Real-Time Workload Classification during Driving using HyperNetworks

Ruohan Wang and Pierluigi V. Amadori and Yiannis Demiris.

***Abstract*— Classifying human cognitive states from behavioral and physiological signals is a challenging problem with important applications in robotics. The problem is challenging due to the data variability among individual users, and sensor artefacts. In this work, we propose an end-to-end framework for real-time cognitive workload classification with mixture Hyper Long Short Term Memory Networks (m-HyperLSTM), a novel variant of HyperNetworks. Evaluating the proposed approach on an eye-gaze pattern dataset collected from simulated driving scenarios of different cognitive demands, we show that the proposed framework outperforms previous baseline methods and achieves 83.9% precision and 87.8% recall during test. We also demonstrate the merit of our proposed architecture by showing improved performance over other LSTM-based methods.**

I. Introduction

Classifying human cognitive states is an important problem with many applications in robotics. In human-robot interaction, the ability to predict human intentions enables robots to perform socially compliant navigation and collaborate with humans [1], [2], [3], [4]. For intelligent vehicles, intention or distraction prediction allows the systems to alert users before potentially dangerous maneuvers [5], [6], [7]. Further, casting driving assistance as a problem of human-in-the-loop control, users' cognitive states provide input for deriving control policies to manage user interfaces and take over control if necessary [8], [9], [10], [11].

Previous studies show that physiological and behavioral signals correlate with cognitive states. For instance, [6] used head movements to predict intention in driving, while [8] showed real-time quantitative correlation between stress and physiological signals including Electrocardiogram (ECG), skin conductance, and respiration in different individuals. Common challenges demonstrated in those studies are data variability among individuals and sensor artefacts. Hand-crafted features are commonly employed to improve data quality and summarize the data into fixed-size feature vectors suitable for classification algorithms such as logistic regression and Support Vector Machine (SVM). However, it is desirable for a model to 1) automatically learn feature representations from data to reduce manual feature engineering, and 2) be sufficiently flexible to tackle data variability.

Towards the goals stated above, we propose a framework for real-time cognitive workload classification using mixture Hyper Long Short Term Memory Networks (m-HyperLSTM), a novel variant of HyperNetworks [12] based on LSTM [13]. HyperLSTM is a class of HyperNetworks wherein the model parameters adapt based on the input. Our choice of the model is motivated by the hypothesis that the adaptive nature of HyperNetworks can be exploited to tackle data variability while LSTMs are known for their ability to capture long range dependency and learning useful feature representations from data [13]. We formulate the classification task as learning a sequence-to-sequence mapping.

Authors are with the Personal Robotics Lab, Department of Electrical and Electronic Engineering, Imperial College London, UK {r.wang16, p.amadori, y.demiris}@imperial.ac.uk

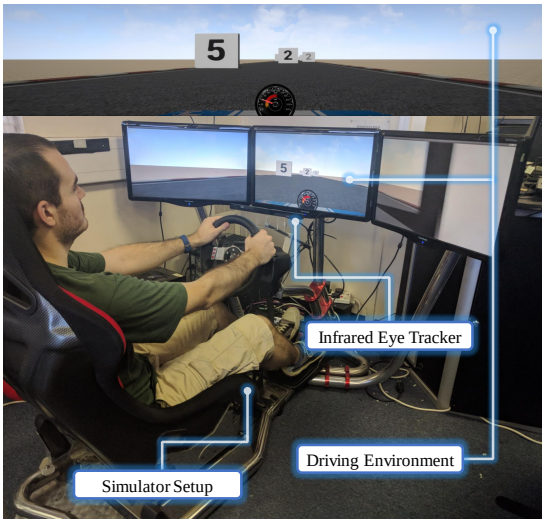

Fig. 1. Top: Simulated driving environment. Numbered obstacles are placed in three lanes. Participants drive along the road to avoid obstacles, and perform mental tasks, while their performance data, and their instantaneous gaze locations are recorded. Bottom: Simulator physical setup.

