Combining Neural Density Estimation and Computational State Space Models for Human Activity Recognition

Master Thesis

presented by
Benedikt Jakob Lasotta
Matriculation Number 1610104

submitted to the
Institute for Enterprise Systems
Dr. Stefan Lüdtke
University of Mannheim

September 2022

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

The task of recognizing human activities from sensor data is relevant for many real world applications. Its uses range from optimizing logistics in warehouses to providing healthcare and assistance systems for elderly and disabled people [9, 4]. Consequently, human activity recognition (HAR) has been an active topic of research in recent years. Historically, methods to perform sensor-based HAR, broadly fall into either of two categories, each with their respective strengths and weaknesses: Symbolic methods as proposed by Krüger et al. or Ramirez and Geffner [8, 14] can employ prior domain knowledge and thus may need less training data, but often lack the capacity to learn from complex, time-series sensor data. One such approach are computational state space models (CSSMs) [8], which employ probabilistic inference on top of symbolic, rule based representations of possible actions within a domain to reason about observed activities. Contrary to this, modern, data-driven methods are capable of handling complex, multi-channel sensor data and have shown promising results in the past. An approach using a convolutional neural network (CNN) is described by Moya Rueda et al. [11]. In recent literature Transformers have been applied to the task of recognizing human activities. Such an approach is described by Shavit and Klein [16], which uses an encoder-only Transformer architecture to aggregate sensor data from the entire time series via the attention mechanism. Their results indicate that such a system improves prediction accuracy over models based on CNNs. Next to this, many other deep learning methods for sensor-based HAR exist. A comprehensive survey over current deep learning HAR methods and challenges is given by Chen et al. [3]. Among the drawbacks of purely data-driven methods is the fact that they are oblivious to the causal structure of activities in the domain and thus may require more training data which is often hard to obtain. Thus recently, attempts have been made to integrate symbolic and data driven models. In the line of this work such approaches are referred to as hybrid models. The goal of these is to combine both approaches in order to obtain state of the art performance while being able to incorporate prior knowledge to reduce the need for training data. This work is not the first to propose such a hybrid model, three other notable systems are listed below: DeepProbLog [10], extends probabilistic-logic modeling with neural networks. The system proposed by Rueda et al. [15], employs a CNN to learn from sensor data and uses the output of this deep model to infer the most probable activity class via Bayesian filtering. DUSTIN [1] is a hybrid system which concatenates features obtained by a knowledge based reasoner to features extracted via a neural network. In this way information about the causal structure of activities can be captured. A more thorough overview of related work is given in chapter 3.

To benefit from the advantages of symbolic and data-driven methods, the main goal of this thesis is to develop a hybrid system for sensor-based HAR by combining existing components in a novel way. The aim of this hybrid system is to achieve a greater sample efficiency than current state of the art systems. That is, with a fixed amount of limited training data the hybrid system should outperform both data-driven and purely symbolic approaches. To build this hybrid model an implementation of computational state space models (CSSM) [8] referred to as computational causal behavior models (CCBM) will be combined with an observation model that is obtained from sensor data via neural density estimation methods. Initial experiments will be conducted with a type of normalizing flows called masked autoregressive flow (MAF) [12] since it has shown promising performance on similar density estimation tasks [7]. The system will be evaluated on a data set of raw sensor data from inertial measurement units (IMUs) of people performing the task of cooking a carrot soup, eating at, and cleaning up afterwards. It is available for download from the University of Rostock [**?**].

Additionally, an important research goal of this thesis is to examine which component contributes in what part to the final performance. To this end, an ablation study will be conducted. For this, the performance of the symbolic approach with a simple observation model, the density estimation component with a simple prediction, the full hybrid model, and a baseline of quadratic discriminant analysis (QDA) will be compared on the data set introduced above. The results of this ablation study can be found in chapter (insert ref).

Chapter 2

Theoretical Framework

2.1 Introduction to Computational State Space Models

The following paragraphs will introduce some preliminary knowledge and terminology that is required to follow the remainder of this thesis. Following convention, vector-valued variables will be written bold. Random variables are indicated by capital letters, while assignments to those random variables are indicated by non-capital letters.

