Combining Symbolic Reasoning and Deep Learning for Human Activity Recognition

Fernando Moya Rueda[1], Stefan Lüdtke[2], Max Schröder[3], Kristina Yordanova[2], Thomas Kirste[2], Gernot A. Fink[1]
[1] *Department of Computer Science*, *TU Dortmund University*, Dortmund, Germany
[2] *Department of Computer Science*, *University of Rostock*, Rostock, Germany
[3] *Department of Communications Engineering*, *University of Rostock*, Rostock, Germany
{fernando.moya, gernot.fink}@tu-dortmund.de, {stefan.luedtke2, max.schroeder, kristina.yordanova, thomas.kirste}@uni-rostock.de

***Abstract*—Activity recognition (AR) plays an important role in situation aware systems. Recently, deep learning approaches have shown promising results in the field of AR. However, their predictions are overconfident even in cases when the action class is incorrectly recognized. Moreover, these approaches provide information about an action class but not about the user context, such as location and manipulation of objects. To address these problems, we propose a hybrid AR architecture that combines deep learning with symbolic models to provide more realistic estimation of the classes and additional contextual information. We test the approach on a cooking dataset, describing the preparation of carrots soup. The results show that the proposed approach performs comparable to state of the art deep models inferring additional contextual properties about the current activity. The proposed approach is a first attempt to bridge the gap between deep learning and symbolic modeling for AR.**

***Index Terms*—Human Activity Recognition, Symbolic Reasoning, Deep learning**

I. Introduction

To provide context-aware services, a system has to be able to reason about the user's whereabouts [13]. Human activity recognition (HAR) addresses the problem of identifying person's activities based on, usually sensor-based, observations. There are two main HAR paradigms: data-driven and knowledge-based (or context aware) [1], [22]. Data-driven approaches rely on large datasets from which a model for a specific problem is learnt. In the recent years, deep learning approaches have shown promising results in HAR especially when given enough training data [6], [8], [16], [21]. However, their predictions could be overconfident, leading to very accurate recognition of classes with enough training samples and to very confident incorrect prediction of classes with few training samples [7]. Besides, data-driven approaches do not incorporate a semantic structure of the recognized activities, which if present, would have allowed reasoning not only about the actions being executed, but also about user goals, situation, and causes of behavior [1]. To address these limitations, knowledge-based approaches rely on symbolic models describing the possible behaviors to reason about the user's actions and whereabouts [17]. One problem with these approaches is that they are often unable to cope with uncertainty in the observations. To solve this, some works propose the combination of symbolic models and probabilistic reasoning [24]. This type of models is also known as computational state space models (CSSM) [12].

In this work, we go a step further and combine the computational power of deep learning and the ability of CSSMs to reason about a person's context. We introduce a joint approach that combines convolutional neural networks (CNNs) with CSSMs to produce additional information about a user's situation. We call this approach Hybrid Computational Causal Behaviour Models (HCCBM). We show that the combined approach does not reduce the recognition performance compared to pure CNNs, but it provides additional information, which the pure neural networks are not able to provide.

The paper is structured as follows. In Section II, we discuss the related work in deep learning architectures and symbolic models for HAR. Section III describes the methods and the proposed combined approach. We evaluate the approach in Section IV and we conclude the work in Section V.

II. Related Work

Deep architectures are the state of the art in different fields of pattern recognition, e.g., image classification and segmentation, word spotting and speech recognition. In HAR applications, deep architectures have been applied to multichannel time series in a sliding window framework [6], [8], [16], [19], [21]. Convolutional neural networks (CNNs) capture the complex human movements by combining the feature extraction and classification. By learning features directly from raw data in conjunction with the classifier, CNNs become more discriminative with respect to human actions exploiting their hierarchical composition. CNNs find temporal relations, invariant to small temporal translations and noise, from raw data by combining different non-linear feature extractors and pooling operations. The authors in [19], [21] proposed a CNN with convolutional layers applied along the time axis, called temporal-convolution layers. They applied the convolutions over single or all sensors' measurements. These CNNs are rather shallow in comparison with the ones used for image classification [10]. They have a maximum of four temporal-convolution layers followed by a pooling layer and a softmax classifier. In [16], CNNs were combined with recurrent neural networks (RNNs). The later ones are suitable for processing