We collect an eye-gaze location dataset from simulated driving whereby 20 participants complete tasks of different cognitive demands. We then evaluate our proposed approach on the dataset for binary classification of cognitive workload levels (low and high). The classification is challenging as the dataset is both noisy and exhibits varying visual scanning patterns across individuals, as shown in Fig. 2. Similar to [5], we choose eye gaze as input because eye tracking is less intrusive compared to skin conductance or ECG tracking, and readily available through consumer products (e.g. cameras in natural environments [14] or smart phones [15], and has wider applicability beyond driving. We stress that the proposed framework is generic to different sensor modalities and multi-class classification, and we intend to explore sensor fusion and more fine-grained cognitive states classification in future works, including the usage of skin conductance, ECG data and extending to multi-class classifications with different workload levels.

We report experiment results comparing m-HyperLSTM with different baseline models, including state-of-the-art

sequence models such as HyperLSTM [12], and LSTM [6], [7], as well as classical models such as SVM [16], [5], and logistic regression [17], [16]. The proposed approach outperforms the baselines and achieves a 83.9% precision and 87.8% recall on the test sets. Improved performance over HyperLSTM and LSTM validates the efficacy of the proposed architecture in tackling the variability in the dataset. Key contributions of the paper are:

- We introduce m-HyperLSTM for real-time cognitive workload estimation. The architecture jointly learns feature representations and adapts itself based on input. Our contribution is a novel weight generation scheme inspired by the idea of mixture models, aimed at tackling data variability and improving generalization performance.
- We evaluate the performance of the proposed model against baseline approaches using multiple evaluation metrics, including F1-score, precision, and recall.
- We validate the proposed architecture ability to handle data variability in simulated driving tasks, in comparison with LSTM variants commonly used for sequence modeling.

II. Related Work

Our work is related to previous works on cognitive states classification, and on Recurrent Neural Networks (RNNs) for sequence prediction.

Using physiological signals for cognitive workload classification presents multiple challenges. Sophisticated engineering is often required to improve data quality and extract useful features from raw senor signals (e.g. [8], [5], [16]. On the other hand, the extent of statistical correlations between cognitive workload and physiological signals can vary significantly among experiment participants [8]. One possible solution to data variability is personalized models, as seen in [18], [17]. However, this approach may become impractical as data collection and model training are required for every new user. Similar to [6], [7], our proposed model aims to learn feature representations directly from data. In addition, the adaptive nature of the proposed model directly tackle data variability. Our experiments demonstrate that adaptive models outperform the static ones, hence suggesting a viable generic technique for tackling data variability found in physiological signals.

LSTM Networks [13] and HyperNetworks [12] are the core components of the proposed model. LSTM Networks were designed to capture long-range dependency within input sequences, and have been shown effective across various sequence modeling tasks, including natural language processing [19] and robotic perception [20]. HyperNetworks refer to the general approach of generating the weights of a network by another network. Specifically, HyperLSTM extends LSTM by using an auxiliary LSTM to dynamically generate the weights for the main LSTM at each time step, thus enabling the main LSTM to adapt itself based on the input. HyperLSTM has been shown to outperform LSTM in language modeling, handwriting generation and neural machine translation [12]. We introduce m-HyperLSTM inspired by the idea of mixture models, to further exploit the dynamic property of HyperNetworks for cognitive workload classification.

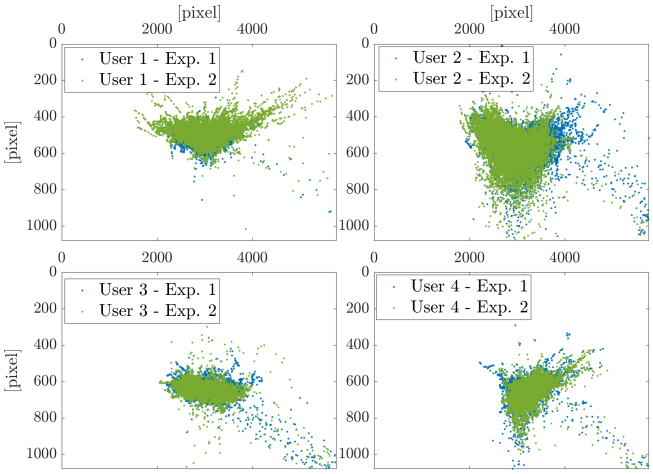

Fig. 2. Scatter plot of instantaneous gaze locations of four participants during low (blue) and high (green) workload situations. The plots highlight that the dataset is challenging as eye gaze patterns differ among individuals. Best viewed in color.

