Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables

Nils Y. Hammerla\textsuperscript{1,2}, Shane Halloran\textsuperscript{2}, Thomas Plötz\textsuperscript{2}

\textsuperscript{1}babylon health, London, UK
\textsuperscript{2}Open Lab, School of Computing Science, Newcastle University, UK
nils.hammerla@babylonhealth.com

Abstract

Human activity recognition (HAR) in ubiquitous computing is beginning to adopt deep learning to substitute for well-established analysis techniques that rely on hand-crafted feature extraction and classification methods. However, from these isolated applications of custom deep architectures it is difficult to gain an overview of their suitability for problems ranging from the recognition of manipulative gestures to the segmentation and identification of physical activities like running or ascending stairs. In this paper we rigorously explore deep, convolutional, and recurrent approaches across three representative datasets that contain movement data captured with wearable sensors. We describe how to train recurrent approaches in this setting, introduce a novel regularisation approach, and illustrate how they outperform the state-of-the-art on a large benchmark dataset. We investigate the suitability of each model for HAR, across thousands of recognition experiments with randomly sampled model configurations, explore the impact of hyperparameters using the FANOVA framework, and provide guidelines for the practitioner who wants to apply deep learning in their problem setting.

1 Introduction

Deep learning represents the biggest trend in machine learning over the past decade. Since the inception of the umbrella term the diversity of methods it encompasses has increased rapidly, and will continue to do so driven by the resources of both academic and commercial interests. Deep learning has become accessible to everyone via machine learning frameworks like Torch\textsuperscript{7} [Collobert et al., 2011], and has had significant impact on a variety of application domains [LeCun et al., 2015].

One field that has yet to benefit from deep learning is Human Activity Recognition (HAR) in Ubiquitous Computing (ubicomp). The dominant technical approach in HAR includes sliding window segmentation of time-series data captured with body-worn sensors, manually designed feature extraction procedures, and a wide variety of (supervised) classification methods [Bulling et al., 2014]. In many cases, these relatively simple means suffice to obtain impressive recognition accuracies. However, more elaborate behaviours which are, for example, of interest in medical applications, pose a significant challenge to this manually tuned approach [Hammerla et al., 2015]. Furthermore, some work has suggested that the dominant technical approach in HAR benefits from biased evaluation settings [Hammerla and Plötz, 2015], which may explain some of the apparent inertia in the field toward the adoption of deep learning techniques.

Deep learning has the potential to have significant impact on HAR in ubicomp. It can substitute for manually designed feature extraction procedures which lack the robust physiological basis that benefits other fields such as speech recognition. However, for the practitioner it is difficult to select the most suitable deep learning method for their application. Work that promotes deep learning almost always provides only the performance of the best system, and rarely includes details on how its seemingly optimal parameters were discovered. As only a single score is reported it remains unclear how this peak performance compares with the average during parameter exploration, and how likely a practitioner would be to discover a system that performs similarly well.

In this paper we provide the first unbiased and systematic exploration of the performance of state-of-the-art deep learning approaches on three different recognition problems typical for HAR in ubicomp. The training process for deep, convolutional, and recurrent models is described in detail, and we introduce a novel approach to regularisation for recurrent networks. In more than 4,000 experiments we investigate the suitability of each model for different tasks in HAR, explore the impact each model’s hyper-parameters have on performance, and conclude guidelines for the practitioner who wants to apply deep learning to their application scenario. Over the course of these experiments we discover that recurrent networks outperform the state-of-the-art and that they allow novel types of real-time application of HAR through sample-by-sample prediction of physical activities.

2 Deep Learning in Ubiquitous Computing

Movement data collected with body-worn sensors are multivariate time-series data with relatively high spatial and temporal resolution (e.g. 20Hz - 200Hz). Analysis of this data in ubicomp is typically follows a pipeline-based approach [Bulling et al., 2014]. The first step is to segment the time-
series data into contiguous segments (or frames), either driven by some signal characteristics such as signal energy [Plötz et al., 2012], or through a sliding-window segmentation approach. From each of the frames a set of features is extracted, which most commonly include statistical features or stem from the frequency domain.