The goal of human activity recognition is to recognize activities in a sequence of potential actions that a human could perform. As such the HAR context is always a dynamic one, i.e. entities within this system are subject to changing states. Starting from some initial state and following a transition model the system state changes as time progresses. Due to combinatorial explosion the space of potential states is very large. Thus, a system which allows tractable inference in dynamic systems is required. In a feasibility study by Krüger et al. [8], Computational State Space models (CSSMs) have been shown to fulfill this property, even in complex, real-world applications. The Authors found that the domain model used in their evaluation of the carrots data set had circa 10^8 states. Yet, the system was able to outperform Hidden Markov Models (HMMs) if the correct inference procedures were chosen. This section introduces the necessary background and notation to understand such models.

In CSSMs the transition model of the dynamic system is described by a computable function, which differentiates it from systems that require the explicit enumeration of states or paths, like HMMs. The behavior of the dynamic system is characterized as a labeled transition system (LTS). It consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, and labeled transition relations $\rightarrow \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ that describe transitions between states. Thus, an LTS is a triple of the form $(\mathcal{S}, \mathcal{A}, \rightarrow)$. It is possible that $\mathcal{S}$ and $\rightarrow$ become infinite while $\mathcal{A}$ is finite. As such it is necessary to find a suitable computational description to avoid explicit enumeration of all (potentially infinitely many) states. This description is then called the *computational action language* and is used to define the transition model. It is especially applicable when the modeled process (for example the act of cooking) can be interpreted as performing sequential computations to arrive at some goal state starting from an initial state. That is, when the observed behavior is somehow goal directed.

In general the model should allow tractable inference in the sense that the hidden state of the LTS can be inferred at time t given a sequence of past observations $Y_{1:t}$ from a space $\mathcal{Y}$ and a sequence of states $X_{1:t}$ from space $\mathcal{X}$. If the joint distribution $p(x_{1:t}, y_{1:t})$ factorizes over time into a *transition model* $p(X_t|X_{t-1})$ and an *observation model* $p(Y_t|X_t)$ such that

$$p(x_{1:t}, y_{1:t}) = p(y_1|x_1)p(x_1) \prod_{i=2}^{t} (p(y_i|x_i)p(x_i|x_{i-1})), \tag{2.1}$$

the joint distribution can be described by a state space model. The underlying idea of this thesis is to obtain the observation model $p(y_t|x_t)$ via a neural density estimator like masked autoregressive flow (MAF) as outlined in section 2.2, and then apply an implementation of CSSM known as CCBM to this observation data.

CCBM follows a symbolic approach that uses precondition-effect rules to define the causal model of the domain it operates on. The computational action language used in CCBM is an extension of the planning domain definition language (PDDL). This makes it possible to specify the causal structure of a domain with its actions, preconditions and effects. The observation model describes which expected observation y_t is caused by any given state x_t at time t, while the transition model describes how states change over time. In their feasibility study on CSSMs [8], Krüger et al. show that the choice of observation model has the largest impact on model accuracy. As a limitation of their work they note, that the sensor model used by them is fairly primitive: Each activity class is represented by a multivariate normal distribution with unconstrained covariance. Improving upon this basic model is thus a logical step for further research and is exactly the approach that this thesis takes. With MAF a powerful model is trained on the sensor data which aims to fit the distribution of the observation data more accurately. Thus, it enables the computation of $p(y_t|x_t)$. The data used to train this model is continuous raw sensor data. A thorough introduction to these datasets can be found in section (insert ref once available)**??**. Due to the limited number of actions and components in the State space model, the state space of the examined domain is discrete categorical.

The rules that govern an action, i.e., what are the parameters of an action, which preconditions have to be fulfilled, what is the effect and duration of the action, are defined for every possible action in a so called action template which is written in the action language. Given all action templates, an initial state distribution and the goal(s), a directed graph from the initial to the goal state(s) can be generated. In particular, given some state x the probability of selecting action a in CCBM is defined by a Log-Linear model with features f_1, f_2, f_3 in the following way [15]:

$$p(a|x) \propto \exp(\sum_{k=1}^{3} \lambda_k f_k(a, x)) \tag{2.2}$$

$$f_1(a, x) = \log \gamma(a(x)) \tag{2.3}$$

$$f_2(a, x) = \log s(a) \tag{2.4}$$

$$f_3(a, x) = \delta(a(x)) \tag{2.5}$$

Each of the features can be weighted by a scalar λ_k. The first feature $\gamma(a(x))$ is the revisiting factor, which is 0 if the state resulting from taking action a in state x has already been visited. It controls if already visited states can be visited again. The parameter $s(a)$ is the saliency of action a which is a weight that can be assigned to any action. A higher saliency indicates the action is more likely to be selected with respect to actions with lower saliency. Lastly, $\delta(a(x))$ is the goal distance of the state x' that results from performing action a in state x. The underlying assumption is, that actions which lead to states that are closer to the goal are selected with a higher probability.