III. Method

We cast the task of cognitive workload classification as supervised learning. Given a dataset $\{(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T)_j, \boldsymbol{y}_j\}_{j=1}^N$ where $\boldsymbol{x}_t$ denotes physiological signals at time t, y the target workload level for the sequence, and T the sequence length, we aim to learn a model θ that maximizes the probability $p(\boldsymbol{y}|\boldsymbol{x}_{1,..,T})$. Instead of relying a single label, we follow [6] to train our model using the following loss function in a sequence-to-sequence manner

$$loss = \sum_{j=1}^{N}\sum_{t=1}^{T} -e^{(t-T)}\log p(y_t^k|\boldsymbol{x}_{<t}) \quad (1)$$

where $\boldsymbol{x}_{<t}$ denotes the subsequence $\boldsymbol{x}_{1,..,t}$, and y_t^k the probability of ground truth label computed by the model at time t. In addition to encouraging the network to fix early mistakes and reducing the possibility of over fitting when the current context is insufficient for classification [6], the loss function also reduces model variance (i.e., changing predicted label between time steps). We implement our model as m-HyperLSTM described in Section III-B.

A. Long Short-Term Memory Networks

LSTM is a RNN that implements a memory cell to maintain contextual information over time, and thus captures long-range dependencies in data sequences [13]. Given an input sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T$, LSTM maps the input sequence to a sequence

of hidden states $\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T$ via the following updates:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}^i \boldsymbol{h}_{t-1} + \boldsymbol{I}^i \boldsymbol{x}_t + \boldsymbol{b}_i) \quad (2)$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}^f \boldsymbol{h}_{t-1} + \boldsymbol{I}^f \boldsymbol{x}_t + \boldsymbol{b}_f) \quad (3)$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^o \boldsymbol{h}_{t-1} + \boldsymbol{I}^o \boldsymbol{x}_t + \boldsymbol{b}_o) \quad (4)$$

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

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \quad (6)$$

where $\odot$ denotes the element-wise product, σ and $\tanh$ the element-wise sigmoid function and hyperbolic tangent function respectively. $\boldsymbol{W}^*$, $\boldsymbol{I}^*$, and $\boldsymbol{b}_*$ are parameters to be learned, with $*$ represents one of $\{i, c, f, o\}$ gates. Eq. 5 shows that the memory $\boldsymbol{c}_t$ of LSTM selectively carries information from the previous time step by controlling what to remember via the forget gate $\boldsymbol{f}_t$. The LSTM defined above is similar to the architecture of [21] but without peep-hole connections. For simplicity, we use the following shorthand for LSTM updates:

$$(\boldsymbol{h}_t, \boldsymbol{c}_t) = \text{LSTM}(\boldsymbol{h}_{t-1}, \boldsymbol{c}_{t-1}, \boldsymbol{x}_t).$$

B. HyperLSTM

HyperNetworks is a family of network architectures that generates the weights of a network via another network, and has achieved state-of-the-art performance in various sequence modeling tasks [12]. In particular, HyperLSTM extends LSTM by using an auxiliary LSTM (LSTM_{aux}) to dynamically generate the weights of a main LSTM (LSTM_{main}), shown in Fig. 3a. Following the update for LSTM_{aux} in [12], we have

$$\hat{\boldsymbol{x}}_t = \begin{pmatrix} \boldsymbol{x}_t \\ \boldsymbol{h}_{t-1} \end{pmatrix}$$

$$(\hat{\boldsymbol{h}}_t, \hat{\boldsymbol{c}}_t) = \text{LSTM}_{aux}(\hat{\boldsymbol{h}}_{t-1}, \hat{\boldsymbol{c}}_{t-1}, \hat{\boldsymbol{x}}_t),$$

where the input $\hat{\boldsymbol{x}}_t$ to LSTM_{aux} is the concatenation of the current input $\boldsymbol{x}_t$ and the hidden state $\boldsymbol{h}_{t-1}$ from LSTM_{main}. HyperLSTM then parametrizes the weights of LSTM_{main} at time t as a function of $\hat{\boldsymbol{h}}_t$. For $\boldsymbol{W}_t^*$, it is defined as

$$\boldsymbol{z}_h^* = \boldsymbol{W}_{hz}^* \hat{\boldsymbol{h}}_t + \boldsymbol{b}_h^* \quad (7)$$