The first deep learning approach explored to substitute for this manual feature selection corresponds to deep belief networks as auto-encoders, trained generatively with Restricted Boltzmann Machines (RBMs) [Plötz et al., 2011]. Results were mixed, as in most cases the deep model was outperformed by a combination of principal component analysis and a statistical feature extraction process. Subsequently a variety of projects have explored the use of pre-trained, fully-connected networks for automatic assessment in Parkinson’s Disease [Hammerla et al., 2015], as emission model in Hidden Markov Models (HMMs) [Zhang et al., 2015; Alsheikh et al., 2015], and to model audio data for ubicomp applications [Lane et al., 2015].

The most popular approach so far in ubicomp relies on convolutional networks. Their performance for a variety of activity recognition tasks was explored by a number of authors [Zeng et al., 2014; Ronao and Cho, 2015; Yang et al., 2015; Ronao and Cho, 2015]. Furthermore, convolutional networks have been applied to specific problem domains, such as the detection of stereotypical movements in Autism [Rad et al., 2015], where they significantly improved upon the state-of-the-art.

Individual frames of movement data in ubicomp are usually treated as statistically independent, and applications of sequential modelling techniques like HMMs are rare [Bulling et al., 2014]. However, approaches that are able to exploit the temporal dependencies in time-series data appear as the natural choice for modelling human movement captured with sensor data. Deep recurrent networks, most notably those that rely on Long Short-Term Memory cells (LSTMs) [Hochreiter and Schmidhuber, 1997], have recently achieved impressive performance across a variety of scenarios (e.g. [Gregor et al., 2015; Hermann et al., 2015]). Their application to HAR has been explored in two settings. First, [Neverova et al., 2015] investigated a variety of recurrent approaches to identify individuals based on movement data recorded from their mobile phone in a large-scale dataset. Secondly, [Ordoñez and Roggen, 2016] compared the performance of a recurrent approach to CNNs on two HAR datasets, representing the current state-of-the-art performance on Opportunity, one of the datasets utilised in this work. In both cases, the recurrent network was paired with a convolutional network that encoded the movement data, effectively employing the recurrent network to only model temporal dependencies on a more abstract level. Recurrent networks have so far not been applied to model movement data at the lowest possible level, which is the sequence of individual samples recorded by the sensor(s).

3 Comparing deep learning for HAR

While there has been some exploration of deep models for a variety of application scenarios in HAR there is still a lack of a systematic exploration of deep learning capabilities. Authors report to explore the parameter space in preliminary experiments, but usually omit the details. The overall process remains unclear and difficult to replicate. Instead, single instantiations of e.g. CNNs are presented that show good performance in an application scenario. Solely reporting peak performance figures does, however, not reflect the overall suitability of a method for HAR in ubicomp, as it remains unclear how much effort went into tuning the proposed approach, and how much effort went into tuning other approaches it was compared to. How likely is a practitioner to find a parameter configuration that works similarly well for their application? How representative is the reported performance across the models compared during parameter exploration? Which parameters have the largest impact on performance? These questions are important for the practitioner, but so far remain unanswered in related work.

In this paper we provide the first unbiased comparison of the most popular deep learning approaches on three representative datasets for HAR in ubicomp. They include typical application scenarios like manipulative gestures, repeti-
tive physical activities, and a medical application of HAR in Parkinson’s disease. We compare three types of models that are described below. To explore the suitability of each method we chose reasonable ranges for each of their hyperparameters and randomly sample model configurations. We report on their performance across thousands of experiments and analyse the impact of hyperparameters for each approach.

3.1 Deep feed-forward networks (DNN)

We implemented deep feed-forward networks, which correspond to a neural network with up to five hidden layers followed by a softmax-group. The DNN represents a sequence of non-linear transformations to the input data of the network. We follow convention and refer to a network with N hidden layers as an N-layer network. Each hidden layer contains the same number of units, and corresponds to a linear transformation and a rectified-linear (ReLU) activation function. We use two different regularisation techniques: i) Dropout: during training each unit in each hidden layer is set to zero with a probability $p_{\text{drop}}$, and during inference the output of each unit is scaled by $1/p_{\text{drop}}$ [Srivastava et al., 2014] (dropout-rate is fixed to 0.5 for all experiments); ii) Max-in norm: After each mini-batch the incoming weights of each unit in the network are scaled to have a maximum euclidean length of $d_{\text{in}}$. To limit the number of hyperparameters of the approach we chose not to perform any generative pre-training and to solely rely on a supervised learning approach. The input data fed into the network corresponds to frames of movement data. Each frame consists of a varying number of samples from $\mathbb{R}^d$, which are simply concatenated into a single vector $F_t \in \mathbb{R}^{s \times d}$. The model is illustrated in figure 1(d).