The CCBM system is used to compile a filter from which it is possible to perform inference about the most probable action given a system sate. Due to the large state space exact inference is computationally infeasible. Hence, particle and marginal filters are used to perform approximate inference. In categorical domains the marginal filter has been shown to outperform the particle filter [8], which is the reason why it is used in this work. Standard Bayesian filtering methods are then employed to reason about state and action sequences from sensor data. To do this, first the prediction of the current state x_t is calculated based on the previous state x_{t-1} and transition probabilities, which are obtained by plugging in equation 2.2 into equation 2.6 below.

$$p(x_t|x_{t-1}) = \sum_{x_t=a(x_{t-1})} p(a|x_t) \tag{2.6}$$

$$p(x_t|y_{1:t-1}) = \sum_{x_{t-1}} p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1}) \tag{2.7}$$

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \tag{2.8}$$

The transition probabilities $p(x_t|x_{t-1})$ are then used for the actual *prediction* step of Bayesian filtering in equation 2.7, producing an estimated state. Using the observation model $p(y_t|x_t)$, the *correction* step then corrects this estimated state by taking into account the most recent observation as shown in equation 2.8.

2.2 Distribution Estimation with Masked Autoregressive Flow

The task of estimating the joint probability distribution $p(\boldsymbol{X})$ from a set of examples $\{\boldsymbol{X}_n\}_{n=1}^N$ is a central aspect in many machine learning applications. In the line of this work it is of great interest to obtain density estimates for the sensor data as this can be used for the observation model. This approach allows greater flexibility than using a simple multivariate normal distribution for all actions of a given class, which was used in the original feasibility study on CSSMs [8].

Let $\boldsymbol{x}$ be an example consisting of D random variables $X_1, X_2, ..., X_D$. The chain rule of probability states that any joint distribution of variables can be decomposed into a product of joint conditional probabilities, where variable X_d only depends on the prior variables $X_{1:d-1}$. According to this rule, the joint distribution $p(\boldsymbol{x})$ can be expressed as

$$p(\boldsymbol{x}) = \prod_{d=1}^{D} p(x_d|x_{1:d-1}). \tag{2.9}$$

This property can be exploited in order to estimate the joint distribution. To perform this task, many methods exist. In the line of this work the focus will be on methods based feed-forward neural networks which fulfill certain properties that allow correct estimation of probability densities. In particular this section will briefly introduce masked autoregressive flow (MAF) [12] as a model of performing neural density estimation from a set of examples. MAF can be thought of as stacking multiple layers of autoregressive models on top of each other to obtain greater modeling flexibility. In the case of this work each individual MAF layer is a masked autoencoder for distribution estimation (MADE) [6] proposed by Germain et al. in 2015. A MADE is an autoencoder, that is a feed forward neural network which takes as input some D-dimensional vector $\boldsymbol{x}$ and learns a compressed hidden representation $h(\boldsymbol{x})$ from which a reconstruction $\hat{\boldsymbol{x}}$ of input $\boldsymbol{x}$ is obtained. It is possible to use an autoencoder for density estimation if each partial output $\hat{x}_d$ only depends on the previous variables x_1 to x_{d-1} and computes the probability of x_d given these previous variables. Note, that these conditionals are the factors of the joint distribution in formula 2.9 and thus their product gives the joint distribution $p(\boldsymbol{x})$. If each output unit only depends on the previous variables the model fulfills the so called *autoregressive property*. To ensure that a MADE has this property, binary masking matrices are multiplied element wise to the encoder and decoder weight matrices of the autoencoder. These masking matrices have the value 1 where connections should be retained and 0 where they should be dropped. This guarantees that variables X_1 to X_{d-1} are connected to the output $\hat{X}_d$, while the remaining variables X_d to X_D are dropped. The MADE paper [6] proposes a straightforward but effective procedure to perform this masking which is explained below:

