Predictive Caching for AR/VR Experiences in a Household Scenario

Sharare Zehtabian, Mina Razghandi, Ladislau Bölöni, Damla Turgut
Department of Computer Science
University of Central Florida
Orlando, Florida, USA
sharare.zehtabian@knights.ucf.edu, {mrazghan, lboloni, turgut}@cs.ucf.edu

***Abstract*—Augmented/virtual reality (AR/VR) technologies can be deployed in a household environment for applications such as checking the weather or traffic reports, watching a summary of news, or attending classes. Since AR/VR applications are highly delay sensitive, delivering these types of reports in maximum quality could be very challenging. In this paper, we consider that users go through a series of AR/VR experience units that can be delivered at different experience quality levels. In order to maximize the quality of the experience while minimizing the cost of delivering it, we aim to predict the users' behavior in the home and the experiences they are interested in at specific moments in time. We describe a deep learning based technique to predict the users' requests from AR/VR devices and optimize the local caching of experience units. We evaluate the performance of the proposed technique on two real-world datasets and compare our results with other baselines. Our results show that predicting users' requests can improve the quality of experience and decrease the cost of delivery.**

***Index Terms*—Intelligent Agents, Predictive Caching, Deep Learning, Long Short Term Memory, Augmented/Virtual Reality**

I. Introduction

Augmented Reality and Virtual Reality (AR/VR) technologies recently made significant progress in several application domains where the specific combination of resources and requirements were favorable to their deployment. For instance, in pilot training, computational and network resources can be brought to ensure a high quality experience. In contrast, in casual gaming, the resources are modest, but the expectations towards the AR/VR experience are significantly lower. AR/VR technologies had seen much less penetration in fields where there is a mismatch between the expectation, the quality of the experience and the amount of resources available.

The focus of this paper is the decisions that need to be taken by the AR/VR controller in order to deliver the best overall experience quality to the user. Intuitively, in order to deliver a high-quality experience, the device needs to have sufficient computational resources, display capabilities, and access to the necessary data. There are several ways in which this data can be accessed: locally (having been cached on the local device), downloaded from the network, or computed (for instance, through local rendering).

It would appear that the ideal situation is one where every device has a locally cached copy of the data associated with the highest experience quality, for all possible experiences the user might ask. However, this is a very expensive strategy in terms of bandwidth and energy expenditure. A more intelligent strategy would be to predict what experience will the user request, from what device and when, and cache the data necessary to deliver that experience on the device.

In this work, we are considering the AR/VR for daily use within a household environment, for applications ranging from checking news, weather or traffic reports to attending classes. The users in an AR/VR-enabled home go through a series of interactions split into *experience units* (a certain short interaction with the AR/VR system), delivered through AR/VR devices. Each unit can be delivered at a wide range of *experience quality* levels. For instance, a weather forecast can be delivered as a short text message, or as a dual-4K, immersive visualization. Delivering an experience requires both networking and computing power. The experience quality is limited by the (1) capabilities of the devices through which it is delivered and by the (2) signal limitations such as network delay and bandwidth limitations.

In this paper, we propose an AR/VR controller that is responsible for making decisions about the forms and quality levels at which the experiences are delivered, as well as the background actions, such as predictive caching and pre-rendering of these experiences. The AR/VR controller learns models of the user behavior that allows it to predict the experiences the user will request and adjusts the caching strategy accordingly. One of the major challenges for performing research on AR/VR in a household environment is that relatively few training data is available. To mitigate this problem, we start from two real-world smart home datasets that describe the daily activities of residents. From this data, we extrapolate into synthetic AR/VR datasets that probabilistically associate relevant AR/VR experiences with specific activities.

The remainder of this paper is organized as follows. Related work is discussed in Section II. The problem of delivering the best AR/VR experience is formulated as an optimization problem in Section III. We consider the challenges of modeling user behavior with regards to the AR/VR experiences in Section IV. The suggested predictive caching methods are described in section V. Section VI presents the results of the experiments and section VII concludes the paper.

II. Related Work

The work described in this paper is related to several active research areas in computer networking.