The DNN is trained in a mini-batch approach, where each mini-batch contains 64 frames and is stratified with respect to the class distribution in the training-set. We minimise the negative log likelihood using stochastic gradient descent.

3.2 Convolutional networks (CNN)

CNNs aim to introduce a degree of locality in the patterns matched in the input data and to enable translational invariance with respect to the precise location (i.e. time of occurrence) of each pattern within a frame of movement data. We explore the performance of convolutional networks and follow suggestions by [Srivastava et al., 2014] in architecture and regularisation techniques. The overall CNN architecture is illustrated in figure 1(c). Each CNN contains at least one temporal convolution layer, one pooling layer and at least one fully connected layer prior to a top-level softmax-group. The temporal convolution layer corresponds to a convolution of the input sequence with $n_f$ different kernels (feature maps) of width $k_w$. Subsequent max-pooling is looking for the maximum within a region of width $m_w$ and corresponds to a subsampling, introducing translational invariance to the system. Width of the max-pooling was fixed to 2 throughout all experiments. The output of each max-pooling layer is transformed using a ReLU activation function. The subsequent fully connected part effectively corresponds to a DNN and follows the same architecture outlined above.

For regularisation we apply dropout after each max-pooling or fully-connected layer, where the dropout-probability $p_{\text{drop}}$ in layer $i$ is fixed for all experiments ($p_{\text{drop}}^1 = 0.1$, $p_{\text{drop}}^2 = 0.25$, $p_{\text{drop}}^3 = 0.5$). Similar to the DNN we also perform max-in norm regularisation as suggested in [Srivastava et al., 2014]. The input data fed into the CNN corresponds to frames of movement data as in the DNN. However, instead of concatenating the different input dimensions the matrix-structure is retained ($F_t \in \mathbb{R}^{s \times d}$). The CNN is trained using stratified mini-batches (64 frames) and stochastic gradient descent to minimise negative log likelihood.

3.3 Recurrent networks

In order to exploit the temporal dependencies within the movement data we implemented recurrent neural networks based on LSTM cells in their vanilla variant which does not contain peephole connections [Greff et al., 2015]. This architecture is recurrent as some of the connections within the network form a directed cycle, where the current timestep $t$ considers the states of the network in the previous timestep $t-1$. LSTM cells are designed to counter the effect of diminishing gradients if error derivatives are backpropagated through many layers “through time” in recurrent networks [Hochreiter et al., 2001]. Each LSTM cell (unit) keeps track of an internal state (the constant error carousel) that represents its “memory”. Over time the cells learn to output, overwrite, or reset their internal memory based on their current input and the history of past internal states, leading to a system capable of retaining information across hundreds of time-steps [Hochreiter and Schmidhuber, 1997].

We implement two flavours of LSTM recurrent networks: i) deep forward LSTMs contain multiple layers of recurrent units and are connected “forward” in time (see figure 1(a)); and ii) bi-directional LSTMs which contain two parallel recurrent layers that stretch both into the “future” and into the “past” of the current time-step, followed by a layer that concatenates their internal states for timestep $t$ (see figure 1(b)).

Practically these two flavours differ significantly in their application requirements. A forward LSTM contextualises the current time-step based on those it has seen previously, and is inherently suitable for real-time applications where, at inference time, the “future” is not yet known. Bi-directional LSTMs on the other hand use both the future and past context to interpret the input at timestep $t$, which makes them suitable for offline analysis scenarios.

In this work we apply recurrent networks in three different settings, each of which is trained to minimise the negative log likelihood using adagrad [Duchi et al., 2011] and subject to max-in norm regularisation.

In the first case the input data fed into the network at any given time $t$ corresponds to the current frame of movement data, which stretches a certain length of time and whose dimensions have been concatenated (as in the DNN above). We denote this model as LSTM-F. The second application case of forward LSTMs represents a real-time application, where each sample of movement data is presented to the network in the sequence they were recorded, denoted LSTM-S. The final scenario sees the application of bi-directional LSTMs to
corresponding model to the test-set. That showed the best validation-set performance and apply the performance for 10 subsequent epochs. We select the epoch 30 epochs, training stops if there is no increase in validation at least 30 epochs and for a maximum of 300 epochs. After of the model on the validation set. Each model is trained for except for the largest networks.