978-1-5386-9151-9/19/$31.00 ©2019 IEEE

sequences. The authors replaced the fully-connected layers by two layers of long short-term memories (LSTMs). In [8], the authors compared a multilayer perceptron (MLP), a CNN, similar to [19], a three-layer LSTM and a one-layer BLSTM networks. BLSTM are similar to LSTM, however, they process sequences following forward and backward directions. Though the BLSTM network performed better, the CNN showed to be more robust against parameter changes. In general, CNNs tend to be over-confident in their predictions. Besides, there is no measure of uncertainty of their predictions. The authors in [5] argued that applying dropout is similar to train an ensemble of different networks. Using dropout, a random set of neurons is dropped from the network. Therefore, multiple paths that produce the same output are learnt. The authors in [4] showed that using dropout to each layer can be considered as a Gaussian process; thus, the predictive mean, variance and uncertainty can be derived. The authors in [7] extended these considerations by adjusting the CNN's predictions based on uncertainty information using Dropout in testing.

In difference to the above mentioned approaches, knowledge-based approaches rely on a model of the possible actions and context [2], [20]. The model is often represented in the form of ontology [17] based on which the system can infer the actions and context. One problem with these approaches is that they are often unable to cope with ambiguous or missing observations. To cope with this problem, some approaches propose the combination of symbolic models with probabilistic reasoning [9], [12], [18]. These approaches use concise rule-based representation of the possible actions and the relevant context and probabilistic inference engines such as Bayesian inference to reason about the observed actions and context in probabilistic manner. Such type of approaches are also known as computational state space models (CSSMs) [12]. Another of their advantage is that one needs just a few rules, based on which rich behavior models with millions of possible executions sequences and variation in context can be generated [24]. This is opposed to ontology based approaches where all behavior variations have to be manually modeled. While most of the existing CSSM approaches are able to generate models with up to tens of thousands of states, there are some CSSMs that can perform adequate inference in models with hundreds of millions of states. One such approach is Computational Causal Behavior Models (CCBM), which uses an extension of the Planning Domain Definition Language (PDDL) to define the rules and particle or marginal filter for probabilistic inference [14]. In this work, we combine CCBM with CNNs to provide context-aware activity recognition.

III. Methods

We propose an approach that uses deep architectures as expected observations for computational state space models to produce additional information about a user's situation. In what follows, we first introduce the CNN used for processing multichannel time-series for human activity recognition. We then describe the procedure for calculating pessimistic predictions of CNNs. Finally, we introduce the CCBM model.

A. Deep architecture

We used two CNN architectures for predicting human actions from multichannel-time series, following the architecture designs in [16] for recognizing activities of daily life and in [6] for recognizing activities in order picking scenarios. Multichannel-time series are recordings from Inertial Measurement Units (IMUs) that are attached to the human body. The architectures are composed of convolutional, pooling and fully-connected layers. Their convolution and pooling operations are carried out along the time axis where temporal-convolution layers extract temporal local features, and pooling operations introduce temporal invariance. Subsequent fully-connected layers combine the local features and create an abstract representation of an input sequence. A temporal convolution layer can be interpreted as a $3D$ volume of filters [6], [16], [19], [21]. It consists of C number of learnable filters that activate to certain type of feature at some temporal state in a feature-map input X^i producing a feature-map X^j of C number of activations. A temporal convolution connecting layers i and j is a combination of: first, a convolution operation between the input feature-map $X_{i[T,D,C^i]}$ and each of the filters $w_{[F,1,C^i]}$ in layer j. Where, T is the length of the sequence, D the number of sensors, C the number of feature maps, and F the length of the filter; second, an addition of a bias $b \in B^j$; and third, a non-linear activation function $\sigma(\cdot)$. The activation function is ReLU.

A max-pooling layer is used for downsampling a feature-map. A CNN for classification uses a softmax activation layer, which computes pseudo-probabilities of an input sequence X^{input} belonging to a class $k_i \in K$. The network's input X^{input} is a sequence segment from D number of different sensors. These segments are extracted following a sliding-window approach with a window's size of T and a step of s [3], [6], [16]. The input's size is $[T, D, 1]$. During training, sequence segments X^{input} with label y from a set of labels Y are forwarded to the network and the cross-entropy loss between the estimated probabilities x_k^{output} and the corresponding target label $y_k \in Y$ is minimized. The first CNN architecture, based on [6], contains B parallel branches. Each branch has sequences per IMU. The branches are composed of two blocks of two temporal convolution and a max-pooling layers. A fully-connected layer creates an intermediate representation of the input data per IMU. The network combines the intermediate representations using a MLP and forwards them to a softmax layer, see Fig. 2. Each convolutional layer has $C = 64$ filters of size $[5 \times 1]$. Max-pooling layers with receptive field of $[2 \times 1]$ and stride of 2 are used. Fully-connected layers contain 256 units. This architecture is called CNN-IMU. The second architecture is a simplification of CNN-IMU with a single branch, which processes the sensor data from all of the IMUs (we call it CNN).