Content/information centric networking (CCN/ICN). The aim is to create a better quality for Internet services in terms of communication bandwidth and users's expectation for a high-speed delivery. In general, CCN allows us to optimize bandwidth utilization and decrease delivery delay by using a caching function inside the intermediate network nodes. A possible approach is for the nodes to cache all content within the network, and in case of any request, they can respond to the user from their sources [1]. However, unnecessary caching increases the network cost, bandwidth utilization, and storage consumption significantly [2]. Thus, how to design an appropriate caching strategy, reducing caching redundancy and increasing cache hit is an active research area [3]

Content caching in AR/VR. In AR/VR applications, deciding what to cache and where to cache are crucial problems [4] and as Sukhmani et al. [5] argues Quality of Experience (QoE) needs to be merged with Quality of Service (QoS).

J. Chakareski [6] designed an optimization framework to maximize the reward that a multi-cellular system can earn when serving AR/VR users by enabling the base stations to select cooperative caching/streaming/edge-computing strategies. Narayanan et al. [7] proposed a seq2seq modeling approach for cache predicting in advance of user request to increase the number of cache hits. Deep Learning also can be useful in content caching optimization with minimizing computation and consequently, delay in the delivery phase [8].

Zhong et al. [9] presented a deep reinforcement learning based algorithm upon Wolpertinger architecture that has better long-term cache hit rates in contrast to LRU, LFU (Least Frequently Used) and FIFO (First In First Out) caching policies.

User modeling. Efficient delivery of experiences requires us to model the user to predict his/her future requests. Real data of users' everyday lives can improve user modelings [10] and observed sample results can be used to make predictions about a user in the context of predictive statistical models. However, such predictions could be very challenging due to the time variation, inter dependency, and periodicity in human behavior. Kurashima et al. [11] proposed a statistical model approach that models these complexities of human behavior.

One of the biggest challenges from the signal processing and machine-learning side remains the generalizability over households [12]. While training for an individual household is easily possible in a lab setup, this approach is not scalable to a real-world scenario with thousands of households and more. A successful approach for generalizability has to consider environmental/climate parameters, building layout, sensor placement, and the behavior of the user.

III. Problem Statement

In this paper, we consider a household scenario where the customers use devices to request AR/VR experiences including (a) summary of the news, (b) weather report, (c) parking availability report, (d) traffic report and (e) food recipe. Statistically, a given type of experience is more likely to be accessed at a specific time. For instance, a recipe might be more likely to be accessed at dinner preparation time. We will consider all experiences split into "units of experiences" which are approximately 15 to 20 seconds long.

TABLE I: Relative quality and caching cost levels of experience units and data chunk size for a 15 to 20 second experience unit.

Type	Size of exp. unit	Relative quality	Relative cost
4K video	32.7 MB	1.00	205.66
HD video (1080p)	10.6MB	0.90	66.67
low res video MKV	483 KB	0.81	3.04
3GP low-res QCIF	159 KB	0.73	1.00
sound only	-	0.66	-
text only	-	0.59	-
3D model animation	2MB-20MB	1.00	125.79

Every experience can be delivered at different levels of quality. Table I (column relative quality) shows the quality levels we are considering in this paper, and the size of the data chunk necessary to deliver the experience unit.

We consider these items in order to define the quality of the experience: user satisfaction (accurate and on time response), less bandwidth utilization (caching useful and more probable requests instead of caching the requests which are not required). It is common to rate the user satisfaction by objective metrics such as video definition (video quality), fluency (interruptions), response speed (initial delay) [13]. According to this, we assess the "user satisfaction" by the score defined as below:

$$score(e_i) = d_d^{d(e_i)} \cdot d_f(f(e_i)) \cdot max\_score \tag{1}$$

in which d_d is the discount factor for missing requests (delay), $d(e_i) = \frac{size\_of\_the\_content}{external\_bandwidth}$ for experience e_i, and $size\_of\_the\_content$ is the content size in MB. Also, d_f is the discount factor of quality of each format and max_score is the value for the maximum quality.