In HAR the context of an individual sample of movement data is not well defined, at least beyond immediate correlations between neighbouring samples, and likely depends on the type of movement and its wider behavioural context. This is a problem well known in the field, and affects the choice of window length for sliding window segmentation [Bulling et al., 2014].

In order to construct b mini-batches that are used to train the RNN we initialise a number of positions (pi) between the start and end of the training-set. To construct a mini-batch we extract the L samples that follow each position in (pi)i=1,...,b and increase (pi) by L steps, possibly wrapping around the end of the sequence. We found that it is important to initialise the positions randomly to avoid oscillations in the gradients. While this approach retains the ordering of the samples presented to the RNN it does not allow for stratification of each mini-batch w.r.t. class-distribution. Figure 2 illustrates the approach.

Training on long sequences has a further issue that is addressed in this work. If we use the approach outlined above to train a sufficiently large RNN it may “memorise” the entire input-output sequence implicitly, leading to poor generalisation performance. In order to avoid this memorisation we need to introduce “breaks” where the internal states of the RNN are reset to zero: after each mini-batch we decide to retain the state of the RNN with a carry-over probability pcarry, and reset it to zero otherwise. This is a novel form of regularisation of RNNs, which should be useful for similar applications of RNNs.

3.4 Training RNNs for HAR

Common applications for RNNs include speech recognition and natural language processing. In these settings the context for an input (e.g. a word) is limited to it’s surrounding entities (e.g. a sentence, paragraph). Training of RNNs usually treats these contextualised entities as a whole, for example by training an RNN on complete sentences.

4 Experiments

The different hyper-parameters explored in this work are listed in table 1. The last column indicates the number of parameter configurations sampled for each dataset, selected to represent an equal amount of computation time. We conduct experiments on three benchmark datasets representative of the problems typical for HAR (described below). Experiments were run on a machine with three GPUs (Nvidia GTX 980 Ti), where two model configurations are run on each GPU except for the largest networks.

After each epoch of training we evaluate the performance of the model on the validation set. Each model is trained for at least 30 epochs and for a maximum of 300 epochs. After 30 epochs, training stops if there is no increase in validation performance for 10 subsequent epochs. We select the epoch that showed the best validation-set performance and apply the corresponding model to the test-set.

4.1 Datasets

We select three datasets typical for HAR in ubicomp for the exploration in this work. Each dataset corresponds to a different application of HAR. The first dataset, Opportunity, contains manipulative gestures like opening and closing doors, which are short in duration and non-repetitive. The second, PAMAP2, contains prolonged and repetitive physical activities typical for systems aiming to characterise energy expenditure. The last, Daphnet Gait, corresponds to a medical application where participants exhibit a typical motor complication in Parkinson’s disease that is known to have a large inter-subject variability. Below we detail each dataset.

The Opportunity dataset (Opp) [Chavarriaga et al., 2013] consists of annotated recordings from on-body sensors from 4 participants instructed to carry out common kitchen activities. Data is recorded at a frequency of 30Hz from 12 locations on the body, and annotated with 18 mid-level gesture annotations (e.g. Open Door / Close Door). For each subject, data from 5 different runs is recorded. We used the subset of sensors that did not show any packet-loss, which included accelerometer recordings from the upper limbs, the back, and complete IMU data from both feet. The resulting dataset had 79 dimensions. We use run 2 from subject 1 as our validation set, and replicate the most popular recognition challenge by using runs 4 and 5 from subject 2 and 3 in our test set. The remaining data is used for training. For frame-by-frame analysis, we created sliding windows of duration 1 second and 50% overlap. The resulting training-set contains approx. 650k samples (43k frames).