B. Pessimistic predictions

CNNs output overconfident predictions leading to either very accurate predictions or very confident and incorrect ones. As shown in [4], a CNN with dropout after every weight

layer can be seen as an approximation of a deep Gaussian process. Let $W_{j,i}\ \forall j = [1,2,...,J_i]$ be the weights of neuron j in layer $i = [1,2,...,L]$ and $\omega_{j,i}\ \forall j = [1,2,...,J_i]$ a set of binary variables that take the value 1 with probability p_i and 0 with probability $1-p_i$. Weights W_i^j are dropped when ω_i^j is set to 0. Feeding R repetitions often input sequence X^{in} with label y_k to a CNN at testing and ω is randomly drawn for each repetition r, the CNN's output is normal distributed. Then, the predictive mean μ, see Equation 1, and the predictive variance $\sigma = \sqrt{Var(y)}$, see Equation 2, of the CNN's output $\hat{y}$ can be computed.

$$\mu(y) \approx \frac{1}{R}\sum_{r=1}^{R}\hat{y}(X^{in}, W_i^r)\ \forall i = [1,2,...,L] \tag{1}$$

$$Var(y) \approx \tau^{-1} I_{[K]} + \frac{1}{R}\sum_{r=1}^{R}\hat{y}(X^{in}, W_i^r)^T\hat{y}(X^{in}, W_i^r)\ -\ E(y)^T E(y) \quad \forall i = [1,2,...,L], \tag{2}$$

where $\tau^{-1} = \frac{pl^2}{2N\lambda}$, being p the inverse of the dropout probability, l the prior length scale, N the number of input samples, and λ the weight decay. $I_{[c]}$ denotes a vector of ones of length K. Under the assumption of this deep Gaussian process, a confidence interval can be computed, see Equation 3 [7]. This interval provides upper and lower bounds for the predictions defining an optimistic and pessimistic behavior. The upper bound or the optimistic prediction tends to favor larger values of the output. Contrary, the lower bound or pessimistic prediction favors numerically smaller values.

$$\left[E(y) - z_{1-\frac{\alpha}{2}}\frac{\sigma(y)}{\sqrt{R}}\ ;\ \ E(y) + z_{1-\frac{\alpha}{2}}\frac{\sigma(y)}{\sqrt{R}}\right], \tag{3}$$

with $z_{1-\frac{\alpha}{2}}$ being $1-\frac{\alpha}{2}$ quantile of the normal distribution.

The pessimistic prediction is essential for the ability of the CCBM model to perform inference. As CCBM relies on dynamic Bayesian inference, overly confident incorrectly recognized classes lead to the inability of CCBM to perform sequential inference (for more details on CCBM see below). The pessimistic prediction, however, allows the possibility of inferring the correct class with CCBM even if it was incorrectly inferred with the deep model.

C. Computational Causal Behaviour Models

Our investigation uses Computational Causal Behaviour Models (CCBM), an implementation of Computational State Space Models. CCBM aims to estimate the state sequence of a dynamic system from noisy and ambiguous sensor data.

CCBM uses an action language based on the Planning Domain Definition Language (PDDL) to describe actions possible in a given problem domain by means of preconditions and effects. We call the rules that describe a given action "action templates". Fig. 1 shows an example of an action template for the action "take". The **:parameters** clause indicates the types of elements to which this action can be applied. In this case, the action can be executed on an element of type object at a given location[1]. The precondition then is that the object is at the given location. If the action is executed, the effect is that the object is no longer at that location and that the object has been taken. Apart from the preconditions and effects, an action template contains a mapping from high level action to the corresponding action duration and the expected observation. The first is done with the **:duration** clause. The duration can be a probability distribution, an exact duration or empirically estimated from the training data duration. The link to the observations is done through the **:observation** clause, which links the action to a function in the observation model.

[1]Note that "?" indicates a variable and the name after "-" is the type of this variable

```
(:action take
  :parameters (?o - object ?from - location)
    :duration (duration (dur-id take ?o ?from))
    :saliency 2
  :precondition (and
          (at ?o ?l))
  :effect   (and
          (not (at ?o ?l))
          (taken ?o))
    :observation  (setActivity (activity-id take ?o ?from))
)
```

Fig. 1: Action template *take* in the PDDL notation.

