Comparing Hidden Markov Models and Long Short Term Memory Neural Networks for Learning Action Representations

Maximilian Panzner and Philipp Cimiano

Semantic Computing Group, CITEC, Bielefeld University,
mpanzner@cit-ec.uni-bielefeld.de,
https://cit-ec.de/

Abstract. In this paper we are concerned with learning models of actions and compare a purely generative model based on Hidden Markov Models to a discriminatively trained recurrent LSTM network in terms of their properties and their suitability to learn and represent models of actions. Specifically we compare the performance of the two models regarding the overall classification accuracy, the amount of training sequences required and how early in the progression of a sequence they are able to correctly classify the corresponding sequence. We show that, despite the current trend towards (deep) neural networks, traditional graphical model approaches are still beneficial under conditions where only few data points or limited computing power is available.

Keywords: HMM, LSTM, incremental learning, recurrent network, action classification

1 Introduction

Representing and incrementally learning actions is crucial for a number of scenarios and tasks in the field of intelligent systems where agents (both embodied and unembodied) need to reason about their own and observed actions. Thus, developing suitable representations of actions is an important research direction. We are concerned with how to develop action models that allow intelligent systems to reason about observed actions in which some trajector is moved relative to some reference object. Such actions correspond to manual actions of humans that manipulate objects. We are concerned with finding models that support tasks such as: classifying observed actions according to their types, forward prediction and completion of actions. A particular focus lies on understanding how to incrementally learn manipulation actions from only few examples and how to enable models to classify actions which are still in progress. We compare two broad classes of models: generative sequence models (HMMs in particular) and discriminatively trained recurrent neural networks (LSTM recurrent networks in particular). We analyze their properties on the task of classifying action sequences in terms of how much training examples are required to make useful predictions and how early in a sequence they are able to correctly classify the respective sequence. Our models are trained on a dataset consisting of 1200 examples of action performances of human test subjects for four basic actions: *jumps over*, *jumps upon*, *circles around* and *pushes*. Our results are encouraging and show that action models can be successfully learned with both types of models, whereby the main advantage of HMMs is that they can learn with fewer examples and they show very favorable training times when compared to LSTMs (10 seconds for HMMs vs. 4 hours for LSTMs) while displaying comparable performance of about 86% F-measure. Future work will involve scaling these models to more complex action types and model actions described in terms of richer features including sensor data.

2 Related Work

There has been a lot of work in the field of intelligent systems on developing formalisms for learning and representing actions, ranging from task-space representations [1] to high-level symbolic description of actions and their effects on objects [2]. Many approaches work directly on raw video streams, which is a challenging problem due to high variance in the video emanating from e.g. different lighting conditions and also due to high intra-class variance in the actions. These problems are often tackled by first finding robustly recognizable interest points to extract and classify features from their spatio-temporal neighborhood. Panzner et. al. [3] detect points of interest using a Harris corner detector extended to 3D spatio-temporal volumes. They extract HOG3D/HOF features from the spatio-temporal neighborhood of these points which are then clustered using a variant of growing neural gas (GNG) to yield an incremental vocabulary of visual words. They represent action sequences as frequency histograms over these visual words (bag of words) which are then classified by another GNG layer. Veeriah et. al. [4] use densely sampled HOG3D features which are directly classified by an extended LSTM network which considers the spatio-temporal dynamics of salient motions patterns. They use the Derivative of States $\frac{\partial s_t}{\partial t}$, where s_t is the state of memory cell s a time t, to gate the information flow in and out of the memory cell. In this paper we focus on mid-level representations to bridge the gap between low-level representations and high-level symbolic descriptions in a way that facilitates transfer of learned concepts between the representational levels.