The PAMAP2 dataset [Reiss and Stricker, 2012] consists of recordings from 9 participants instructed to carry out 12 lifestyle activities, including household activities and a variety of exercise activities (Nordic walking, playing soccer, etc). Accelerometer, gyroscope, magnetometer, temperature and heart rate data are recorded from inertial measurement units located on the hand, chest and ankle over 10 hours (in total). The resulting dataset has 52 dimensions. We used runs
In order to estimate the impact of each hyper-parameter on the performance observed across all experiments we apply the fANOVA analysis framework. fANOVA [Hutter et al., 2014] determines the extent to which each hyper-parameter contributes to a network’s performance. It builds a predictive model (random forest) of the model performance as a function of the model’s hyper-parameters. This non-linear model is then decomposed into marginal and joint interaction functions of the hyper-parameters, from which the percentage contribution to overall variability of network performance is obtained. fANOVA has been used previously to explore the hyper-parameters in recurrent networks by [Greff et al., 2015].

For the practitioner it is important to know which aspect of the model is the most crucial for performance. We grouped the hyper-parameters of each model into one of three categories (see table 1): i) \textit{learning}, parameters that control the learning process; ii) \textit{regularisation}, parameters that limit the modelling capabilities of the model to avoid overfitting; and iii) \textit{architecture}, parameters that affect the structure of the model. Based on the variability observed for each hyper-parameter we estimate the variability that can be attributed to each parameter category, and to higher order interactions between categories.

### Table 1: Hyper-parameters of the models and the ranges of values explored in experiments.

<table>
<thead>
<tr>
<th>Category</th>
<th>Architecture</th>
<th>Learning</th>
<th>Regularisation</th>
<th>#layers</th>
<th>#bats</th>
<th>conv-layers</th>
<th>kW1</th>
<th>kW2</th>
<th>kW3</th>
<th>nl1</th>
<th>nl2</th>
<th>nl3</th>
<th>experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>$10^{-4}$</td>
<td>- 0.99 4.0</td>
<td>3 2048</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>$10^{-5}$</td>
<td>0.0 0.0 0.5</td>
<td>1 64</td>
<td>1</td>
<td>1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>256</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>$10^{-4}$</td>
<td>- 0.99 4.0</td>
<td>3 2048</td>
<td>3 9</td>
<td>5 3</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>$10^{-5}$</td>
<td>0.0 0.0 0.5</td>
<td>1 64</td>
<td>1</td>
<td>3 3 3 16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM-F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>$10^{-4}$</td>
<td>- 196 4.0</td>
<td>3 384</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>$10^{-5}$</td>
<td>- 32 0.5 0.0</td>
<td>1 64</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM-S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>$10^{-4}$</td>
<td>- 196 4.0</td>
<td>3 384</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>$10^{-5}$</td>
<td>- 32 0.5 0.0</td>
<td>1 64</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b-LSTM-S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>max</td>
<td>$10^{-4}$</td>
<td>- 196 4.0</td>
<td>3 384</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>min</td>
<td>$10^{-5}$</td>
<td>- 32 0.5 0.0</td>
<td>1 64</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1 and 2 for subject 5 in our validation set and runs 1 and 2 for subject 6 in our test set. The remaining data is used for training. In our analysis, we downsampled the accelerometer data to 33.3Hz in order to have a temporal resolution comparable to the Opportunity dataset. For frame-by-frame analysis, we replicate previous work with non-overlapping sliding windows of 5.12 seconds duration with one second stepping between adjacent windows (78% overlap) [Reiss and Stricker, 2012]. The training-set contains approx. 470k samples (14k frames).

The Daphnet Gait dataset (DG) [Bachlin et al., 2009] consists of recordings from 10 participants affected with Parkinson’s Disease (PD), instructed to carry out activities which are likely to induce freezing of gait. Freezing is a common motor complication in PD, where affected individuals struggle to initiate movements such as walking. The objective is to detect these freezing incidents, with the goal to inform a future situated prompting system. This represents a two-class recognition problem. Accelerometer data was recorded from above the ankle, above the knee and on the trunk. The resulting dataset has 9 dimensions. We used run 1 from subject 9 in our validation set, runs 1 and 2 from subject 2 in our test set, and used the rest for training. In our analysis, we downsampled the accelerometer data to 32Hz. For frame-by-frame analysis, we created sliding windows of 1 second duration and 50% overlap. The training-set contains approx. 470k samples (30k frames).

### 4.2 Influence of hyper-parameters

In order to estimate the impact of each hyper-parameter on the performance observed across all experiments we apply the fANOVA analysis framework. For the practitioner it is important to know which aspect of the model is the most crucial for performance. We grouped the hyper-parameters of each model into one of three categories (see table 1): i) \textit{learning}, parameters that control the learning process; ii) \textit{regularisation}, parameters that limit the modelling capabilities of the model to avoid overfitting; and iii) \textit{architecture}, parameters that affect the structure of the model. Based on the variability observed for each hyper-parameter we estimate the variability that can be attributed to each parameter category, and to higher order interactions between categories.