To illustrate the principle of a masked autoencoder which fulfills the autoregressive property we look at a simple example which starts from a fully connected autoencoder with one hidden layer consisting of K hidden units. This standard autoencoder computes the hidden representation $h(\boldsymbol{x})$ of an input $\boldsymbol{x}$ and from the hidden representation computes the reconstruction $\hat{\boldsymbol{x}}$ of. Let $W \in \mathbb{R}^{K \times D}$ be the input weight matrix for the single hidden layer and $V \in \mathbb{R}^{D \times K}$ be the output weight matrix of that layer. These two matrices parameterize the autoencoder. As elaborated above, a masked autoregressive autoencoder can be constructed from this fully connected network by calculating the binary masking matrices $\mathbf{M}^W$, $\mathbf{M}^V$ and multiplying them element wise to the weight matrices. In the network this corresponds to dropping connections which would introduce information from subsequent variables. Calculating the masking matrices is straightforward and only requires that each of the K hidden units is assigned a value $m(k)$ in the range from 1 to $D-1$. The value $m(k)$ can be viewed as the number of inputs the k^{th} hidden unit is connected to. Hence, $m(k) = 0$ and $m(k) = D$ are disallowed because they would create hidden units which are connected to none, respectively all input units. Both of these would not be useful for modeling conditionals. To assign these $m(k)$ values Germain et al. [6] suggest that for each hidden unit the value is sampled from a uniform discrete distribution defined on the integers from 1 to $D-1$. In expectation this means that each possible value of $m(k)$ will be assigned to an approximately equal number of hidden units. Following this, the masking matrices $\mathbf{M}^W$, $\mathbf{M}^V$ can be computed through the equations 2.10 and 2.11 respectively. For

$d \in \{1, ..., D\}$ and $k \in \{1, ..., K\}$:

$$\mathbf{M}^{W}_{k,d} = \begin{cases} 1 & \text{if } m(k) \geq d \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

$$\mathbf{M}^{V}_{d,k} = \begin{cases} 1 & \text{if } d > m(k) \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

Figure 2.1 illustrates this masking for a simple autoencoder with $D = 3$ and one hidden layer with $K = 5$ hidden units. However, the process can be extended to autoencoders with multiple hidden layers. Furthermore it is not necessary that the order of inputs is kept. In fact, Germain et al. suggest that order-agnostic training, in which multiple MADEs are trained on different random orderings, may be beneficial [6].

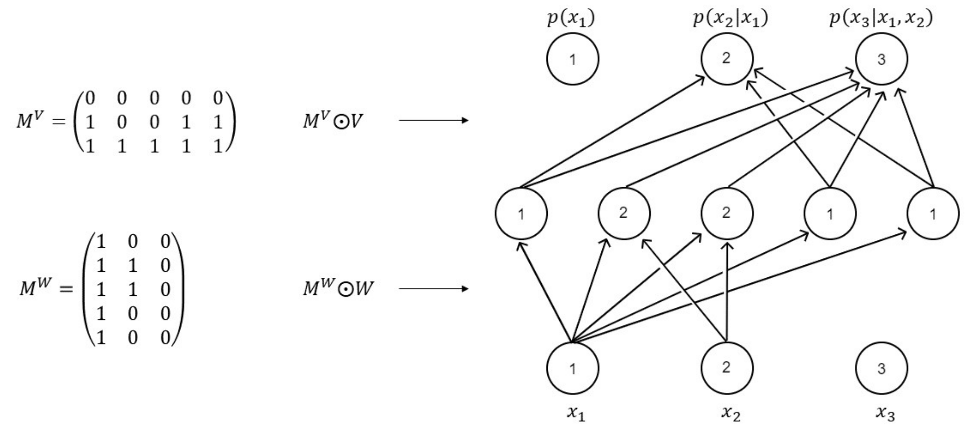
$M^V = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$
$M^V \odot V$
$M^W = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$
$M^W \odot W$
$p(x_1)$
$p(x_2|x_1)$
$p(x_3|x_1, x_2)$
1
2
3
1
2
2
1
1
1
2
3
x_1
x_2
x_3

Figure 2.1: Illustration of the effect of masking matrices on the graph structure of an autoencoder. The digits inside the hidden layer nodes are the $m(k)$ values that were used to calculate $\mathbf{M}^W$ and $\mathbf{M}^V$, the digits inside input / output nodes indicate the ordering. Connections where the masking matrix is 0 are dropped while others are retained (own figure).