3 Models

We compare two approaches to modeling sequences of positional relation between a trajector and a landmark. The first approach models the relation between the two objects as class-specific generative Hidden Markov Models[5]. The second approach utilizes a discriminative two-layer neural network with a LSTM (long-short term memory) recurrent network as the first layer followed by a fully connected linear output layer with softmax normalization to interpret the network output as class probabilities. We are particularly interested in how fast the models converge when there is only little training data and how early in a sequence they are able to correctly classify the respective sequence. As representation learning generally requires rather large training sets, we abstract away from the raw positions and velocities by discretizing them into qualitative basic relations between trajector and landmark following the QTC (qualitative trajectory calculus) [6] framework. This abstraction is especially beneficial for the HMM approach because the markov assumption is better satisfied when we encode position and velocity in a joint representation. LSTM networks are usually considered to work best when the input space is continuous but in this case preliminary experiments showed that the QTC discretization is also favorable for LSTM in terms of classification performance and training time. When the dataset is large enough it should also be possible to train both models on raw features like coordinates relative to a trajectory-intrinsic reference point as shown by Sugiura et. al. [1] for HMMs.

3.1 Qualitative trajectory representation

To describe the relative position and movement between landmark and trajector we build on the qualitative trajectory calculus - double cross (QTC_{C1}) [6] as a formal foundation. In general, QTC describes the interaction between two moving point objects k and l with respect to the reference line RL that connects them at a specific point t in time. The QTC framework defines 4 different subtypes as a combination over different basic relations between the two objects. As we only have one actively moved object in our experiments, we decided on QTC_{C1} to give the best trade off between generalization and specificity of the qualitative relations. QTC_{C1} consists of a 4-element state descriptor (C_1, C_2, C_3, C_4) where each $C_i \in \{-, 0, +\}$ represents a so called constraint with the following interpretation:

C_1 Distance constraint: Movement of k with respect to l at time t_1:
- - k is moving towards l
- 0 k is not moving relative to l
- + k is moving away from l

C_2 Distance constraint: Movement of l with respect to k at time t_1: analogously to C_1

C_3 Side constraint: Movement of k with respect to RL at time t_1:
- - k is moving to the left-hand side of RL
- 0 k is moving along RL or not moving at all
- + k is moving to the right-hand side of RL

C_4 Side constraint: Movement of l with respect to RL at time t_1: analogously to C_3

As the positions in our dataset were sampled at a fixed rate, we could have missed some situations where one or more state descriptor elements transition through 0. These discretization artifacts are compensated by inserting the missing intermediate relations one at a time from left to right. QTC_{C1} is a rather coarse discretization, leading to situations where the qualitative relation between the two objects can hold for a longer portion of the trajectory and is, due to the fixed rate sampling, repeated many times. Unlike many spatial reasoning systems, where repeating states are simply omitted, we use a logarithmic compression of repetitive subsequences:

$$|\hat{s}| = \min(|s|, 10\ln(|s|+1)) \tag{1}$$

where $|s|$ is the original number of repeated symbols in the sequence and $|\hat{s}|$ is the new number of repeated symbols. By applying this compression scheme, we preserve information about the acceleration along the trajectory, which increases the overall performance especially for very similar actions like *jumps over* and *jumps upon*, while still allowing to generalize over high variations in relative pace of the action performances. The logarithmic compression of repetitive symbols in a sequence is in line with findings from psychophysics known as the Weber-Fechner law [7].

3.2 HMM

For the HMM parameter estimation, we apply an incremental learning scheme utilizing the best first model merging framework [8,9]. Model merging is inspired by the observation that, when faced with new situations, humans and animals alike drive their learning process by first storing individual examples (memory based learning) when few data points are available and gradually switching to a parametric learning scheme to allow for better generalization as more and more data becomes available [10]. Our approach mimics this behavior by starting with simple models with just one underlying sequence, which evolve into more complex models generalizing over a variety of different sequences as more data is integrated. Learning a new sequence in this framework is realized by first constructing a maximum likelihood (ML) Markov Chain which exactly reproduces the respective sequence, which is then integrated between the start and the end state of the existing (possibly empty) model. When incorporating more and more sequences, the resulting models would constantly grow and consist only of ML chains connected to the start and end states of the model. To yield more compact models, which are able to generalize to similar but unseen sequences, we consecutively merge similar states and thus intertwine their corresponding paths through the model. This way learning as generalization over the concrete observed examples is driven by structure merging in the model in a way that we trade model likelihood against a bias towards simpler models. This is known as the Occam's Razor principle, which among equally well predicting hypothesis prefers the simplest explanation requiring the fewest assumptions. As graphical models, HMMs are particularly well suited for a model merging approach because integrating a new sequence, merging similar states and evaluating the model's likelihood given the constituting dataset are straightforward to apply in this framework and implemented as graph manipulation operations:
Data integration: When a new sequence is to be integrated into a given model