### 4.3 Performance metrics

As the datasets utilised in this work are highly biased we require performance metrics that are independent of the class distribution. We estimate the mean f1-score, which corresponds to the harmonic mean of precision and recall:

\[
F_m = \frac{2}{|c|} \sum_{c} \frac{\text{prec}_c \times \text{recall}_c}{\text{ prec}_c + \text{recall}_c}
\]

(1)

Related work has previously used the weighted f1-score as primary performance metric (for Opportunity). In order to compare our results to the state-of-the-art we estimate the weighted f1-score:

\[
F_w = 2 \sum_{c} \frac{N_c \times \text{prec}_c \times \text{recall}_c}{N_{total}}
\]

(2)

where \(N_c\) is the number of samples in class \(c\), and \(N_{total}\) is the total number of samples.

### 5 Results

Results across all experiments are illustrated in figure 3. Graphs (a) to (c) show the cumulative distribution of the mean f1-score (OPP, PAMAP2), or of the f1-score for the objective class (DG), across all the models utilised in this work. Graph (d) illustrates the importance of each group of hyper-parameters estimated using fANOVA. Table 2 lists the best performing systems and baseline scores of other deep approaches on Opportunity. Furthermore it lists the difference between the best performing model and the median model for each setting.
Figure 3: (a)-(c): Cumulative distribution of recognition performance for each dataset. (d): results from fANOVA analysis, illustrating impact of hyperparameter-categories on recognition performance (see table 1).

Table 2: Best results obtained for each model and dataset, along with some baselines for comparison. Delta from median (lower part of table) refers to the absolute difference between peak and median performance across all experiments.

Overall we observe a large spread of peak performances between models on OPP and DG, with more than 15% mean f1-score between the best performing approach (b-LSTM-S) and the worst (DNN) on OPP (12% on DG) (see table 2). On PAMAP2 this difference is smaller, but still considerable at 7%. The best performing approach on OPP (b-LSTM-S) outperforms the current state-of-the-art by a considerable margin of 4% mean f1-score (1% weighted f1-score). The best CNN discovered in this work further outperforms previous results reported in the literature for this type of model by more than 5% mean f1-score and weighted f1-score (see table 2). The good performance of recurrent approaches, which model movement at the sample level, holds the potential for novel applications in HAR, as they alleviate the need for segmentation of the time-series data, which is discussed below.

The distributions of performance scores differ between the models investigated in this work. CNNs show the most characteristic behaviour: a fraction of model configurations do not work at all (e.g. 20% on PAMAP2), while the remaining configurations show little variance in their performance. On PAMAP2, for example, the difference between the peak and median performance is only 7% mean f1-score (see table 2). The DNNs show the largest spread between peak and median performance of all approaches of up to 35.7% on OPP. Both forward RNNs (LSTM-F, LSTM-S) show similar behaviour across the different datasets. Practically all of their configurations explored on PAMAP2 and OPP have non-trivial recognition performance.

The effect of each category of hyperparameter on the recognition performance is illustrated in figure 3(d). Interestingly, we observe the most consistent effect of the parameters in the CNN. Contrary to our expectation it is the parameters surrounding the learning process (see table 1) that have the largest main effect on performance. We expected that for this model the rich choice of architectural variants should have a larger effect.

For DNNs we do not observe a systematic effect of any category of hyperparameter. On PAMAP2, the correct learning parameters appear to be the most crucial. On OPP it is the architecture of the model. Interestingly we observed that relatively shallow networks outperform deeper variants. There is a drop in performance for networks with more than 3 hidden layers. This may be related to our choice to solely rely on supervised training, where a generative pre-training may improve the performance of deeper networks.

The performance of the frame-based RNN (LSTM-F) on OPP depends critically on the carry-over probability introduced in this work. Both always retaining the internal state and always forgetting the internal state lead to the lowest performance. We found that \( p_{\text{carry}} \) of 0.5 works well for most settings. Our findings merit further investigation, for example into a carry-over schedule, which may further improve LSTM performance.