$$\boldsymbol{d}_h^* = \boldsymbol{W}_{hd}^* \boldsymbol{z}_h^* \quad (8)$$

$$\boldsymbol{W}_t^* = \begin{pmatrix} (\boldsymbol{d}_h^*)_1 \boldsymbol{W}_{(1)}^* \\ (\boldsymbol{d}_h^*)_2 \boldsymbol{W}_{(2)}^* \\ ... \\ (\boldsymbol{d}_h^*)_{N_h} \boldsymbol{W}_{(N_h)}^* \end{pmatrix}, \quad (9)$$

where $\boldsymbol{W}_{(j)}^*$ denotes the j-th row of $\boldsymbol{W}^*$, and $\boldsymbol{W}_{hz}^*$, $\boldsymbol{b}_h^*$ and $\boldsymbol{W}_{hd}^*$ parameters to be learned. Both $\boldsymbol{I}_t^*$ and $\boldsymbol{b}_t^*$ follow the same update rule, omitted here for brevity. For further details on HyperLSTM, we refer the readers to [12].

IV. m-HyperLSTM

Many other mappings from current contexts to network weights are possible. We present a novel scheme for weights generation, inspired by the idea of mixture models, shown in Fig. 3b. The scheme mixes N_z LSTMs before the nonlinearity with their activation strengths (from 0 to 1) determined by the current context. Analytically, we formulate the update rule as

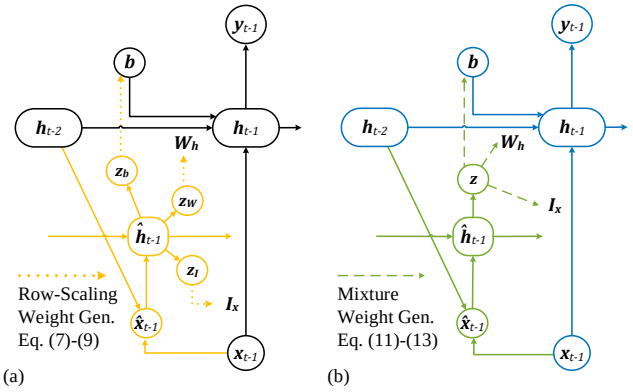

Fig. 3. Comparison between the original HyperLSTM (left) and the proposed m-HyperLSTM architecture (right). Key differences between the two are the weight generation schemes and the associated parameter sharing. Here, x_t represents input at time t, h_t denotes learned features/hidden states, and y_t identifies prediction output.

$$\boldsymbol{z} = \sigma(\boldsymbol{W}^z \hat{\boldsymbol{h}}_t + \boldsymbol{b}^z) \quad (10)$$

$$\boldsymbol{W}_t^* = < \boldsymbol{W}_z^*, \boldsymbol{z} > \quad (11)$$

$$\boldsymbol{I}_t^* = < \boldsymbol{I}_z^*, \boldsymbol{z} > \quad (12)$$

$$\boldsymbol{b}_t^* = < \boldsymbol{b}_z^*, \boldsymbol{z} >, \quad (13)$$

where $\boldsymbol{W}_z^* \in \mathbb{R}^{N_h \times N_h \times N_z}$, $\boldsymbol{I}_z^* \in \mathbb{R}^{N_h \times N_x \times N_z}$, $\boldsymbol{b}_z^* \in \mathbb{R}^{N_h \times N_z}$. N_h, N_x and N_z denote the dimensions of $\boldsymbol{h}_t$, $\boldsymbol{x}_t$ and $\boldsymbol{z}$ respectively. $< \circ, \circ >$ denotes the dot product.

The key differences between the proposed weight generation scheme and the original scheme are 1) parameter allocation and 2) increased regularization. Given a fixed parameter budget, our model trade-offs the size of hidden states for more expressive weight generation, while the original model does the opposite. In addition, the proposed scheme is more flexible as it may learn up to N_z components by turning on and off each element of $\boldsymbol{z}$ independently, which helps to prevent overfitting. If only a single element of $\boldsymbol{z}$ is turned on at all time steps, our model reduces to a standard LSTM Network. Further, our scheme shares $\boldsymbol{z}$ for all weights generation to improve regularization. We found that m-HyperLSTM outperforms the original in the experiments, suggesting that expressive weight generation and additional regularization contribute to the improved performance.

