frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 30 September 2022
DOI 10.3389/fcomp.2022.924954

OPEN ACCESS

EDITED BY
Katia Vega,
University of California, Davis,
United States

REVIEWED BY
Sten Hanke,
FH Joanneum, Austria
Fuhong Min,
Nanjing Normal University, China

*CORRESPONDENCE
Marius Bock
marius.bock@uni-siegen.de

SPECIALTY SECTION
This article was submitted to Mobile and Ubiquitous Computing, a section of the journal Frontiers in Computer Science

RECEIVED 20 April 2022
ACCEPTED 01 September 2022
PUBLISHED 30 September 2022

CITATION
Bock M, Hoelzemann A, Moeller M and Van Laerhoven K (2022) Investigating (re)current state-of-the-art in human activity recognition datasets. *Front. Comput. Sci.* 4:924954. doi: 10.3389/fcomp.2022.924954

Investigating (re)current state-of-the-art in human activity recognition datasets

Marius Bock[1,2]*, Alexander Hoelzemann[2], Michael Moeller[1] and Kristof Van Laerhoven[2]

[1]Computer Vision, Department of Electrical Engineering and Computer Science, University of Siegen, Siegen, Germany, [2]Ubiquitous Computing, Department of Electrical Engineering and Computer Science, University of Siegen, Siegen, Germany

Many human activities consist of physical gestures that tend to be performed in certain sequences. Wearable inertial sensor data have as a consequence been employed to automatically detect human activities, lately predominantly with deep learning methods. This article focuses on the necessity of recurrent layers—more specifically Long Short-Term Memory (LSTM) layers—in common Deep Learning architectures for Human Activity Recognition (HAR). Our experimental pipeline investigates the effects of employing none, one, or two LSTM layers, as well as different layers' sizes, within the popular DeepConvLSTM architecture. We evaluate the architecture's performance on five well-known activity recognition datasets and provide an in-depth analysis of the per-class results, showing trends which type of activities or datasets profit the most from the removal of LSTM layers. For 4 out of 5 datasets, an altered architecture with one LSTM layer produces the best prediction results. In our previous work we already investigated the impact of a 2-layered LSTM when dealing with sequential activity data. Extending upon this, we now propose a metric, r_{GP}, which aims to measure the effectiveness of learned temporal patterns for a dataset and can be used as a decision metric whether to include recurrent layers into a network at all. Even for datasets including activities without explicit temporal processes, the r_{GP} can be high, suggesting that temporal patterns were learned, and consequently convolutional networks are being outperformed by networks including recurrent layers. We conclude this article by putting forward the question to what degree popular HAR datasets contain unwanted temporal dependencies, which if not taken care of, can benefit networks in achieving high benchmark scores and give a false sense of overall generability to a real-world setting.

KEYWORDS

human activity recognition, CNN-RNNs, deep learning, network architectures, datasets

1. Introduction

Research in activity recognition has early on recognized the importance of capturing temporal dependencies in sensor-based data with early approaches using classical generative Machine Learning algorithms in order to do so (e.g., Lester et al., 2005). With the advancement of Deep Learning—becoming the de facto standard in sensor-based Human Activity Recognition (HAR)—approaches shifted to using Recurrent Neural Networks (RNNs) (e.g., Hammerla et al., 2016), Long-Short-Term Memory networks (LSTMs) (e.g., Edel and Köppe, 2016) and Transformers (e.g., Dirgová Luptáková et al., 2022). Based on findings presented in Jaakkola and Haussler (1998), early works already demonstrated the effectiveness of combining both discriminative and generative models (Lester et al., 2005). Within the area of Deep Learning, the combination of convolutional (discrimnative) and recurrent (generative) layers saw particular success, with the DeepConvLSTM (Ordóñez and Roggen, 2016) being the first architecture to take advantage of the capabilities of both types of layers. The architecture achieved state-of-the-art results on both the Opportunity challenge (Roggen et al., 2010) and Skoda Mini Checkpoint dataset (Zappi et al., 2008).