we construct a unique path between the initial and the final state of the model where each symbol in the sequence corresponds to a fresh state in the new path. Each of these states emits its respective symbol in the underlying sequence and simply transitions to the next state with probability 1, yielding a sub path in the model which exactly reproduces the corresponding sequence.
State merging: The conversion of the memory based learning scheme with unique sub paths for each sequence in the underlying dataset into a model which is able to generalize to a variety of similar trajectories is achieved by merging states which are similar according to their emission and transition densities. Merging two states q_1 and q_2 means replacing these states with a new state $\hat{q}$ whose transition and emission densities are a weighted mixture of the densities of the two underlying states.
Model evaluation: We evaluate the models resulting in the merging process using a mixture composed of a structural model prior $P(M)$ and the data dependent model likelihood $P(X|M)$:

$$P(M|X) = \lambda P(M) + (1 - \lambda) P(X|M) \tag{2}$$

The model prior $P(M)$ acts as a data independent bias. Giving precedence to simpler models with fewer states makes this prior the primary driving force in the generalization process:

$$P(M) = e^{-|M|}, \tag{3}$$

where the model size $|M|$ is the number of states in the model. It is also possible to include the complexity of the transitions and emissions per state. For our dataset we found that using only the number of states generates the best performing models. While the structural prior favors simpler models, its antagonist, the model likelihood, has its maximum at the initial model with the maximum likelihood sub-paths. The exact likelihood of the dataset X given the model M is computed as:

$$P(X|M) = \prod_{x \in X} P(x|M) \tag{4}$$

with

$$P(x|M) = \sum_{q_1 \ldots q_l \in Q^l} p(q_I \to q_1) p(q_1 \uparrow x_1) \ldots\ p(q_l \uparrow x_l) p(q_l \to q_F) \tag{5}$$

where l is the length of the sample and q_I, q_F denote the initial and final states of the model. The probability to transition from a state q_1 to q_2 is given as $p(q_1 \to q_2)$ and $p(q_1 \uparrow x_1)$ denotes the probability to emit the symbol x_1 while being in state q_1. As we do not want to store the underlying samples explicitly, we use an approximation, which considers only the terms with the highest contribution, the Viterbi path:

$$P(X|M) \approx \prod_{q \in Q} \left(\prod_{q\prime \in Q} p(q \to q\prime)^{c(q \to q\prime)} \prod_{\sigma \in \Sigma} p(q \uparrow \sigma)^{c(q \uparrow \sigma)} \right) \tag{6}$$

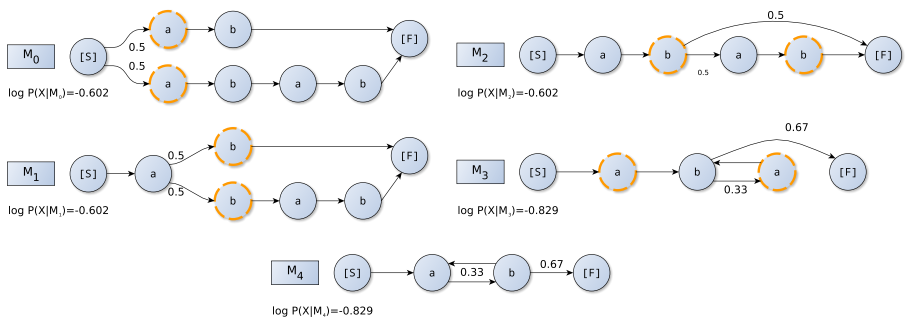


Fig. 1. Sequence of models obtained by merging samples from an exemplary language $(ab)^+$ and subsequently merging the highlighted states. Transitions without special annotations and all emissions have a probability of 1. The highlighted nodes are selected by the similarity of their transition and emission characteristics and consecutively merged to yield the subsequent model. Reproduced from [8].

where $c(q \to q\prime)$ and $c(q \uparrow \sigma)$ are the total counts of transitions and emissions occurring along the Viterbi path associated with the samples in the underlying dataset (see [8] for more details).