A. Network Architecture and Training Procedure

Given an input sequence $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$, we use the proposed architecture to map the input sequence to a sequence of hidden states $\{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T\}$. We then project the hidden states with a fully-connected layer with Rectified Linear Unit (ReLU) nonlinearity, followed by a softmax layer to predict the probability for each possible label.

$$y_t = \text{softmax}(\boldsymbol{W}_2 \text{ReLU}(\boldsymbol{W}_1 \boldsymbol{h}_t + \boldsymbol{b}_1) + \boldsymbol{b}_2).$$

To stabilize hidden state dynamics, we apply layer normalization as suggested in [12]. To improve generalization, we employ L2 regularization and a label smoothing

technique [22]. The label smoothing technique penalizes overconfident predictions by assigning $1-\epsilon+\frac{\epsilon}{K}$ probability to the correct label, and $\frac{\epsilon}{K}$ to all other labels, where ϵ is a tunable hyper parameter, and K the number of possible labels. Label smoothing replaces $log\ p(y_t^k|\boldsymbol{x}_{<t})$ in Eq. 1 with cross-entropy $H(p, y_t)$ where p is the smoothed ground truth distribution. Label smoothing naturally fits with cognitive workload classification as there is inherent uncertainty in the ground truth labels, given that cognitive workload is not directly observable [23]. All models are trained with Adam [24] with a fixed learning rate of 0.0001. We set $\epsilon = 0.2$ as recommended in [22].

V. Experiments

We evaluate the proposed approach on the real-time classification of cognitive workload using eye gaze patterns. We detail in the following sections the experimental procedures for collecting the gaze location dataset of the participants under different cognitive workloads. We evaluate our architecture on the collected dataset and compare it to baseline methods, including LSTM [6], [7] HyperLSTM [12], SVM [16], [5] and logistic regression [16], [17]. We aim to address the following questions:

- Is m-HyperLSTM capable of learning useful feature representations from eye gaze patterns and classifying cognitive workload across individuals in driving scenarios?
- How does m-HyperLSTM compare to the state-of-the-art sequence models as well as classical methods in terms of classification performance?

A. Participants

Twenty participants (12 males, 8 females, mean age 24.3, standard deviation 3.2) with normal or corrected to normal vision consented to participate in the experiment. After a brief introduction to the experiment and calibration procedure, participants were given a trial period to familiarize themselves with the simulator environment before the actual experiment.

B. Setup

A realistic driving simulation was set up for the experiment (Fig. 1). The environment comprised of monitors, a physical simulator, and a remote eye tracker, mounted above the steering wheel. We developed a customized simulated driving environment based on the Unreal Engine (Fig. 1).

C. Experimental Procedure

Since cognitive workload is not directly observable [23], we follow previous approaches [5], [8], [16], [11] to modulate the cognitive workload experienced by the participants by varying task difficulties using a validated experimental approach for workload generation. The experiment includes two scenarios with different workload levels (low and high), and therefore binary ground truth labels. Though only a coarse classification of cognitive workload is considered in this work, the information is nevertheless an important input for assistive robotics as demonstrated in [9], [10], [11]. We also intend to explore more fine-grained classification in future works.

For both low and high workload scenarios, the primary objective is to drive along a straight road and avoid stationary rectangular obstacles. The obstacles are numbered 0 through 9 and placed at a regular interval (Fig. 1). Participants are asked to maintain their speed between 120 and 130 km/h to ensure a consistent workload level throughout the scenarios. Participants were asked to repeat a scenario if their driving speed deviated from the specified range by 10km/h. The road is divided into three lanes and obstacles are randomly placed at one of three lanes. Obstacles are designed to block an entire lane, so that participants must steer to avoid them. Furthermore, to ensure that a lane is not free of obstacles for extended periods of time, thus reducing primary task difficulty, we employ a custom-defined discrete distribution for obstacle placement. Consider c_i as the distance between the current obstacle location and the previous obstacle location in lane i, we define the obstacle placement probability distribution in lane i as

$$p(i) = \frac{e^{c_i/\text{IntervalSize}}}{\sum_i e^{c_i/\text{IntervalSize}}}.$$

where IntervalSize represents the distance between two adjacent obstacles. This ensures that a lane would almost certainly be blocked if the lane has not been chosen for the previous few obstacles.