A main disadvantage of MADE is that the order of the conditionals has a large impact on the quality of the learned density. With the correct order it may be possible to learn the density perfectly while for other orders this is not the case. To remedy this, MAF stacks L MADEs and models the random numbers which are used by the MADE in layer l with the model in layer $l + 1$ and so on, until the random numbers of the model in the last layer are modeled by a standard Gaussian.

Although each individual MADE unit has unimodal conditionals the authors of MAF argue that such a model can learn multimodal conditionals [12]. According to them, this added flexibility improves model fit.

The central idea behind MAF is, that the distribution to be learned is viewed as a transformation of a base density $\pi_u(\boldsymbol{u})$ to the target distribution via a differentiable and invertible transformation f. The base density should be simple in the sense that $\pi_u(\boldsymbol{u})$ can be easily evaluated for all inputs $\boldsymbol{u}$. For example the standard multivariate Gaussian where $\boldsymbol{u} \sim \mathcal{N}(0, I)$ is a common choice. This means that f takes an input $\boldsymbol{u}$, which follows the base density distribution $\boldsymbol{u} \sim \pi_u(\boldsymbol{u})$ and transforms it to data space via the invertible function $\boldsymbol{x} = f(\boldsymbol{u})$. As f is invertible it holds that $\boldsymbol{u} = f^{-1}(\boldsymbol{x})$, and the density $p(\boldsymbol{x})$ can be calculated as

$$p(\boldsymbol{x}) = \pi_u\big(f^{-1}(\boldsymbol{x})\big)\left|\det\left(\frac{\partial f^{-1}}{\partial \boldsymbol{x}}\right)\right|. \tag{2.12}$$

According to Papamakarios et al. [12] the computation of this expression is tractable by design. The first part is easy to compute because f is by design invertible and the base density can be evaluated for any input. The absolute determinant of the Jacobian is tractable, because the Jacobian matrix is triangular. This is due to the autoregressive property where the d-th variable x_d only depends on the prior $d-1$ variables $x_{1:d-1}$. Consequently, the Jacobian of f^{-1} is zero for all variables that this variable does not depend on. In each layer the mean and the log standard deviation of the d-th conditional given only the previous previous variables are computed via functions which are implemented as MADE units. Stacking multiple such units on top of each other is possible because if f_1 and f_2 are differentiable and invertible functions their composition $f_1 \circ f_2$ is also invertible and differentiable. This means the necessary properties to keep expression 2.12 tractable are maintained, even when stacking multiple layers. The use of masking makes it possible to compute the density $p(\boldsymbol{x})$ of data $\boldsymbol{x}$ in a single forward pass.

Chapter 3

Related Work

While it has been an active research topic for many years, the field of neurosymbolic AI has recently gained renewed traction after calls for more explainable and semantically sound AI systems have gotten louder [5]. This section gives a concise overview of recent developments in this field and introduces successful models related to the presented work. Intellectually closest to this thesis is the approach proposed by Rueda et al. [15]. They present the idea of using deep neural architectures as the observation model required by CSSMs. In particular the authors use a CNN to learn from time-series sensor data provided by inertial measurement units (IMUs), and predict the observed action class. In that sense the purpose of the system is very similar to the model presented here. The authors evaluated their model on the same carrots data set that will be used for evaluation in this thesis and found that the model performance was comparable to that of state of the art deep models at the time. In the line of this work, a similar approach will be taken. With the difference being, that the observation model is obtained by a neural density estimator like MAF. The benefit of this approach over using a CNN is that MAF actually learns a distribution and can compute the density of sensor data in a single forward pass. While a CNN computes as surrogate the predicted action from sensor data, MAF in fact allows computation of the observation model $p(y_t|x_t)$. Subsequently, this observation model is combined with CCBM to perform probabilistic inference. As the same dataset is used for evaluation in this thesis and the work introduced above, it will be interesting to compare the results with this approach.