The simplest model in our approach is a model which simply produces a single sequence. These models are called maximum likelihood models because they produce their respective sequences with the highest possible probability. Starting from maximum likelihood models over individual sequences we build more general HMMs by merging simpler models and iteratively joining similar states to intertwine sub-paths constructed from different sequences, allowing them to generalize across different instances of the same action class. The first model M_0 of the example in Figure 1 can be seen as a joint model of two maximum likelihood sequences $\{ab, abab\}$. When generating from such a model, the actual sequence which will be generated is determined early by taking one of the possible paths emanating from the start state. Only the transitions from the start state display stochastic behavior, the individual sub-paths are completely deterministic and generate either ab or $abab$. Intertwining these paths is done trough state merging, where we first build a list of possible merge candidates using a measure of similarity between state emissions and transition probability densities. In this approach we use the symmetrized Kullback-Leibler (KL) divergence

$$D_{\mathrm{SKL}}(P, Q) = D_{\mathrm{KL}}(P, Q) + D_{\mathrm{KL}}(Q, P). \tag{7}$$

with

$$D_{\mathrm{KL}}(P, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{8}$$

Then we greedily merge the best pair of states and re-evaluate the model likelihood. In the example above, the first two merges lead to model M_3 where we experienced a drop in log likelihood from -0.602 to -0.829. We continue the merging process until we reach a point where merging more states would deteriorate the model likelihood to a point where it is no longer compensated by the prior favoring simpler models (Equation 3). The final model M_4 is now able to generate the whole set of sequences from the exemplary language $(ab)^+$ from which the two initial samples where generated from.

As neighboring QTC state IDs have no semantic similarity, we model emission densities as frequency distributions. We train a separate HMM for each of the four action classes. Sequences are classified according to the class specific HMM having the highest probability to have produced the respective sequence. When none of the HMMs assigns a probability higher than zero for a sequence, the model is unable to classify this sequence and assigns no class label (reject option). Training with 1100 sequences takes about 10 seconds on a 2 CPU – 20 core machine. See [11] for a more detailed analysis of the properties of this model.

3.3 LSTM

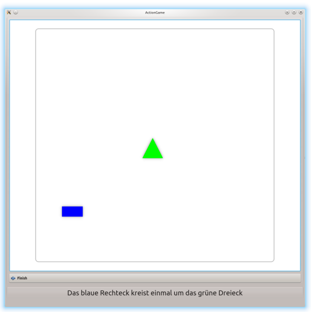
The discriminative recurrent network consists of a long-short term memory (LSTM) layer with 128 memory blocks followed by a fully connected linear layer with softmax normalization and one output unit per target class. The softmax normalization allows the network output to be interpreted as class probabilities. Unlike the HMM approach where we train a separate HMM for each class, this network learns a joint representation between the sequences of all 4 classes. Several variants of the block architecture of LSTM memory cells have been proposed [12,13]. Our implementation is based on the architecture proposed by Hochreiter and Schmidhuber [14]. We optimize the cross entropy

$$H(p,q) = -\sum_{x} p(x)\log(q(x)) \tag{9}$$

between the predicted p and target classes q using RMSprop [15] with learning rate $lr = 0.001$ and decay factor $\rho = 0.9$ over batches of 50 sequences. We randomly select 10 percent of the training data per class as validation set to asses the performance of the model for each training epoch. The model with the highest accuracy on the validation set is used to perform the experiments. Other than the model selection we apply no other regularization technique such as dropout, because the models showed no tendency to overfit. The network layout was optimized in a separate experiment to give the best trade-off between classification performance and training time. Training with 1100 sequences over 150 epochs takes about 4 hours on a single GTX-770 GPU.