Given the action templates, the initial state distribution of the problem, and the goals, one can generate a directed graph from the initial to the goal states. The probability of choosing a given action in a certain state is given through action selection heuristics. The probability of choosing action a in state x is defined as follows:

$$p(a \mid x) \propto \exp(\sum_{k=1}^{3}\lambda_k f_k(a,x)) \tag{4}$$

$$f_1(a,x) = \log\gamma(a(x)) \tag{5}$$

$$f_2(a,x) = \log s(a) \tag{6}$$

$$f_3(a,x) = \delta(a(x)) \tag{7}$$

$\gamma(a(x))$ is the revisiting factor that is 0 if the state $x' = a(x)$ (i.e. the state resulting from applying a to x) has already been visited. The revisiting factor controls whether already visited states can be visited again; $s(a)$ is the saliency of a that is specified in the action template. The saliency assigns a weight to the action with respect to the remaining actions; and $\delta(a(x))$ is the goal-distance of state $x' = a(x)$. The goal distance assumes that actions closer to the goal are selected with a higher probability. Each feature can be weighted with λ_k. The usage of these parameters depends on the specific type of behaviour to be modelled. For example, goal oriented behaviour can be controlled through the goal distance, which is automatically calculated by searching the state space for the shortest path from the initial state to the goal.

In that manner, CCBM is able to reason about the most probable action given the previous state of the user. CCBM makes use of Bayesian Filtering methods to estimate the state and action sequences from sensor data. It makes a prediction of the current state x_t at time step t based on the previous state x_{t-1} and the transition probabilities $p(x_t \mid x_{t-1}) = \sum_{x_t=a(x_{t-1})} p(a \mid x_t)$ that are generated from the action distribution in Equation 4.

$$p(x_t \mid y_{1:t-1}) = \int_X p(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1})dx \qquad \textit{prediction} \tag{8}$$

Based on the predicted state, it then takes the observation into account (given by $p(y_t \mid x_t)$) and makes a correction of the estimated state (Equation 9)

$$p(x_t \mid y_{1:t}) = \frac{p(y_t \mid x_t)p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})} \qquad \textit{correction} \qquad (9)$$

As exact inference in large state spaces is unfeasible, CCBM uses particle filtering or marginal filtering (a variant of particle filtering for categorical domains [15]) to perform approximate inference. As the marginal filter has been shown to be superior to the particle filter in categorical domains, we use marginal filtering in this work.

D. Hybrid Computational Causal Behaviour Models

Our combined approach uses the CCBM architecture, with the difference that the expected observations are replaced with the output from the deep model. We call this new approach Hybrid Computational Causal Behaviour Models (HCCBM). This is done in the following manner.

1) the deep architecture is used to learn and predict the observed actions;
2) the output of the deep model is used as observations for CCBM, providing a high-level observation sequence that is used to update the state prediction of the CCBM by weighting each state by the probability of the corresponding action class, as indicated by the deep model;
3) the action templates, initial and goal states are manually defined;
4) the observation model $P(y \mid x)$ is generated from the distribution of the action classes, given the current observation obtained from the deep model;
5) the action templates, initial and goal states together with the observation model are compiled into a marginal filter;
6) the filter estimates the action class and contextual properties for each time step from the deep model's output.

The procedure is illustrated in Fig. 2.

IV. Evaluation

We evaluate the approach on a multichannel time-series dataset, which consists of measurements of persons preparing a carrot soup [11].

A. Dataset

A typical meal time routine was selected as trial setting, consisting of the following phases: (i) Prepare meal (prepare ingredients; cook meal). (ii) Set table. (iii) Eat meal. (iv) Clean up and put away utensils. (A symbolic map of the spatial structure of the trial domain and the involved domain objects are given in [11]). The dataset contains recordings from 7 persons performing $K = 16$ classes, which describe preparing a carrot soup. The recording contains measurements from five inertial measurement units (IMUs), see Fig. 2, with a total of $D = 30$ sensors. The recording sample is $110Hz$. A sliding window of $T = 64$ or $576ms$ and a step size of $s = 5$ or $45ms$ are used for segmenting sequences. The window step is small for generating a higher quantity of samples. A segment sequence is assigned the most frequent action class $y \in Y_k$ within.