Next to this, many other approaches to combine deep neural architectures with symbolic reasoning have been proposed in recent literature. DeepProbLog [10] is introduced as a framework which combines neural networks and probabilistic-logical modeling via the existing language ProbLog [13]. This is done by extending ProbLog with neural predicates. These special predicates essentially compute the probability of atomic expressions in a probabilistic logic via a neural network. The evaluation of DeepProbLog shows that this framework is capable of performing symbolic reasoning as well as deep learning from examples in an end-to-end fashion. The benefit of this system over previous work is, that it integrates neural networks into a probabilistic-logical framework instead of approximating reasoning with neural nets. Consequently, the resulting model is able to perform probabilistic-logical inference, deep learning and combinations thereof. The downside of this approach is that its inference algorithms and language are ill-suited for dynamic systems, such as HAR.

Another hybrid architecture for HAR in an order-picking scenario is proposed by Lüdtke et al. [9]. This model first predicts higher level movement descriptors from sensor data via a temporal convolutional neural network. These attributes are then used by a shallow classifier to estimate the most likely activity class. Additionally, the current process step can be taken into account as context information by making it accessible to the shallow classifier. One main benefit of this system is that it allows the integration of prior context information without re-training the entire network. Consequently, the deep model can be interpreted as a tool for feature extraction from sensor data. This means that it is largely domain independent and only the shallow prediction head has to be adjusted when switching domains. This model is reported to achieve state-of-the-art HAR performance even in the absence of context information, but performance increases further if relevant information from the process step is included, even if it is not always correct.

A very recent approach that shows promising results for HAR is DUSTIN [1] which employs knowledge infusion on top of features extracted via a neural network. That is, the model uses a CNN to extract features from time series sensor data as well as the high-level context and concatenates to this representation a set of features which are obtained by a knowledge based reasoner before its classification layer. Just as for CSSMs, this has the added benefit that common-sense knowledge can be introduced to the model, which means that estimated action sequences are consistent with the user-context. However, unlike most models DUSTIN allows infusion of additional knowledge into the deep model itself. That is instead of applying such information before or after the learning process, additional knowledge from a symbolic reasoner is added before the classification layer of the deep model. This is achieved by aggregating raw context data (e.g. the GPS data of the user) to high level context data (e.g. the users semantic location in the world). From this, the symbolic reasoner produces only activities which are consistent with the observed context (e.g. if the user is at a library, they are not doing sports). The evaluation of DUSTIN shows that it is able to outperform other state-of-the-art neuro-symbolic approaches while achieving a high sample efficiency. In comparison with other hybrid approaches that employ knowledge infusion [2, 17], DUSTIN is the first to apply the knowledge infusion within the deep model for the task of recognizing human activities.

Chapter 4

Methods

Important notes: It is important that examples are shuffled to ensure that they are independent and identically distributed. If this is not done the samples are not independent, e.g. first action is much more likely. For segementing small values work much better, in Literature often we see window sizes of 0.5 ms to 2 s, but MAF does not seem to cope well with high dimensional inputs, maybe try out 16 samples or so (128 ms)

Observation: using z-score over normalizing every individual sample in the range [0.1] often results in inf or -inf densities on validation samples. Somehow the learned distribution becomes unimodal and only has not-nan densities on one class and nan densities everywhere else. This problem can be solved by the following measures: With H=1024, jitter the training set enough $\sigma = 0.35$. With H=512, we don't need to jitter. With a small learning rate $lr = 1e - 5$ no jittering is needed and H=1024 works