4 Dataset

To acquire a dataset we implemented a simple game (Figure 2) where the test subjects were asked to perform an action with two objects according to a given instruction. The game screen was divided into two parts. The upper part was the actual gamefield with the two freely movable objects and below the gamefield was a textfield, where the test subjects could see the instruction describing the desired action performance. We had a total of 12 test subjects (9 male, 3 female, mean age = 29,4 years) yielding a dataset with 1200 trajectory sequences balanced over the four action classes *jumps over*, *jumps upon*, *circles around* and *pushes*. The recorded trajectories are given as tuples with the current timestamp and the positions of trajector and landmark. The raw positions are then converted to QTC state sequences with an average length of 182 symbols. The compression scheme (Equation 1) reduces the average length to 150 symbols which corresponds to a compression rate of $21,\overline{3}\%$. See [16] for a complete description and download [17] of the dataset.



Das blaue Rechteck kreist einmal um das grüne Dreieck

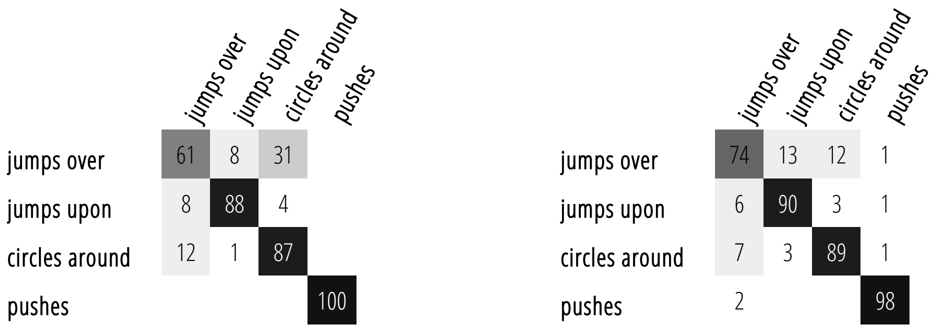
Fig. 2. Left: Simple game with two geometric objects which can be freely moved on the gamefield. In this screen test subjects are tasked to circle the blue rectangle around the green triangle (instruction in the lower part of the screen).

5 Experiments

In this section we evaluate the overall performance of both models against each other and specifically explore their performance regarding their ability to make useful predictions when there is only little training data per class. We also explore how early in the progression of a sequence the models are able to correctly classify the respective sequence. Both models receive sequences of QTC state IDs. The IDs are simply derived by interpreting the QTC descriptors as a number in a ternary number system. For the LSTM network state IDs are normalized to 1 to better fit the range of the activation functions. We also tried a 4 element vector representation of the QTC state descriptors as input for the LSTM Network, which slightly reduced the number of epochs the training needed to converge but left the overall accuracy unchanged. Thus, for the sake of comparability we decided to use the same input data representation for both approaches.

5.1 HMM vs LSTM Network

In this experiment we compare the performance of the HMM approach to the performance of the LSTM network. The dataset was divided such, that we trained on data collected from 11 test subjects and let the models classify sequences



	jumps over	jumps upon	circles around	pushes
jumps over	61	8	31	
jumps upon	8	88	4	
circles around	12	1	87	
pushes				100

	jumps over	jumps upon	circles around	pushes
jumps over	74	13	12	1
jumps upon	6	90	3	1
circles around	7	3	89	1
pushes	2			98

Fig. 3. Averaged class confusion matrix for the HMM (left) and the LSTM network (right).

from a 12-th test subject as a novel performer. As can be seen in Table 1 both approaches perform comparably well. The LSTM network had a considerably higher standard deviation between the results of the 12 folds. We assume that the LSTM network gives higher precedence to differences in relative pace of the action performance, which is the most varying factor in our dataset. Only the HMM approach benefits from compressing long repetitive subsequences, while the LSTM approach is unaffected. Note that only the HMM approach had a reject option, when none of the class specific HMMs assigned a probability higher than zero to the respective sequence. As the LSTM approach learns a joint representation across all 4 classes there is no implicit reject option, because the model will always assign softmax normalized confidences to each class. Figure 3 shows the class confusion matrices for both approaches. The HMM approach frequently misclassifies *jumps over* sequences as sequences corresponding to *circles around* actions, for which the first half of the movement is identical to *jumps over*. This could be caused by the limitation of temporal context HMMs are able to take into consideration due to the markov assumption.