We calculate the cost of caching as $cost = n_c \cdot r_c$, where n_c is the number of cached items and r_c is the relative cost of each type of content which are available in Table I (column relative cost). Since the size of sound-only and text-only formats are very small, we do not consider cache cost for these type of contents. We obtain relative cost by $\frac{size}{min\_size}$ in which $size$ is the content size for a 15 to 20 second experience unit based on type, and min_size is the minimum content size which is for 3GP low-res QCIF type and is equal to 159KB.

We define the final score as a function of user satisfaction and caching cost:

$$final\_score = \alpha \cdot score - \beta \cdot cost \tag{2}$$

where α and β are coefficients for score and cost value, respectively.

IV. User modeling

Our objective is to maximize the quality of the AR/VR experiences for the user, by predicting the time when certain

experiences will be requested and using this information for efficient predictive caching. The prediction of the requests is ultimately rooted in the regularities of everyday life.

One of the challenges of such an approach is the lack of existing datasets for AR/VR requests. As the AR/VR systems are just starting to emerge, no extensive data is yet available. However, the design of the system would need exactly such data to learn the user model. To solve this chicken/egg problem we propose to generate training data starting from real-world user behavior datasets that had been acquired in homes without AR/VR components and augmenting/extending these datasets with logical assumptions about when the user's would have requested experiences, should they have been available.

A. Real-world datasets of user behavior in homes

In the last several years, the emergence of sensor-augmented smart homes made it possible to acquire datasets that track certain aspects of the inhabitants' behavior. In general, tracking the personal life of users opens serious privacy issues. However, several projects captured and made publicly anonymized datasets of human behavior in the home, tracking a project-specific collection of actions. While these actions might not directly map to AR/VR experience requests, they can anchor the generation of training data.

In the work described in this paper, we started from two publicly available datasets:

Dataset 1: The dataset [14] describes the activities of a 26-year-old man in a smart home with 14 state-change sensors installed at doors, cupboards, the refrigerator, and the toilet flush. Sensors were left unattended, collecting data for 28 days in the apartment. Eight annotated activities of daily living (ADLs) were *Shave*, *Brush teeth*, *Get a drink*, *Get dressed*, *Prepare for leaving*, *Prepare brunch* and *Prepare dinner*.

Dataset 2: This dataset [15] collected by CASAS research group describes the activities of 2 residents in an apartment for 57 days. This dataset contains sensor data that was collected in the home of a volunteer adult couple. Residents R1 and R2 can do different tasks in the house. Annotated actions in this dataset includes *Night Wandering*, *Bed to toilet*, *R1 wake*, *R2 wake*, *R2 take medicine*, *Breakfast*, *Leave home*, *Lunch*, *Dinner*, *R1 sleep*, *R2 sleep*, *R1 work in office* and *Laundry*.

B. Creating realistic synthetic datasets of AR/VR experience requests

As the amount of existing datasets for human behavior is limited, it is desirable to extend this collection of datasets with synthetically generated datasets, that allow us to train and validate the proposed algorithms in scenarios that are predictive to the way they will be used in the real world.

Thus, instead of randomly generating or hand engineering specific scenarios (both would lead to unrealistic data) we decided to generate our scenarios by matching the statistical properties of the real-world datasets. A further problem is that our datasets do not have entries for the specific AR/VR experiences we are considering (but they have entries for experiences that correlate with them). To solve this problem, we probabilistically associated certain experiences with activities that are present in the dataset using common-sense associations. For instance, the user is likely to ask for a recipe while preparing dinner. These mappings are shown in the top and bottom part of Table II. We divide the dataset to train, validation and test sets.

TABLE II: Mapping approach from daily task to daily request of the users for dataset 1 (top) and dataset 2 (bottom)

task (Dataset 1)	**corresponding request**
Shave, Brush teeth, Get a drink	Summary of news
Get dressed, Prepare for leaving (30% of the times)	Weather report
Prepare for leaving (50% of the times)	Traffic report
Prepare for leaving (20% of the times)	Parking status
Prepare brunch, Prepare dinner	Recipe

task (Dataset 2)	**corresponding request**
R1 wake, R2 wake	Summary of news
Breakfast (70% of the times), Leave home (30% of the times)	Weather report
Leave home (50% of the times)	Traffic report
Leave home (20% of the times)	Parking status
Breakfast (30% of the times), Lunch, Dinner	Recipe