We employ a visual "n-back" task [25] as the secondary objective for controlling the workload level of the participants. The task induces different levels of cognitive workload by varying the amount of information that participants need to memorize in their working memory. This approach has been validated in previous studies to provide a consistent level of cognitive workload [25], [26], [27]. In our experiment, low workload scenario is associated with a 0-back task (i.e., no memorization required), wherein participants are simply required to determine the parity of the number on the nearest obstacle ahead, and press a corresponding button located on the steering wheel. In the high workload scenario, a 1-back task is employed, so that participants need to recall the parity of the number on the previous obstacle and, as they drive past a new obstacle, press the corresponding button. It is important to stress that the only difference between the two scenarios is the additional cognitive workload stemming from the memorization of numbers. This is pivotal for mitigating the risk of the model classifying other variables, such as additional visual stimuli rather than the cognitive workload.

D. Data Collection and Pre-processing

We collected instantaneous gaze locations in the reference plane of the center monitor at 60 Hz. The data is recorded in the format of {*timestamp, x-coordinate, y-coordinate*}. For each sample, we augment the data with the following attributes: distance from the previous sample (horizontal distance, vertical distance and overall) and the instantaneous speed from previous sample (horizontal speed, vertical speed and overall). In total, each time step contains 8 attributes {*x-coordinate, y-coordinate, x-distance, y-distance, distance, x-speed, y-speed, speed*}.

For logistic regression and SVM, we reduce a temporal sequence of attributes into a fixed-size feature vector by capturing the central tendencies, variability, and extremes of each attribute. These features include mean, standard deviation, median, 25th and 75th percentiles, maximum, minimum and range over a window size of t_w seconds for each attribute, resulting in a total of $8 \times 8 = 64$ features. The window size determines the amount of context available for classification and directly impacts the real-timeliness of the method. For LSTM-based models, the input sequence consists of t_w steps for the same window size, with the input for each time step being the features defined above across a 1-second window. All features are normalized to the interval $[0, 1]$ and we uniformly sample the input sequences using a sliding window with 90% overlap to generate training samples for all the models.

E. Evaluation Setup

We follow an evaluation framework similar to [28], [6]. The evaluation metrics include precision, recall, and F1-score. We train on 80% of data, setting aside 10% each for validation and testing using uniformly random splits. We use the validation set to select the model with lowest validation loss within 50 epochs of training, and the decision threshold that maximizes the F1-score. For each model, we report the mean and the standard deviation for each metrics over five randomly sampled and non-overlapping test sets.

For SVM, we use a simple grid search to determine the best hyper parameters ($C = 5, \gamma = 0.01$). For logistic regression, L2 regularization is used. For all LSTM-based methods, the training procedures and the usage of regularization techniques are identical, as described in IV-A. We choose the network sizes for all LSTM-based models such that each model has approximately the same number of parameters and thus similar model capacity. Specifically, we consider a LSTM with a hidden state size of 100. For the original HyperLSTM, we consider a LSTM$_{main}$ with hidden state size of 75, a LSTM$_{aux}$ with hidden state size of 16, and $N_z = 4$. For the proposed model, we use a hidden state size of 32 for LSTM$_{main}$ and all other settings are identical to those of the original HyperLSTM.

VI. Results

We present the classification performance of all evaluated models in Table I, for $t_w = 5s, 10s, 20s$ respectively. The results show that m-HyperLSTM achieves the highest F1-score across all window sizes. At 10s window, m-HyperLSTM also outperforms all baselines for precision and recall. Similar to the previous studies [5], [17], [6], our results verify that that longer contextual information yield better classification accuracies across all evaluated models. The results also suggest that a trade-off between the timeliness and performance of the classification. For real-time applications, the results suggest that our method using a 10s window may offer the best trade-off.

The results suggest that LSTM-based methods are a class of flexible and expressive models capable of learning useful feature representations from gaze locations sequence and outperform the baselines that utilize handcrafted features. While it may be possible to match the performance of LSTM-based methods with more sophisticated feature engineering, the ability to automatically extract features from data is an important advantage of LSTM-based methods. The improved performance from m-HyperLSTM over the original HyperLSTM validates the efficacy of the proposed weight generation scheme, which uses more expressive weight generation and additional regularization.