Results for sample-based forward LSTMs (LSTM-S) mostly confirm earlier findings for this type of model that found learning-rate to be the most crucial parameter [Greff et al., 2015]. However, for bi-directional LSTMs (b-LSTM-S) we observe that the number of units in each layer has a surprisingly large effect on performance, which should motivate practitioners to first focus on tuning this parameter.

6 Discussion
In this work we explored the performance of state-of-the-art deep learning approaches for Human Activity Recognition
using wearable sensors. We described how to train recurrent approaches in this setting and introduced a novel regularisation approach. In thousands of experiments we evaluated the performance of the models with randomly sampled hyper-parameters. We found that bi-directional LSTMs outperform the current state-of-the-art on Opportunity, a large benchmark dataset, by a considerable margin.

However, interesting from a practitioner’s point of view is not the peak performance for each model, but the process of parameter exploration and insights into their suitability for different tasks in HAR. All experiments in this work were based on a random exploration of the parameter space for each model. We found that performing hundreds or thousands of recognition experiments is a suitable way to find configurations with state-of-the-art performance, and recommend this approach to other practitioners. Additionally, more sophisticated exploration methods (see e.g. [Snoek et al., 2012]) are likely to discover good configurations in fewer iterations.

In our experiments we found that RNNs outperform convolutional networks significantly on activities that are short in duration but have a natural ordering, where a recurrent approach benefits from the ability to contextualise observations across long periods of time. We see some indication that this ability and the sensitivity to long term dependencies between activities (such as opening / closing doors in Opportunity), introduce constraints on study design. In order for the RNN to learn long-term effects it needs to be trained with data collected in a naturalistic environment, where the sequence of activities is similar to that in the envisioned application scenario. Movement data which was collected adhering to a fixed protocol will be less suitable to train RNNs, as long-term context brings little to no benefit over localised pattern matching during training and inference (and may even lead to overfitting). This would explain, at least to some extent, the poor performance of RNNs on PAMAP2, a data-set that contains only well segmented, scripted activities.

However, not all application scenarios of HAR in ubicomp require a model that can maintain long term dependencies between activities. The activities a user engages in may already be pre-determined, for example in applications like gait analysis, or skill assessment in sports [Khan et al., 2015], where the analysis focuses on (changes in) short term movement patterns. For these applications we recommend the use of CNNs, as they can reliably model local patterns, are straight-forward to train, and are supported by the majority of deep learning frameworks “out-of-the-box”. As CNNs (and DNNs) only focus on a fixed context-length they allow for more liberty in study design, where naturalistic, as well as protocol-driven data can be useful for training. For CNNs we recommend to start exploring learning-rates, before optimising the architecture of the network, as the learning-parameters had the largest effect on performance in our experiments.

The approach that is probably the most approachable for the practitioner are simple feed-forward DNNs. Their peak performance across the data-sets in this work can approach both the performance of CNNs and RNNs. However, we found that DNNs are very sensitive to their hyper-parameters and require a significant investment into parameter exploration. We advise the practitioner not to discard this model even if DNNs do not work well in a preliminary exploration. Furthermore we found that results from other fields do not necessarily translate into the ubicomp domain. Deep networks with more than three hidden layers do not work well in the data-sets utilised in this work. To an extend this may be explained in the lack of a generative pre-training procedure (e.g. [Hinton et al., 2006]).

6.1 Real-time HAR through RNNs
So far the application of HAR in ubicomp relies almost exclusively on a pipeline-based architecture that involves a segmentation step which mostly corresponds to a sliding-window extraction procedure [Bulling et al., 2014]. Inherent to this approach is that it introduces a delay between the recording of movements using inertial sensors and the earliest time they can contribute significantly to the inference procedure. While for offline analysis scenarios this is of no concern, it effectively prohibits applications that require real-time processing or an immediate response to e.g. user input.

We believe that this work is the first to apply a model to movement data that allows immediate, real-time inference at the same rate as the data is collected by the sensors. While we do not explicitly explore this in this work we believe that there is significant potential in the use of sample-by-sample RNNs (LSTM-S in this work) to allow for novel application scenarios of HAR in ubicomp and in HCI in general.

Acknowledgements
Part of this work was supported by the Engineering and Physical Sciences Research Council, Centre for Doctoral Training in Cloud Computing for Big Data [grant number EP/L015358/1]

References