V. A PREDICTIVE AGENT FOR AR/VR CACHING

In this section, we describe our design for a predictive AR/VR controller which, knowing the preferences and habits of the user, makes intelligent decisions about what to cache. Implemented caching strategies for the proposed AR/VR controller are as follows[1]:

A. Probability-based caching

We define 24 intervals with the length of 1-hour for each day in the datasets. The proposed algorithm is based on calculating the probability of a specific request in a specific time interval by counting the occurrences of the data in the training set. Accordingly, requests with a probability higher than a threshold are cached for each interval in each day. We validate this approach on the different threshold for the number of request occurrence in a certain time interval, then the best result on the validation dataset is applied to the test dataset.

B. LSTM-based caching

The LSTM-based caching algorithm we proposed is based on training a Long Short Term Memory (LSTM) [16] recurrent neural network on the training dataset. The LSTM model is shown in Fig. 1.

Input data is one day requests. Since we do not have many recorded requests in a day, we divide a day to 24 intervals and each has a fixed list of requests shown with 0s and 1s. We have N different type of requests: $\{r_1, r_2, ..., r_N\}$. The data for each interval is a vector x of length N and N is five in our experiments (Summary of news, Weather report, Parking status, Traffic report, Food recipe). The value of element $x_i (i \in \{1, 2, ..., N\})$ is 1 if the request r_i has occurred in the interval, otherwise its value is 0. Furthermore,

[1]The code is available here: https://github.com/sharare90/AR-VR-Research

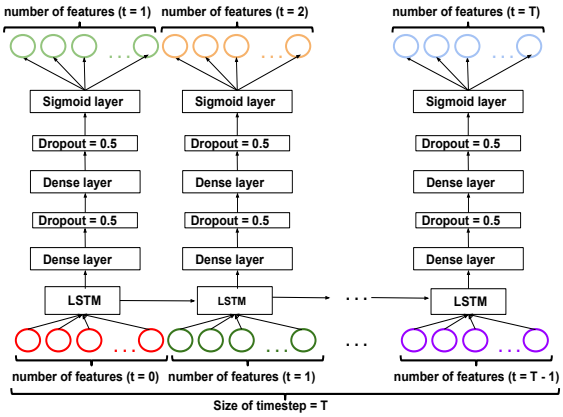

Fig. 1: The neural network used in the LSTM-based caching algorithm.

TABLE III: Selected values for hyperparameters of majority vote-based prediction.

Hyperparameters	Values
learning rate	0.001, 0.01
number of epochs	225, 300, 500, 1000
number of dense layers	2, 3
regularization method	dropout (0.0, 0.2, 0.5, 0.8), l1 and l2

the number of classes equals the number of valid requests. The network processes each interval vector at a time and outputs the occurrence probabilities of requests for the next interval (Fig. 1).

C. Majority vote-based caching

Majority voting is one of the basic prediction/classification methods in which multiple classifiers are used in order to predict the label based on the majority vote of the classifiers [17], [18]. We create 15 different LSTM models by altering hyperparameters such as learning rate, number of epochs, number of layers, or changing regularization method (dropout or l1 or l2 regularization), initial weights, and etc. See Table III for majority voting hyperparameters. After that we predict the label value: $\hat{y} = mode\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_{15}\}$.

VI. Experiments

In this series of experiments, we compared the three predictive caching algorithms that we propose to three baseline algorithms. Thus in the remainder of this section, we refer to the following five caching algorithms:

- Probability-based caching - as described in Section V-A.
- LSTM-based caching - as described in Section V-B.
- Majority vote-based caching - as described in Section V-C.
- Oracle: a caching algorithm where we assume that the algorithm can predict the requests with 100% accuracy.
- Cache everything: an algorithm that caches every possible experience.
- Random caching: this strategy caches a randomly chosen request from the pool of possible requests considering no prior knowledge.