After Ordóñez and Roggen (2016) introduced us to the DeepConvLSTM, Hammerla et al. (2016) analyzed the effectiveness of the RNNs as well as their combination with convolutional layers and came to the conclusion that whether RNNs are beneficial and increase prediction performance is dependent on the nature of activities and dataset itself. The original DeepConvLSTM (Ordóñez and Roggen, 2016) architecture features a 2-layered LSTM with 128 hidden units. In Bock et al. (2021) we demonstrated that altering the DeepConvLSTM to employ a 1-layered instead of a 2-layered LSTM not only heavily decreases training time, but also significantly increases prediction performance across 5 popular HAR datasets (Roggen et al., 2010; Scholl et al., 2015; Stisen et al., 2015; Reyes-Ortiz et al., 2016; Sztyler and Stuckenschmidt, 2016). Our findings stood in contrast to the belief that one needs at least a 2-layered LSTM when dealing with sequential data (Karpathy et al., 2015). In this article we want to extend upon the work conducted by Hammerla et al. (2016) by putting it into context with what we found out in Bock et al. (2021). Unlike Hammerla et al. (2016) we use a larger variety of datasets with different types of activities and fix all parts of the training process across experiments to ensure that changes in predictive performance can be accredited to changes to the recurrent parts of the network. Furthermore, we want to go one step further by questioning the necessity of recurrent layers in sensor-based HAR altogether and introduce a second variation of the DeepConvLSTM (Ordóñez and Roggen, 2016) which has all LSTM layers removed. We include said variation into the experimental workflow we illustrated in Bock et al. (2021).

Our article's contributions are threefold:

1. We provide an in-depth analysis of the performance of the DeepConvLSTM (Ordóñez and Roggen, 2016), as well as two variations of it, on 5 popular HAR datasets (Roggen et al., 2010; Scholl et al., 2015; Stisen et al., 2015; Reyes-Ortiz et al., 2016; Sztyler and Stuckenschmidt, 2016), showing trends in which types of activities or datasets benefit the most from the removal of LSTM layers.
2. We propose a correlation factor r_{GP} which can measure the effectiveness of learned temporal patterns and which can be used as guidance to whether inclusion of recurrent layers into a Deep Learning architecture for a given HAR dataset can be predicted to be helpful.
3. Based on our findings, we extend upon our claims in Bock et al. (2021) and argue that larger LSTMs are more likely to overfit on temporal patterns of a dataset, which, depending on the use case, might end up hurting generability of trained models to a real-world setting.

2. Related work

The predictive performance of classical Machine Learning approaches highly relies on sophisticated, handcrafted features (Pouyanfar et al., 2018). In the last decade, Deep Learning has shown to outperform classical Machine Learning algorithms in many areas, e.g., image recognition (Farabet et al., 2013; Tompson et al., 2014; Szegedy et al., 2015), speech recognition (Mikolov et al., 2011; Hinton et al., 2012; Krizhevsky et al., 2012; Sainath et al., 2013) and Natural Language Processing (Collobert et al., 2011; Bordes et al., 2014; Jean et al., 2014; Sutskever et al., 2014). Much of this success can be accredited to the fact that Deep Learning does not require manual feature engineering, but is able to automatically extract discriminative features from raw data input (Najafabadi et al., 2015). The advantage of being able to apply algorithms on raw data and not being dependent on handcrafted features has led to studies investigating the effectiveness of Deep Learning in HAR, which e.g., suggested different architectures (Ordóñez and Roggen, 2016), evaluated the generality of architectures (Hammerla et al., 2016) and assessed the applicability in real-world scenarios (Guan and Plötz, 2017).

Research in activity recognition has early on recognized the importance of capturing temporal regularities and dependencies in sensor-based data (Lester et al., 2005). Early approaches involved using classical generative Machine Learning algorithms such as Hidden Markov Models (Lester et al., 2005, 2006; Patterson et al., 2005; van Kasteren et al., 2008; Reddy et al., 2010), Conditional Random Fields (Liao et al., 2005; van Kasteren et al., 2008) or Bayesian networks (Patterson et al., 2005). With the advancement of Deep Learning methods like Recurrent Neural Networks (RNNs) (Hammerla et al., 2016;

Inoue et al., 2018), Long-Short-Term Memory (LSTM) networks (Edel and Köppe, 2016; Ordóñez and Roggen, 2016), Gated Recurrent Units (GRUs) (Xu et al., 2019; Abedin et al., 2021; Dua et al., 2021) and Transformers (Haresamudram et al., 2020; Dirgová Luptáková et al., 2022) became the de facto standard when trying to model temporal dependencies.

