

We now propose new features for maneuver anticipation which significantly improve upon the features from Jain et al. [19]. Instead of tracking discriminative points on the driver's face we use the Constrained Local Neural Field (CLNF) model [3] 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 6 CLNF offers us two distinct benefits over the features from Jain et al. [19] (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 $\mathbf{x}_t \in \mathbb{R}^{12}$.

In Section VI we present results with the features from Jain et al. [19], as well as the results with our improved features obtained from the CLNF model.

VI. Experiments

We evaluate our proposed architecture on the task of maneuver anticipation [1, 19, 27, 36]. This is an important problem because in the US alone 33,000 people die in road accidents every year – a majority of which are due to dangerous maneuvers. Advanced Driver Assistance Systems (ADAS) have made driving safer by alerting drivers whenever they commit a dangerous maneuver. Unfortunately, many accidents are unavoidable because by the time drivers are alerted it is already too late. Maneuver anticipation can avert many accidents by alerting drivers *before* they perform a dangerous maneuver [30].

We evaluate on the driving data set publicly released by Jain et al. [19]. The data set consists of 1180 miles of natural freeway and city driving collected from 10 drivers over a period of two months. It contains videos with both inside and outside views of the vehicle, the vehicle's speed, and GPS coordinates. The data set is annotated with 700 events consisting of 274 lane changes, 131 turns, and 295 randomly sampled instances of driving straight. Each lane

change and turn is also annotated with the start time of the maneuver, i.e. right before the wheel touches the lane marking or the vehicle yaws at the intersection, respectively. We augment the data set using the technique described in Section IV-D and generate 2250 events from the 700 original events. We train our deep learning architectures on the augmented data. We will make our code for data augmentation and maneuver anticipation publicly available at: `http://www.brain4cars.com`

We compare our deep RNN architecture with the following baseline algorithms:

1) *Chance:* Uniformly randomly anticipates a maneuver.
2) *Random-forest:* A discriminative classifier that learns an ensemble of 150 decision trees.
3) *SVM [27]:* Support Vector Machine classifier used by Morris et al. [27] for maneuver anticipation.
4) *IOHMM [19]:* Input-Output Hidden Markov Model [5] used by Jain et al. [19] for maneuver anticipation.
5) *AIO-HMM [19]:* This model extends IOHMM by including autoregressive connections in the output layer. AIO-HMM achieved state-of-the-art performance in Jain et al. [19].

In order to study the effect of our design choices we also compare the following modifications of our architecture:

6) *Simple-RNN (S-RNN):* In this architecture sensor streams are fused by simple concatenation and then passed through a single RNN with LSTM units.
7) *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 $-\log(y_t^k)$.
8) *Fusion-RNN-Exp-Loss (F-RNN-EL):* This architecture is similar to F-RNN-UL, except that the loss exponentially grows with time $-e^{-(T-t)}\log(y_t^k)$.

We use the RNN and LSTM implementations provided by Jain [18]. In our RNNs 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 [7]. The model training takes less than an hour on a K40 GPU.

A. Evaluation setup

We follow an evaluation setup similar to Jain et al. [19]. Algorithm 1 shows the inference steps for maneuver anticipation. At each time step t, features $\mathbf{x}_t$ and $\mathbf{z}_t$ are computed over the last 0.8 seconds of driving (20 frames). Using the temporal context $\{(\mathbf{x}_1, ..., \mathbf{x}_t),\ (\mathbf{z}_1, ..., \mathbf{z}_t)\}$, each anticipation algorithm computes the probability $\mathbf{y}_t$ for maneuvers in $\mathcal{M} = \{$*left lane change, right lane change, left turn, right turn, driving straight*$\}$. The prediction threshold is denoted by $p_{th} \in (0, 1]$ in Algorithm 1. The algorithm predicts *driving straight* if none of the softmax probabilities for the other maneuvers exceeds p_{th}.

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**
    $m^* = m_t^*$
    $t_{before} = T - t$
    break
  end if
end while
Return m^*, t_{before}

In order to evaluate an anticipation algorithm, we compute the following quantities for each maneuver $m \in \mathcal{M}$: (i) N_m: the total number of instances of maneuver m; (ii) TP_m: the number of instances of maneuver m correctly predicted by the algorithm; and (iii) P_m: the number of times the algorithm predicts m. Based on these quantities we evaluate the precision and recall of an anticipation algorithm as defined below:

$$Pr = \frac{1}{|M| - 1} \sum_{m \in \mathcal{M} \setminus \{\textit{driving straight}\}} \left(\frac{TP_m}{P_m} \right) \tag{14}$$

$$Re = \frac{1}{|M| - 1} \sum_{m \in \mathcal{M} \setminus \{\textit{driving straight}\}} \left(\frac{TP_m}{N_m} \right) \tag{15}$$

We should note that *driving straight* maneuver is not included in evaluating precision (14) and recall (15). This is because anticipation algorithms by default predict *driving straight* when they are not confident about other maneuvers. For each anticipation algorithm, we choose a prediction threshold p_{th} that maximizes their $F1$ score: $F1 = 2 * Pr * Re/(Pr + Re)$. In addition to precision and recall, we also measure the interval between the time an algorithm makes a prediction and the start of maneuver. We refer to this as the *time-to-maneuver* and denote it with t_{before} in Algorithm 1. We uniformly randomly partition the data set into five folds and report results using 5-fold cross-validation. We train on four folds and test on the fifth fold, and report the metrics averaged over the five folds.