B. Predictions using Convolutional Neural Networks

We evaluate the performance of the two deep architectures, see Subsection III-A, on the cooking dataset. As the data is scarce, we followed a k-cross testing for setting the training and testing sets, where the segment sequences of one person is taken for testing x^{test} and the rest for training X^{train}. For training, all sequence segments $x \in X^{train}$ and their respective $y \in Y_k^{train}$ are fed to a network at random as follows: parameters are updated by minimizing the cross-entropy loss, using batch gradient descent with RMSProp update rule, as in [16]. We use a base learning rate of $5x10^{-5}$, RMS decay of 0.95 and batch size of 128. Dropout of $p = 50\%$ on the last two fully-connected layers is used. As the learning rate is relatively small, networks are trained to a maximum of 100 epochs, an epoch being the number of times the training algorithm uses the entire training set. We used Dropout on the first and second fully-connected layer. Sensor values are normalized to a range of $[0.1]$, and a Gaussian noise with zero mean and $\sigma = 5x10^{-4}$ is added for simulating sensor's noise, as suggested in [6], [16]. The CNN-IMU contains $B = 5$ branches, following the number of IMUs from the dataset.

For testing, sequence segments X^{test} are forwarded to the network, computing class probabilities x_k^j. The label corresponding to the maximum probability in x_k^j is taken as the activity class $k \in K$. The accuracy and weighted F1 ($wF1$) measures are used as performance measures, in which the predicted activity class is compared with the expected activity target. In addition, sequence segments X^{test} are forwarded to the network, which uses dropout in testing, $R = 100$ times for computing pessimistic predictions, see Subsection III-B. We have used $\alpha = 0.045$ with $1 - \frac{\alpha}{2} \approx 0.977$ for adapting the CNN's predictions. Table I shows the accuracies and $wF1$ for each of the persons' sequences as testing sets. Differently from [6], the architecture CNN, containing a single branch processing segments from all the D sensors, shows better performance than the CNN-IMU, containing $B = 5$ branches. The performance on person 3 shows the worst accuracy for both networks. The accuracy and $wF1$ of the networks when using pessimistic predictions is, respectively, around 0.8% and 1.7% lesser than using the original predictions.

Table II shows the accuracy per action class $k \in K$ for both the networks and their pessimistic predictions. Activities such as "wash", "move", and "eat" present a classification accuracy superior than 75% with a relative low deviation. However, activities like "turn on" and "turn off" show relative low accuracies. Using the CNN, these classes are not recognized. This is explained with the highly unbalanced dataset, as these activities contain small quantity of samples in comparison with other activities. Using pessimistic predictions did not increase the over-all performance of the networks. However, they allow the better recognition of infrequent classes.

C. Evaluating the Hybrid CCBM Approach

To evaluate the hybrid CCBM approach, we used the output of P.CNN with $\alpha = 0.045$ with $1 - \frac{\alpha}{2} \approx 0.977$ as input observations for the HCCBM model. The HCCBM

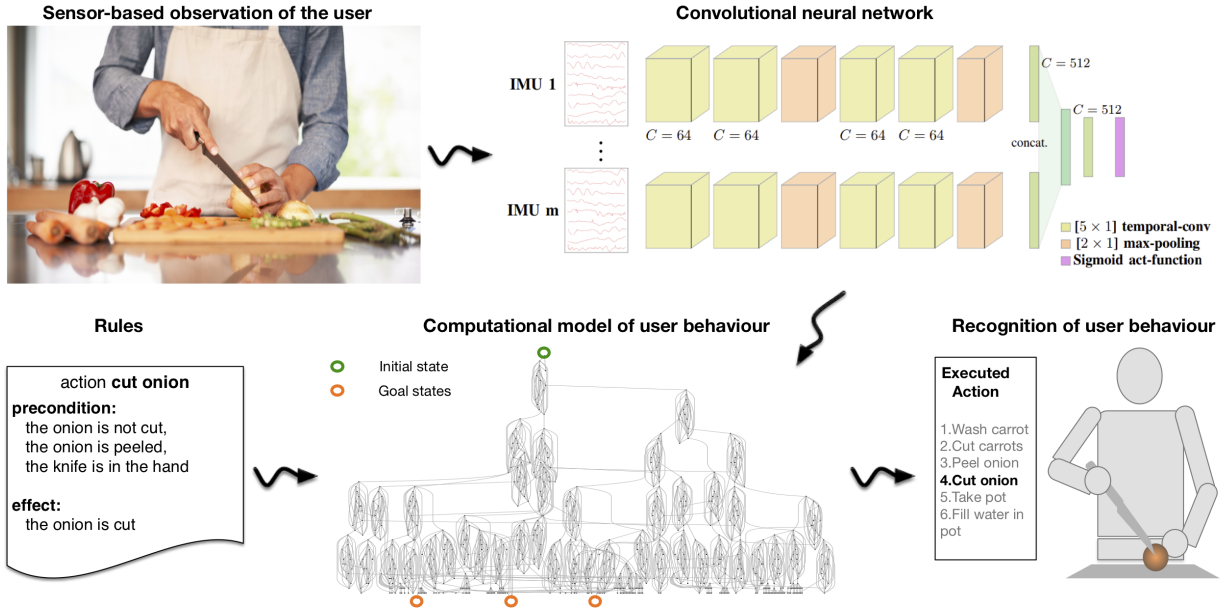

Fig. 2: The hybrid CCBM approach combining CNNs and CCBM to provide additional reasoning about the situation.