Model	Compressed				Not compressed			
	F_1	P	R	σ	F_1	P	R	σ
HMM	0.86	0.82	0.90	0.12	0.82	0.75	0.91	0.10
LSTM	0.88	0.88	0.88	0.18	0.88	0.88	0.88	0.18

Table 1. Results of the 12-fold cross-validation for the HMM and the LSTM approach with and without sequence compression (Equation 1). Results are given as F_1 score, precision, recall and standard deviation σ.

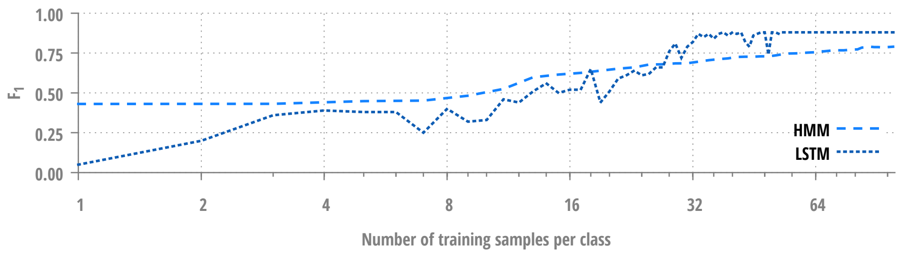


Fig. 4. Early learning behavior, averaged over 10 random folds. HMMs outperform the LSTM network where very little training data is available, while the latter performs better with more than 28 training sequences. Note that the x-axis is scaled logarithmically.

5.2 Early learning behavior

In this experiment we evaluate the performance of the two models when they are presented with only few training sequences per class. Both networks are not trained from scratch when more training sequences are added to the training set, instead both are incrementally re-trained on the extended training set. Figure 4 shows that with an F_1 score of 0.44 the HMMs perform notably better than the LSTM network which starts at chance level after being trained on a single sequence per class. When trained with more than 23 sequences, the LSTM outperforms the HMMs clearly.

5.3 Early Sequence Classification

Applications in the field of human robot interaction often require the robot to produce action hypothesis when the respective action sequence is not yet completed. This experiment simulates this condition by letting the models classify incomplete sequences. We evaluate the classification performance with sequences truncated to 1 to 100 percent of their original length. The LSTM network has been trained for only 100 epochs to keep it from striving too much towards relying on late parts of the sequences. Apart from that both approaches were not specifically trained for the classification of truncated sequences. Because we trained one HMM per class they were quicker in capturing the essence of the *pushes* action, which was the only action where both objects moved. LSTM performed better on *jumps upon*, supporting the suspicion that LSTM assigns more weight on the pace of action performances, which is also the case for the *circles around* action. Because *jumps over* is the first part of an *circles around* action, both models need about half of the trajectory to separate these actions.

6 Conclusion

Our results show that HMMs can be used competitively on action modeling tasks compared to discriminatively trained neural networks in scenarios where train-