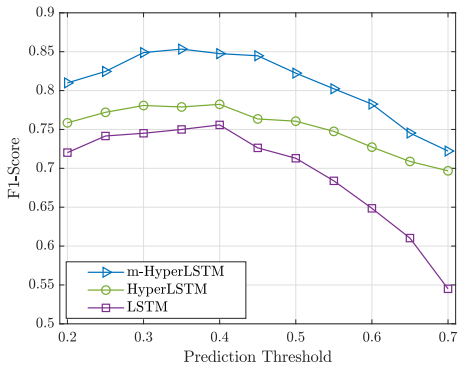

Fig. 4. F1-score against decision threshold for the proposed method, HyperLSTM and LSTM at $t_w = 10$. The proposed method outperforms the other two and achieves a fairly consistent F1-score across increasing decision thresholds.

To further evaluate the performance of our model, we show in Fig. 4 the impact of decision threshold on F1-score for $t_w = 10s$. Small variations in F1-score across multiple decision thresholds indicate that the model is capable of trading off between precision and recall depending on the application requirements without hurting the overall evaluation metric [6]. We observe that m-HyperLSTM outperforms both HyperLSTM and LSTM for all the spectrum of decision thresholds, while achieving fairly consistent F1-scores across increasing decision thresholds.

A. Real-Time Inference

m-HyperLSTM is readily usable for real-time classification. During real-time inference, the gaze locations are aggregated into feature vectors at each second and a context of the latest t_w seconds are used as input for the network to predict the current workload level. In our supplementary video, we present the real-time classification of workload for the same participants.

Real-Time classification of workload has many potential applications. As a starting point, the predicted workload could be used to manage non-critical user interaction within intelligent vehicles, such as lowering music volumes or diverting calls to voice mails to reduce workload of users [8]. As model performance continue to improve, the predicted workload may be applied to more critical tasks such as deriving the control policy for human-in-loop control, as demonstrated in [9], [10].

TABLE I

CLASSIFICATION PERFORMANCE ON COGNITIVE WORKLOAD USING GAZE LOCATION SEQUENCE

	5s			10s			20s		
	F1	Pr (%)	Re (%)	F1	Pr (%)	Re (%)	F1	Pr (%)	Re (%)
SVM	0.52 ± 0.01	$\mathbf{68.1} \pm 0.5$	42.1 ± 1.0	0.60 ± 0.01	69.5 ± 0.66	52.7 ± 0.7	0.69 ± 0.01	69.2 ± 1.2	67.8 ± 1.2
Log Reg	0.67 ± 0.005	51.9 ± 0.6	$\mathbf{93.6} \pm 2.1$	0.69 ± 0.01	55.8 ± 1.9	88.9 ± 4.0	0.71 ± 0.01	60.3 ± 1.4	$\mathbf{87.7} \pm 1.6$
LSTM	0.67 ± 0.01	54.6 ± 1.4	86.9 ± 3.3	0.76 ± 0.01	70.6 ± 2.1	82.4 ± 3.6	0.79 ± 0.03	74.0 ± 2.9	84.6 ± 4.4
HyperLSTM	0.70 ± 0.02	58.5 ± 3.2	88.9 ± 3.2	0.79 ± 0.04	73.8 ± 5.9	87.8 ± 1.6	0.77 ± 0.03	76.4 ± 7.9	78.2 ± 2.2
m-HyperLSTM	$\mathbf{0.71} \pm 0.01$	62.4 ± 2.9	83.8 ± 5.8	$\mathbf{0.86} \pm 0.01$	$\mathbf{83.9} \pm 5.1$	$\mathbf{87.9} \pm 3.5$	$\mathbf{0.88} \pm 0.03$	$\mathbf{90.1} \pm 1.9$	86.3 ± 6.6

VII. CONCLUSIONS

The ability to predict human cognitive states is an important problem with many applications. In this work, we addressed the problem of cognitive workload classification using a sequence of gaze locations with only consumer-grade hardware. The proposed framework is task-agnostic and generic enough for other temporal data such as EEG or ECG readings. The proposed method is able to tackle data variability commonly found in physiological signals and outperforms state-of-the-art sequence models. For future work, an interesting direction would be multi-sensory fusion, which may further improve model performance and reliability.