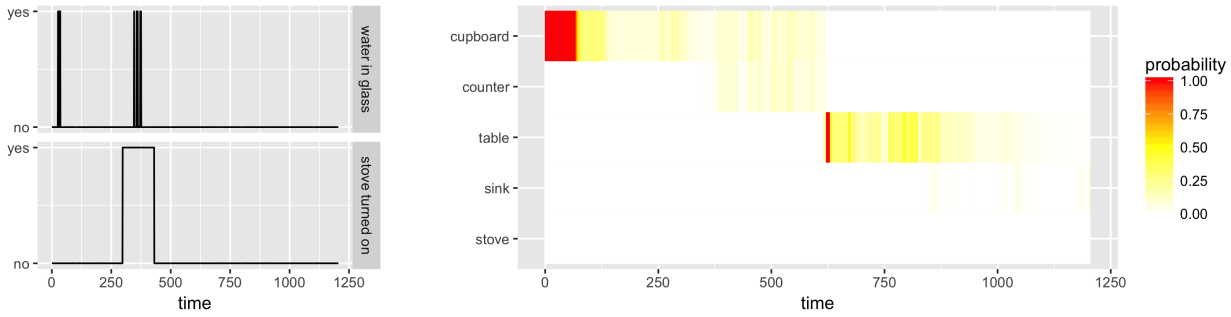

Fig. 3: Left: answering "Is water in the glass?" and "Is the stove turned on?" (solid line shows the most probable state of the property at a given time). Right: inference about the plate's location (the darker the color, the more probable the location).

TABLE I: Classification accuracy [%] and $wF1$ [%] for the cooking dataset for each person's sequences taken as a testing set. P. CNN refers to pessimistic predictions and HCCBM refers to the hybrid approach.

Acc.	**Person**							
Networks	1	2	3	4	5	6	7	**Average**
CNN	56.2	**69.8**	55.3	61.4	**58.4**	**66.2**	65.2	**61.8 ± 5.5**
P. CNN	54.2	67.5	54.3	59.3	57.6	64.8	**68.8**	60.9 ± 6.1
CNN-IMU	54.2	62.0	49.7	57.8	58.1	64.5	66.2	58.9 ± 5.8
P. CNN-IMU	**56.8**	60.0	46.1	57.6	57.1	62.6	66.8	58.1 ± 6.4
HCCBM	46.7	68.9	48.0	**63.5**	44.0	66.1	64.1	57.3 ± 10.6
P-HCCBM	54.3	66.5	**58.2**	62.1	56.6	64.2	66.2	61.2 ± 4.8
wF1	**Person**							
Networks	1	2	3	4	5	6	7	**Average**
CNN	**59.0**	**71.6**	56.3	**63.1**	**60.4**	**67.1**	65.6	**63.3 ± 5.3**
P. CNN	54.7	68.5	54.6	60.1	59.8	64.6	**69.6**	61.7 ± 6.1
CNN-IMU	57.1	64.4	52.3	59.5	61.5	66.2	67.6	61.2 ± 5.4
P. CNN-IMU	58.7	62.0	48.2	58.6	58.0	62.6	67.9	59.4 ± 6.0
HCCBM	47.1	67.0	47.5	60.9	40.8	64.9	63.8	56.0± 10.6
P-HCCBM	52.4	65.0	**57.0**	60.1	55.0	65.1	66.1	60.1 ± 4.8

TABLE II: Classification accuracy [%] for each of the activity classes in the carrot dataset for the two architectures and the pessimistic predictions. Here, / stands for ±.