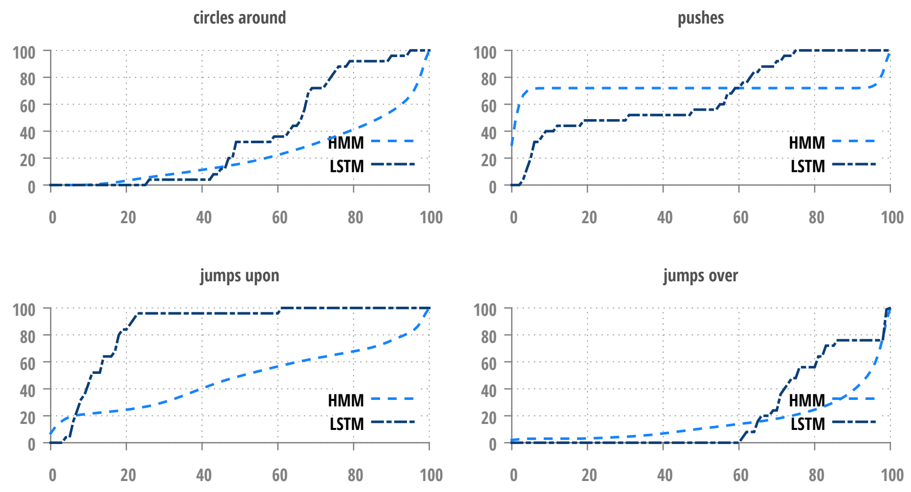


Fig. 5. Early sequence classification behavior of the HMM and LSTM network for each of the four action categories. The x-axis represents the percentage of the sequence presented to the classifier and the y-axis the percentage of correctly classified sequences for which their classification result will not change if more of the sequence was presented (stable classification results).

ing data is limited. In particular, our experiments show that the classification performance of HMMs is already useful when training with only a single example per class, consuming only a few milliseconds of CPU time. This is especially useful for example when a robot has to quickly adapt to a human interaction partner. The LSTM network required about 23 examples to reach the classification performance of a class dependent HMM and took a minimum of 50 seconds training time to converge which is too long for fluent interaction. When classifying incomplete actions, both models had their strength and weaknesses. The HMMs were better in capturing strong discriminative features, as in the *pushes* action where two objects are moving, while LSTM displayed a slightly better overall performance and was able to make decisions earlier throughout all four action categories. For our targeted field of application it could be beneficial to use both models in parallel. HMM for early learning (short-term action memory) or when quick reaction towards new stimuli is required and LSTM when enough data is available (long-term). In the future we would also like to explore whether deep neural networks can learn interpretable qualitative relations that are comparable to the QTC relations we used for this evaluation.

This research/work was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

References

1. Komei Sugiura, Naoto Iwahashi, Hideki Kashioka, and Satoshi Nakamura. Learning, generation and recognition of motions by reference-point-dependent probabilistic models. *Advanced Robotics*, 25(6-7):825–848, 2011.
2. Moritz Tenorth and Michael Beetz. KnowRob – Knowledge Processing for Autonomous Personal Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4261–4266, 2009.
3. Maximilian Panzner, Oliver Beyer, and Philipp Cimiano. Human Activity Classification with Online Growing Neural Gas. In *Workshop on New Challenges in Neural Computation (NC2)*, 2013.
4. Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. Differential recurrent neural networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4041–4049, 2015.
5. Lawrence Rabiner and B Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.
6. Nico Weghe, Bart Kuijpers, Peter Bogaert, and Philippe Maeyer. A Qualitative Trajectory Calculus and the Composition of Its Relations. *GeoSpatial Semantics SE - 5*, 3799(Dc):60–76, 2005.
7. Thomas Bruss and Ludger Rüschendorf. On the perception of time. *Gerontology*, 56(4):361–370, 2010.
8. Stephen Omohundro. Best-first model merging for dynamic learning and recognition. In *Advances in Neural Information Processing Systems 4*, pages 958–965. Morgan Kaufmann, 1992.
9. Andreas Stolcke and Stephen Omohundro. Inducing probabilistic grammars by bayesian model merging. In *Grammatical inference and applications*, pages 106–118. Springer, 1994.
10. Roger N Shepard. Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323, 1987.
11. Maximilian Panzner and Philipp Cimiano. Incremental learning of action models as HMMs over qualitative trajectory representations. Workshop on New Challenges in Neural Computation (NC2) , 2015.
12. Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.
13. Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.
14. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
15. Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4:2, 2012.
16. Maximilian Panzner, Judith Gaspers, and Philipp Cimiano. Learning linguistic constructions grounded in qualitative action models. IEEE International Symposium on Robot and Human Interactive Communication, 2015.
17. Maximilian Panzner. TLS Dataset (doi:10.4119/unibi/2904362), 2016.