B. Results

We evaluate anticipation algorithms on the maneuvers not seen during training with the following three prediction settings: (i) Lane change: algorithms only anticipate lane changes, i.e. $\mathcal{M} = \{$*left lane change, right lane change, driving straight*$\}$. This setting is relevant for freeway driving; (ii) Turns: algorithms only anticipate turns, i.e. $\mathcal{M} = \{$*left turn, right turn, driving straight*$\}$; and (iii) All maneuvers: algorithms anticipate all five maneuvers. Among these prediction settings, predicting all five maneuvers is the hardest.

Table I compares the performance of the baseline anticipation algorithms and the variants of our deep learning

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. [19].

	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	-
	SVM [27]	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
	IOHMM [19]	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
	AIO-HMM [19]	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
Our Methods	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
	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

model. All algorithms in Table I were evaluated on the features provided by Jain et al. [19], which ensures a fair comparison. We observe that variants of our architecture outperform the previous state-of-the-art a majority of the time. This improvement in performance is because RNNs with LSTM units are very expressive models, and unlike Jain et al. [19] they do not make any assumption about the generative nature of the problem.

The performance of several variants of our architecture, reported in Table I, justifies our design decisions to reach the final architecture as discussed here. S-RNN performs a very 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 at each time step to fuse the high-level representations. This form of sensory fusion is more principled since the sensor streams represent different data modalities. Fusing high-level representations instead of concatenating raw features gives a significant improvement in performance, as shown in Table I. When predicting all maneuvers, F-RNN-EL has a 6% higher precision and recall than S-RNN.

As shown in Table I, exponentially growing the loss improves 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 over-fitting 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.

We study the effect of our improved features in Table II. We replace the pipeline for extracting features from the driver's face [19] by a Constrained Local Neural Field (CLNF) model [3]. Our new vision pipeline tracks 68 facial landmark points and estimates the driver's 3D head pose as described in Section V-A. 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 6.

The confusion matrix in Figure 7 shows the precision for each maneuver. F-RNN-EL gives a higher precision than AIO-HMM on every maneuver when both algorithms

TABLE II: **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 [3] to track 68 facial landmark points and estimate 3D head-pose.

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

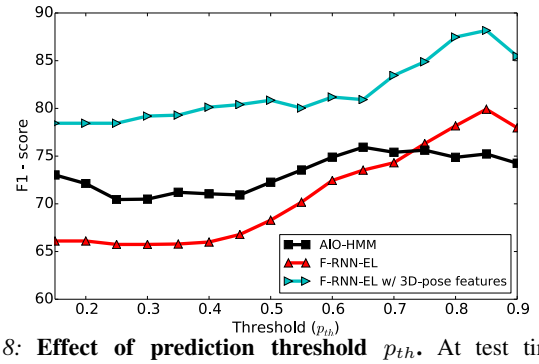


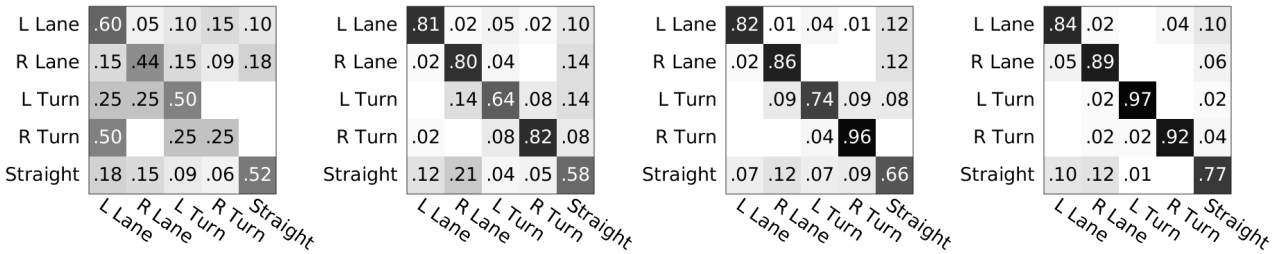
Fig. 8: **Effect of prediction threshold** p_{th}**.** At test time an algorithm makes a prediction only when it is at least p_{th} confident in its prediction. This plot shows how F1-score vary with change in prediction threshold.

are trained on same features (Fig. 7c). Our new vision pipeline further improves the precision of F-RNN-EL on all maneuvers (Fig. 7d). Additionally, both F-RNN and AIO-HMM perform significantly better than previous work on maneuver anticipation by Morris et al. [27] (Fig. 7a).