ACKNOWLEDGMENT

The authors would like to thank Antoine Cully for useful discussions on this work, and all the experiment participants.

REFERENCES

[1] Y. Demiris, "Prediction of intention in robotics and multiagent systems," *Cognitive Processing*, vol. 8, no. 3, pp. 151–158, 2007.

[2] M. K. H. Kretzschmar and C. S. W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," *Robotics: Science and Systems VIII*, p. 193, 2013.

[3] H. S. Koppula and A. Saxena, "Anticipating human activities using object affordances for reactive robotic response," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 14–29, 2016.

[4] J. Mainprice and D. Berenson, "Human-robot collaborative manipulation planning using early prediction of human motion," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 299–306.

[5] Y. Liang, M. L. Reyes, and J. D. Lee, "Real-time detection of driver cognitive distraction using support vector machines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 340–350, June 2007.

[6] A. Jain, A. Singh, H. S. Koppula, S. Soh, and A. Saxena, "Recurrent neural networks for driver activity anticipation via sensory-fusion architecture," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3118–3125.

[7] M. Wollmer, C. Blaschke, T. Schindl, B. Schuller, B. Farber, S. Mayer, and B. Trefflich, "Online driver distraction detection using long short-term memory," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 574–582, 2011.

[8] J. A. Healey and R. W. Picard, "Detecting stress during real-world driving tasks using physiological sensors," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 156–166, June 2005.

[9] C. P. Lam, A. Y. Yang, K. Driggs-Campbell, R. Bajcsy, and S. S. Sastry, "Improving human-in-the-loop decision making in multi-mode driver assistance systems using hidden mode stochastic hybrid systems," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 5776–5783.

[10] K. Driggs-Campbell, V. Shia, and R. Bajcsy, "Improved driver modeling for human-in-the-loop vehicular control," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1654–1661.

[11] T. Carlson and Y. Demiris, "Collaborative control for a robotic wheelchair: evaluation of performance, attention, and workload," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 876–888, 2012.

[12] D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks," *International Conference on Learning Representations*, 2017.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] T. Fischer, H. J. Chang, and Y. Demiris, "Rt-gene: Real-time eye gaze estimation in natural environments," in *Proceedings of the European Conference on Computer Vision*, 2018.

[15] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] E. T. Solovey, M. Zec, E. A. Garcia Perez, B. Reimer, and B. Mehler, "Classifying driver workload using physiological and driving performance data: Two field studies," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '14. New York, NY, USA: ACM, 2014, pp. 4057–4066.

[17] T. Georgiou and Y. Demiris, "Adaptive user modelling in car racing games using behavioural and physiological data," *User Modeling and User-Adapted Interaction*, vol. 27, no. 2, pp. 267–311, Jun 2017.

[18] E. Ferreira, D. Ferreira, S. Kim, P. Siirtola, J. Rning, J. F. Forlizzi, and A. K. Dey, "Assessing real-time cognitive load based on psycho-physiological measures for younger and older adults," in *2014 IEEE Symposium on Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB)*, Dec 2014, pp. 39–48.

[19] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[20] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Advances in Neural Information Processing Systems*, 2016, pp. 64–72.

[21] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[23] D. Gopher and E. Donchin, "Workload-an examination of the concept," *Handbook of Perception and Human Performance, Vol II, Cognitive Processes and Performance. New York: Wiley & Sons*, 1986.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference for Learning Representations*, 2015.

[25] B. Mehler, B. Reimer, and J. A. Dusek, "Mit agelab delayed digit recall task (n-back)," *Cambridge, MA: Massachusetts Institute of Technology*, 2011.

[26] B. Mehler, B. Reimer, and J. F. Coughlin, "Sensitivity of physiological measures for detecting systematic variations in cognitive demand from a working memory task: an on-road study across three age groups," *Human factors*, vol. 54, no. 3, pp. 396–412, 2012.

[27] B. Reimer, B. Mehler, Y. Wang, and J. F. Coughlin, "A field study on the impact of variations in short-term memory demands on drivers visual attention and driving performance across three age groups," *Human Factors*, vol. 54, no. 3, pp. 454–468, 2012.

[28] Z. C. Lipton, D. C. Kale, and R. Wetzel, "Modeling missing data in clinical time series with rnns," *Machine Learning for Healthcare*, 2016.