TABLE IV: F1-score of the prediction, using the LSTM model (time interval length = 1 hour) on dataset 1 (top) and dataset 2 (bottom)

Results for dataset 1	Train	Validation	Test
Precision	0.69	0.62	0.52
Recall	0.54	0.52	0.31
F1-Score	0.61	0.54	0.39

Results for dataset 2	Train	Validation	Test
Precision	0.72	0.54	0.54
Recall	0.55	0.43	0.46
F1-Score	0.62	0.47	0.50

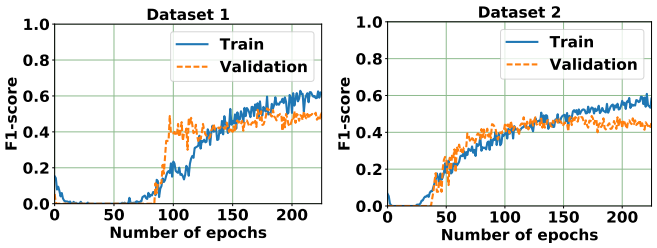

Fig. 2: The F1-score of the prediction, using the LSTM model on dataset 1 (left) and dataset 2 (right) after 225 epochs (time interval length = 1 hour).

We are interested in two performance metrics:

- Prediction accuracy as measured by the F1-score for training phase on different datasets.
- The *final_score*, that combines the cost and user's satisfaction (Equation 2). The less latency that the agent has, the more satisfied the user would be. Also, the more optimized caching, the less cost the predictive model has.

A. Experimental results for training the predictive caching agent

The experimental results for trained LSTM based predictive agent are shown in Table IV for dataset 1 and 2. To prevent overfitting, the training was stopped after 225 epochs. Since the number of 0s is much more than the number of 1s in the LSTM input matrix, accuracy could be very high even if the network predict all of the 1s as 0s. Therefore, we report our results by F1-score alongside with precision and recall to handle datasets unbalances in terms of requests.

Fig. 2 shows the F1-score on the training and validation data for dataset 1 (left) and dataset 2 (right). Fig. 3 shows that by increasing the number of days of data collection in a real situation, the F1-score on validation is approaching the

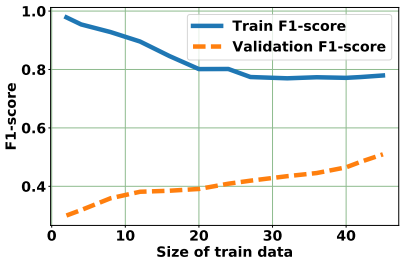

Fig. 3: Train and validation F1-score by increasing the size of data.

TABLE V: Caching approaches final scaled score (Eq 2) results for different delivery formats and with $\alpha = \beta = 1$.

Caching Algorithm	4K video	HD video (1080p)	low-res video MKV	3D model animation
Dataset 1				
Oracle	0.89	0.60	0.30	0.93
Cache everything	0.0	0.31	0.28	0.39
Random	0.06	0.32	0.29	0.39
Probability based	0.40	0.44	0.29	0.62
LSTM based	**0.43**	**0.45**	0.29	**0.64**
Majority Voting	0.37	**0.45**	0.29	0.63
Dataset 2				
Oracle	0.97	0.62	0.30	0.98
Cache everything	0.0	0.31	0.28	0.39
Random	0.04	0.30	0.29	0.40
Probability based	0.62	0.51	0.29	0.76
LSTM based	0.60	0.50	0.29	0.75
Majority Voting	**0.71**	**0.54**	0.29	**0.80**

train F1-score. Therefore, the current gap between train F1-score and validation F1-score is a variance error which can get decreased by training over a bigger dataset. This could be a good direction for future work.

B. Experimental results for the overall agent

If the user sends a request and we have the requested content cached in the system, we increment the score by using Equation 1 and the $d(e_i) = 0$ since there is no delay in this case. However, if the user has a request and the content has not been cached in the system, we need to load it with a delay. The delay depends on the size of the content and external bandwidth. We consider $external\_bandwidth = 100Mbps$ and $max\_score = 1$ for the experiments in this paper. Caching cost is another item which we need to evaluate as the storage could be limited. In order to facilitate the interpretation of the results for each algorithm, we scaled the satisfaction score and caching cost to a number between zero and one. Then we calculate *final_score* (Table V). The *final_score* for 3GP low-res QCIF is zero, since it is the minimum quality for delivery.