Activity	**Accuracy**					
	CNN	P. CNN	CNN-IMU	P. CNN-IMU	HCCBM	P-HCCBM
wash	82.7/6.2	81.4/6.1	**86.5/6.6**	79.5/8.7	76.7/17.5	84.6/6.0
wait	42.4/18.1	39.9/17.2	34.3/21.3	35.5/17.9	**48.6/18.2**	44.3/19.3
move	74.1/15.1	75.0/13.7	73.8/12.5	75.4/12.2	76.4/15.2	**77.4/16.2**
take	31.5/11.9	28.5/7.1	22.2/9.1	29.7/10.5	**32.0/12.5**	31.4/10.9
put	35.1/15.2	**35.3/16.1**	33.3/14.5	32.9/12.4	31.9/12.3	31.6/14.5
cut	83.8/23.2	84.4/19.0	82.1/29.0	78.3/32.1	86.6/21.7	**87.4/13.4**
fill	51.3/9.1	**53.4/5.8**	51.2/10.0	48.5/13.1	26.9/9.5	40.5/18.5
turn on	0.0/0.0	6.4/10.0	4.5/9.4	**13.6/24.8**	0.0/0.0	3.6/9.45
cook	82.8/9.8	**83.2/8.4**	64.5/36.7	62.8/36.2	69.3/32.9	75.3/20.9
turn off	1.0/2.7	46.4/41.5	30.0/36.4	45.9/39.6	0.0/0.0	**46.8/42.4**
open	25.4/23.7	**45.8/23.0**	36.4/19.9	45.4/20.6	26.2/21.4	38.5/22.7
close	44.0/22.7	43.9/23.6	43.6/16.3	41.4/27.5	**46.3/19.2**	42.3/16.5
sit down	34.3/34.6	**52 5/18.9**	49.2/19.6	47.5/23.0	37.4/35.5	48.4/23.6
eat	**91.6/5.8**	81.5/21.2	84.1/20.9	84.8/16.8	81.6/36.2	87.9/12.6
drink	39.4/31.9	**77.7/20.4**	76.1/35.8	70.1/33.2	42.6/32.4	49.6/31.2
stand up	11.7/17.4	**48.3/38.8**	44.8/39.9	43.0/40.2	4.2/11.0	32.2/39.1

model consisted of 44 action templates, 99 ground actions, $> 1.47 \times 10^8$ states and $> 6.13 \times 10^8$ plans (execution sequences leading from the initial to the goal states). Due to the very large state space, marginal filtering [15] was used to

estimate the current state and the corresponding properties. For the action selection heuristics, we used the goal distance as a heuristic. Revisiting already visited states was also enabled. The rest of the heuristics were not used. The last row of Table I shows the overall accuracy and F1 score for HCCBM. It can be seen that HCCBM had a bit lower performance than the CNN (although in the case of person 2, it outperformed the deep models). Applying Wilcoxon-Mann-Whitney test on the results from the CNN and the HCCBM shows that there is no significant difference in the results (p-value of 0.1282). This indicates that the approach performs comparable to state of the art deep models and that the symbolic model does not significantly reduce the recognition rate. Looking at the class-wise performance in Table II, it can be seen that HCCBM outperformed the remaining models for the action classes "wait", "move", "take", "put", and "cut". It was, however, unable to recognize the actions "drink" and "stand up". This can be explained with the fact that HCCBM performs sequential state estimation. If a given action was not correctly recognized, then the preconditions for the next action were not met, thus the action was not recognized. For example, for the action "drink", it is often that the earlier action "fill water" was not recognized.

Although we used the pessimistic CNN to reduce the effect of incorrectly identified actions, HCCBM still had problems with actions with very low probability. In difference to standard deep learning approaches, however, the HCCBM enables answering application specific questions such as "Is water in the glass?", "Is the stove turned on?", or "Where is the plate?". Fig. 3 illustrates these inference targets for an excerpt of the data. This is possible thanks to the symbolic structure of the model. It allows reasoning about context elements related to the executed actions, causal relations, and goals the person is following. This makes HCCBM a powerful tool for reasoning beyond the action classes.

V. Conclusion

In this work, we presented a hybrid approach for activity recognition that combines deep learning methods with symbolic modeling and Bayesian inference. The approach is able to recognize a person's actions, locations, objects being manipulated as well as causal dependencies between actions and elements in the environment. The results showed that the approach performs comparable to state of the art deep learning approaches. It also provides additional context information that the deep models are not able to infer.

In the future, we plan to test the approach on new datasets from the warehouses domain recorded both in laboratory and real settings. Furthermore, we plan to replace the manual rule definition with an automatic generation from textual sources as proposed in works such as [23].