Chapter 5

Experimental Evaluation

5.1 Settings

5.2 Experiments

5.3 Results

Dataset	Window	W_size	W_step	Augment	Noise	MVN LL	MAF LL	MVN ACC	MAF ACC
CARROTS	False	0	0	False	False	63.0739	68.2259	0.6006	0.5384
CARROTS	False	0	0	False	True	56.7594	59.6865	0.5932	0.6037
CARROTS	False	0	0	True	False	52.1412	68.1716	0.5683	0.5624
CARROTS	False	0	0	True	True	45.6814	59.1802	0.5103	0.6166
CARROTS	True	26	13	False	False	2986.2914	1555.2181	0.3299	0.3290
CARROTS	True	26	13	False	True	2016.7640	1416.1207	0.4398	0.3861
CARROTS	True	26	13	True	False	1881.0891	1720.2389	0.4312	0.4121
CARROTS	True	26	13	True	True	1566.8524	1513.4050	0.4909	0.5351
CARROTS	True	8	4	True	False	530.5136	695.9487	0.3759	0.5341
CARROTS	True	64	32	True	False	1375.9667	3319.6480	0.1215	0.4030
UCIHAR	True	0	0	False	False	717.9488	351.7945	0.9444	0.6291

Chapter 6

Conclusion

6.1 Summary

6.2 Future Work

Bibliography

[1] Luca Arrotta, Gabriele Civitarese, and Claudio Bettini. Knowledge Infusion for Context-Aware Sensor-Based Human Activity Recognition. In *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8, June 2022. ISSN: 2693-8340.

[2] Claudio Bettini, Gabriele Civitarese, and Riccardo Presotto. CAVIAR: Context-driven Active and Incremental Activity Recognition. *Knowledge-Based Systems*, 196:105816, May 2020.

[3] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. Deep Learning for Sensor-based Human Activity Recognition: Overview, Challenges, and Opportunities. *ACM Computing Surveys*, 54(4):1–40, May 2022.

[4] Liming Chen, Jesse Hoey, Chris D. Nugent, Diane J. Cook, and Zhiwen Yu. Sensor-Based Activity Recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):790–808, November 2012. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).

[5] Artur d'Avila Garcez and Luis C. Lamb. Neurosymbolic AI: The 3rd Wave, December 2020. arXiv:2012.05876 [cs].

[6] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked Autoencoder for Distribution Estimation. page 9.

[7] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, November 2021. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[8] Frank Krüger, Martin Nyolt, Kristina Yordanova, Albert Hein, and Thomas Kirste. Computational State Space Models for Activity and Intention Recognition. A Feasibility Study. *PLOS ONE*, 9(11):e109381, November 2014. Publisher: Public Library of Science.

[9] Stefan Lüdtke, Fernando Moya Rueda, Waqas Ahmed, Gernot A. Fink, and Thomas Kirste. Human Activity Recognition using Attribute-Based Neural Networks and Context Information, October 2021. arXiv:2111.04564 [cs, eess].

[10] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural Probabilistic Logic Programming. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[11] Fernando Moya Rueda, René Grzeszick, Gernot A. Fink, Sascha Feldhorst, and Michael Ten Hompel. Convolutional Neural Networks for Human Activity Recognition Using Body-Worn Sensors. *Informatics*, 5(2):26, June 2018. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[12] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[13] Luc De Raedt. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. page 6.

[14] Miquel Ramirez and Hector Geffner. Goal Recognition over POMDPs: Inferring the Intention of a POMDP Agent. In *Twenty-second international joint conference on artificial intelligence*, page 6, 2011.

[15] Fernando Moya Rueda, Stefan Lüdtke, Max Schröder, Kristina Yordanova, Thomas Kirste, and Gernot A. Fink. Combining Symbolic Reasoning and Deep Learning for Human Activity Recognition. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 22–27, March 2019.

[16] Yoli Shavit and Itzik Klein. Boosting Inertial-Based Human Activity Recognition With Transformers. *IEEE Access*, 9:53540–53547, 2021. Conference Name: IEEE Access.

[17] Abdul Syafiq Abdull Sukor, Ammar Zakaria, Norasmadi Abdul Rahim, Latifah Munirah Kamarudin, Rossi Setchi, and Hiromitsu Nishizaki. A hybrid

approach of knowledge-driven and data-driven reasoning for activity recognition in smart homes. *Journal of Intelligent & Fuzzy Systems*, 36(5):4177–4188, May 2019.

Appendix A

Program Code / Resources

The source code, a documentation, some usage examples, and additional test results are available at ...

They as well as a PDF version of this thesis is also contained on the CD-ROM attached to this thesis.

Appendix B

Further Experimental Results

In the following further experimental results are ...

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Master-/Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Er- klärung rechtliche Folgen haben wird.

Mannheim, den 31.09.2022 Unterschrift