It can be inferred from the results in this table that LSTM-based and majority vote-based approaches outperform other approaches in order to optimize caching and have the maximum quality of delivery. This improvement is increasingly significant for higher quality of delivery format.

VII. Conclusions

In this paper, we proposed an approach to perform a local caching of AR/VR experiences for a household scenario. We investigated an approach based on the probability of the individual accesses in specific time slots and an approach that predicts based on the time series of accesses using an LSTM recurrent deep neural network. We also investigated an approach that performs majority voting over multiple LSTM networks with a range of hyperparameters.

To verify the performance of these algorithms, we compared them against several baselines. We found that the majority voting over LSTMs approach yielded the performance that best balances the cost of caching with the perceived experience value. However, we also found that this performance could be significantly improved if more real-world data is available.

Acknowledgement

The support for this work was provided by the National Science Foundation under Award No. 1800961, 1560302, and 1852002. Any opinions, findings, and conclusions and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] F. Li, K.-Y. Lam, L. Wang, Z. Na, X. Liu, and Q. Pan, "Caching Efficiency Enhancement at Wireless Edges with Concerns on User's Quality of Experience," *Wireless Communications and Mobile Computing*, 2018.

[2] F. Qazi, O. Khalid, R. N. B. Rais, I. A. Khan, and A. u. R. Khan, "Optimal Content Caching in Content-Centric Networks," *Wireless Communications and Mobile Computing*, vol. 2019, 2019.

[3] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip, "Cache-aided coded multicast for correlated sources," in *Proc. of the International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2016, pp. 360–364.

[4] E. Bastug, M. Bennis, M. Médard, and M. Debbah, "Toward interconnected virtual reality: Opportunities, challenges, and enablers," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 110–117, 2017.

[5] S. Sukhmani, M. Sadeghi, M. Erol-Kantarci, and A. El Saddik, "Edge Caching and Computing in 5G for Mobile AR/VR and Tactile Internet," *IEEE MultiMedia*, vol. 26, no. 1, pp. 21–30, 2019.

[6] J. Chakareski, "VR/AR immersive communication: Caching, edge computing, and transmission trade-offs," in *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, 2017, pp. 36–41.

[7] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "Deepcache: A deep learning based framework for content caching," in *Workshop on Network Meets AI & ML, (NetAI'18)*, 2018, pp. 48–53.

[8] L. Lei, L. You, G. Dai, T. X. Vu, D. Yuan, and S. Chatzinotas, "A deep learning approach for optimizing content delivering in cache-enabled HetNet," in *Proc. of the International Symposium on Wireless Communication Systems (ISWCS)*, 2017, pp. 449–453.

[9] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. of the Annual Conference on Information Sciences and Systems (CISS)*, 2018, pp. 1–6.

[10] F. Cena, S. Likavec, and A. Rapp, "Real world user model: Evolution of user modeling triggered by advances in wearable and ubiquitous computing," *Information Systems Frontiers*, vol. 21, no. 5, pp. 1085–1110, 2019.

[11] T. Kurashima, T. Althoff, and J. Leskovec, "Modeling interdependent and periodic real-world action sequences," in *Proc. of the World Wide Web Conference*, 2018, pp. 803–812.

[12] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer, "Monitoring activities of daily living in smart homes: Understanding human behavior," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 81–94, 2016.

[13] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[14] T. L. van Kasteren, G. Englebienne, and B. J. Kröse, "Human activity recognition from wireless sensor network data: Benchmark and software," in *Activity Recognition in Pervasive Intelligent Environments*. Springer, 2011, pp. 165–186.

[15] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: A smart home in a box," *Computer*, vol. 46, no. 7, pp. 62–69, 2013.

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

[17] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. of the International Workshop on Multiple Classifier Systems*. Springer, 2000, pp. 1–15.

[18] Y. Guan and T. Plötz, "Ensembles of deep lstm learners for activity recognition using wearables," *Proc. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, p. 11, 2017.