References

[1] Chen, L., Hoey, J., Nugent, C.D., Cook, D.J., Yu, Z.: Sensor-based activity recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42(6), 790–808 (2012)

[2] Chen, L., Nugent, C., Okeyo, G.: An ontology-based hybrid approach to activity modeling for smart homes. IEEE Trans. on Human-Machine Systems 44(1), 92–105 (Feb 2014)

[3] Feldhorst, S., Masoudenijad, M., ten Hompel, M., Fink, G.A.: Motion classification for analyzing the order picking process using mobile sensors. In: Proc. Int. Conf. Pattern Recognition Applications and Methods. pp. 706–713. SCITEPRESS (2016)

[4] Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: Int. Conf. on machine learning. pp. 1050–1059 (2016)

[5] Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout networks. Proc. Conf. computer vision and pattern recognition pp. 770–778 (2013)

[6] Grzeszick, R., Lenk, J.M., Rueda, F.M., Fink, G.A., Feldhorst, S., ten Hompel, M.: Deep neural network based human activity recognition for the order picking process. In: Proc. Int. Workshop on Sensor-based Activity Recognition and Interaction. ACM (2017)

[7] Grzeszick, R., Sudholt, S., Fink, G.A.: Optimistic and Pessimistic Neural Networks for Object Recognition. In: Proc. Intl. Conf. on Image Processing (2017)

[8] Hammerla, N.Y., Halloran, S., Ploetz, T.: Deep, convolutional, and recurrent models for human activity recognition using wearables. arXiv preprint arXiv:1604.08880 (2016)

[9] Hiatt, L.M., Harrison, A.M., Trafton, J.G.: Accommodating human variability in human-robot teams through theory of mind. In: Proc. Int. J. Conf. Artificial Intell. pp. 2066–2071. AAAI, Barcelona, Spain (2011)

[10] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)

[11] Krüger, F., Hein, A., Yordanova, K., Kirste, T.: Recognising user actions during cooking task – imu data. University of Rostock (2017), http://purl.uni-rostock.de/rosdok/id00000154

[12] Krüger, F., Nyolt, M., Yordanova, K., Hein, A., Kirste, T.: Computational state space models for activity and intention recognition. a feasibility study. PLoS ONE 9(11), e109381 (11 2014)

[13] Krüger, F., Yordanova, K., Burghardt, C., Kirste, T.: Towards creating assistive software by employing human behavior models. J. Ambient Intelligence and Smart Environments 4(3), 209–226 (May 2012)

[14] Nyolt, M., Krüger, F., Yordanova, K., Hein, A., Kirste, T.: Marginal filtering in large state spaces. Int. Journal of Approximate Reasoning 61, 16–32 (June 2015)

[15] Nyolt, M., Krüger, F., Yordanova, K., Hein, A., Kirste, T.: Marginal filtering in large state spaces. Int. J. Approx. Reasoning 61, 16–32 (June 2015)

[16] Ordóñez, F.J., Roggen, D.: Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors 16(1), 115 (2016)

[17] Rafferty, J., Nugent, C.D., Liu, J., Chen, L.: From activity recognition to intention recognition for assisted living within smart homes. IEEE Trans. on Human-Machine Systems 47(3), 368–379 (June 2017)

[18] Ramirez, M., Geffner, H.: Goal recognition over pomdps: Inferring the intention of a pomdp agent. In: Proc. J. Conf. Artificial Intelligence. pp. 2009–2014. AAAI Press, Barcelona, Spain (2011)

[19] Ronao, C.A., Cho, S.B.: Deep convolutional neural networks for human activity recognition with smartphone sensors. In: Int. Conf. on Neural Information Processing. pp. 46–53. Springer (2015)

[20] Roy, P.C., Giroux, S., Bouchard, B., Bouzouane, A., Phua, C., Tolstikov, A., Biswas, J.: A possibilistic approach for activity recognition in smart homes for cognitive assistance to Alzheimer's patients. In: AR in Pervasive Intelligent Environments, Atlantis Ambient and Pervasive Intelligence, vol. 4, pp. 33–58. Atlantis Press, Amsterdam (2011)

[21] Yang, J., Nguyen, M.N., San, P.P., Li, X., Krishnaswamy, S.: Deep convolutional neural networks on multichannel time series for human activity recognition. In: IJCAI. pp. 3995–4001 (2015)

[22] Ye, J., Stevenson, G., Dobson, S.: Usmart: An unsupervised semantic mining activity recognition technique. ACM TiiS 4(4), 16:1–16:27 (2014)

[23] Yordanova, K.: Extracting planning operators from instructional texts for behaviour interpretation. In: German Conf. on Artificial Intelligence. Berlin, Germany (Sptember 2018)

[24] Yordanova, K., Kirste, T.: A process for systematic development of symbolic models for activity recognition. ACM TiiS 5(4), 20:1–20:35 (Dec 2015)