In Figure 8 we study how F1-score varies as we change the prediction threshold p_{th}. 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 and II, we used the threshold values which gave the highest F1-score.

VII. Conclusion

In this work we addressed the problem of anticipating maneuvers several seconds before they happen. 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



(a) SVM [27] *(b)* AIO-HMM [19] *(c)* F-RNN-EL *(d)* F-RNN-EL w/ 3D-pose

Fig. 7: **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.

layer for anticipation which prevents over-fitting.

Our deep learning architecture outperformed the previous state-of-the-art on 1180 miles of natural driving data set. It improved the precision from 78% to 84.5% and recall from 71.1% to 77.1%. We further showed that improving head tracking and including the driver's 3D head pose as a feature gives a significant boost in performance by increasing the precision to 90.5% and recall to 87.4%. We believe that our approach is widely applicable to many activity anticipation problems. As more anticipation data sets become publicly available, we expect to see a similar improvement in performance with our architecture.

Acknowledgement. This work was supported by NRI, ONR, and NSF Career Awards and by Microsoft Faculty Fellowship to Saxena.

REFERENCES

[1] Bosch urban. `http://bit.ly/1feM3JM`. Accessed: 2015-04-23.
[2] A. Andreas, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.
[3] T. Baltrusaitis, P. Robinson, and L-P. Morency. Constrained local neural fields for robust facial landmark detection in the wild. In *ICCV Workshop*, 2013.
[4] Y. Bengio and O. Delalleau. On the expressive power of deep architectures. In *Algorithmic Learning Theory*, pages 18–36, 2011.
[5] Y. Bengio and O. Frasconi. An input output hmm architecture. *NIPS*, 1995.
[6] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun. Learning motion patterns of people for compliant robot motion. *IJRR*, 2005.
[7] 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.
[8] 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 *CVPR*, 2015.
[9] A. Doshi, B. Morris, and M. M. Trivedi. On-road prediction of driver's intent with multimodal sensory cues. *IEEE Pervasive Computing*, 2011.
[10] A. Dragan and S. Srinivasa. Formalizing assistive teleoperation. In *RSS*, 2012.
[11] K. Driggs-Campbell, V. Shia, and R. Bajcsy. Improved driver modeling for human-in-the-loop vehicular control. In *ICRA*, 2015.
[12] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015.
[13] L. Fletcher, N. Apostoloff, L. Petersson, and A. Zelinsky. Vision in and out of vehicles. *IEEE IS*, 18(3), 2003.
[14] L. Fletcher, G. Loy, N. Barnes, and A. Zelinsky. Correlating driver gaze with the road scene for driver assistance systems. *RAS*, 52(1), 2005.
[15] 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.
[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8), 1997.
[17] 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.
[18] A. Jain. Neuralmodels. `https://github.com/asheshjain399/NeuralModels`, 2015.
[19] 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.
[20] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *Proc. ECCV*. 2012.
[21] H. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *RSS*, 2013.
[22] H. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from rgb-d videos. *IJRR*, 32(8), 2013.
[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
[24] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *RSS*, 2012.
[25] J. Mainprice and D. Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *IROS*, 2013.
[26] D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *RSS*, 2014.
[27] B. Morris, A. Doshi, and M. Trivedi. Lane change intent prediction for driver assistance: On-road design and evaluation. In *IEEE IVS*, 2011.
[28] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *ICML*, 2011.
[29] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *arXiv:1211.5063*, 2012.
[30] 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.
[31] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*, 2011.
[32] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994.
[33] V. Shia, Y. Gao, R. Vasudevan, K. D. Campbell, T. Lin, F. Borrelli, and R. Bajcsy. Semiautonomous vehicular control using driver modeling. *IEEE Trans. on ITS*, 15(6), 2014.
[34] J. Sung, S. H. Jin, and A. Saxena. Robobarista: Object part-based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *ISRR*, 2015.
[35] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.
[36] 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.
[37] M. Trivedi, T. Gandhi, and J. McCall. Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety. *IEEE Trans. on ITS*, 8(1), 2007.
[38] R. Vasudevan, V. Shia, Y. Gao, R. Cervera-Navarro, R. Bajcsy, and F. Borrelli. Safe semi-autonomous control with enhanced driver modeling. In *ACC*, 2012.
[39] H. Wang and C. Schmid. Action recognition with improved trajectories. In *CVPR*, 2013.
[40] 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. *IJRR*, 2013.
[41] 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 *IROS*, 2009.