Based on the work of Jaakkola and Haussler (1998) and Lester et al. (2005) suggested to combine both generative and discriminative models in order to overcome weaknesses of the generative models in the context of HAR. More recently (Ordóñez and Roggen, 2016) introduced the DeepConvLSTM, a deep learning architecture which is a hybrid model combining both convolutional and recurrent layers. Similar to Lester et al. (2005) and Ordóñez and Roggen (2016) claim that by combining both types of layers the network is able to automatically extract discriminative features and model temporal dependencies. Extending up on the work of Ordóñez and Roggen (2016), researchers proposed variations of the DeepConvLSTM e.g., by appending attention layers to the original architecture (Murahari and Plötz, 2018) or adding dilated convolution layers in addition to normal convolution layers (Xi et al., 2018). Other publications followed up on the idea of combining convolutional and recurrent layers proposing their own architectures, e.g., combining Inception modules and GRUs (Xu et al., 2019), self-attention mechanisms and GRUs (Abedin et al., 2021) or having two ConvLSTM networks handling different time lengths to analyze more complex temporal hierarchies (Yuki et al., 2018).

Upon the experiments conducted by Karpathy et al. (2015) and Chen et al. (2021) claim within their recent survey paper on Deep Learning for HAR that *"the depth of an effective LSTM-based RNN needs to be at least two when processing sequential data."* Within our recent publication we investigated the effect of employing a 1-layered instead of a 2-layered LSTM within the DeepConvLSTM architecture (Bock et al., 2021). As Karpathy et al. (2015) obtained their results using character-level language models, i.e., text data, our paper aimed at challenging the belief that their claim is applicable to sensor-based HAR. With this publication we want to go one step further in analyzing the necessity of recurrent layers in HAR altogether by completely removing the LSTM, and thus all recurrent layers, from the DeepConvLSTM (Ordóñez and Roggen, 2016). Our analysis is most similar to the one conducted by Hammerla et al. (2016) which compared Deep Neural Networks (DNNs) against CNNs, RNNs and, as a combination of both, the DeepConvLSTM (Ordóñez and Roggen, 2016). Hammerla et al. (2016) based their analysis on three datasets namely the Physical Activity Montioring (PAMAP2) (Reiss and Stricker, 2012), the Daphnet Freezing of Gait (Bachlin et al., 2009) and the Opportunity challenge (Roggen et al., 2010) dataset. Hammerla et al. (2016) concluded that RNNs are beneficial and increase prediction performance for activities that are short in duration and have a natural ordering, while for movement data which focuses on short-term changes in patterns CNNs are a better option. Unlike Hammerla et al. (2016) we additionally analyze datasets which include complex (Scholl et al., 2015) and transitional activities (Reyes-Ortiz et al., 2016) as well as datasets which were recorded in a naturalistic environment (Scholl et al., 2015; Stisen et al., 2015; Sztyler and Stuckenschmidt, 2016). Furthermore, we made sure to use the same hyperparameters across all experiments to give every variation the same starting point. This way we want to ensure that differences in performance can be singled down to changes in the recurrent layers.

3. Methodology

Contrary to the belief that one needs at least a two-layered LSTM when dealing with sequential data Karpathy et al. (2015) and Bock et al. (2021) we proposed to change the DeepConvLSTM to employ a one-layered LSTM. Extending upon these experiments we now want to investigate the overall necessity of recurrent layers in HAR and therefore additionally evaluate the performance of an altered DeepConvLSTM architecture which has the LSTM completly removed. Figure 2 illustrates the evaluated architecture changes.

3.1. Datasets

We evaluated our architecture changes using five popular HAR datasets, namely the Wetlab (Scholl et al., 2015), the RealWorld HAR (RWHAR) (Sztyler and Stuckenschmidt, 2016), the Smartphone-Based Recognition of Human Activities and Postural Transitions (SBHAR) (Reyes-Ortiz et al., 2016), the Heterogeneity Activity Recognition (HHAR) (Stisen et al., 2015) and the Opportunity (Roggen et al., 2010) dataset. To ensure that our assessment on the necessity of recurrent layers in HAR is not biased toward specific traits of a dataset, we made sure to use a variety of different datasets which differ in the type of activities being performed and circumstances under which the data was recorded. Activity-wise we differentiate between *sporadic, simple/ periodical, transitional* and *complex* activities. *Sporadic* activities, e.g., opening a door, are short in time and do not contain any reoccurring patterns within their execution. Contrarily, *simple/ periodical* activities, e.g., walking, are (usually) longer in time of execution and tend to contain subject-independent, periodical patterns in the sensor data. *Transitional* activities as seen for example in the SBHAR dataset (Reyes-Ortiz et al., 2016) mark the period in-between two activities, e.g., the period from lying down to standing up. Similar to sporadic activities they are (usually) short in time and do not contain any reoccurring patterns within their execution. *Complex* activities, e.g., making a coffee, are activities which are composed of more than one sub-activities, which itself can be either *sporadic, simple/ periodical* or *transitional*. In order to

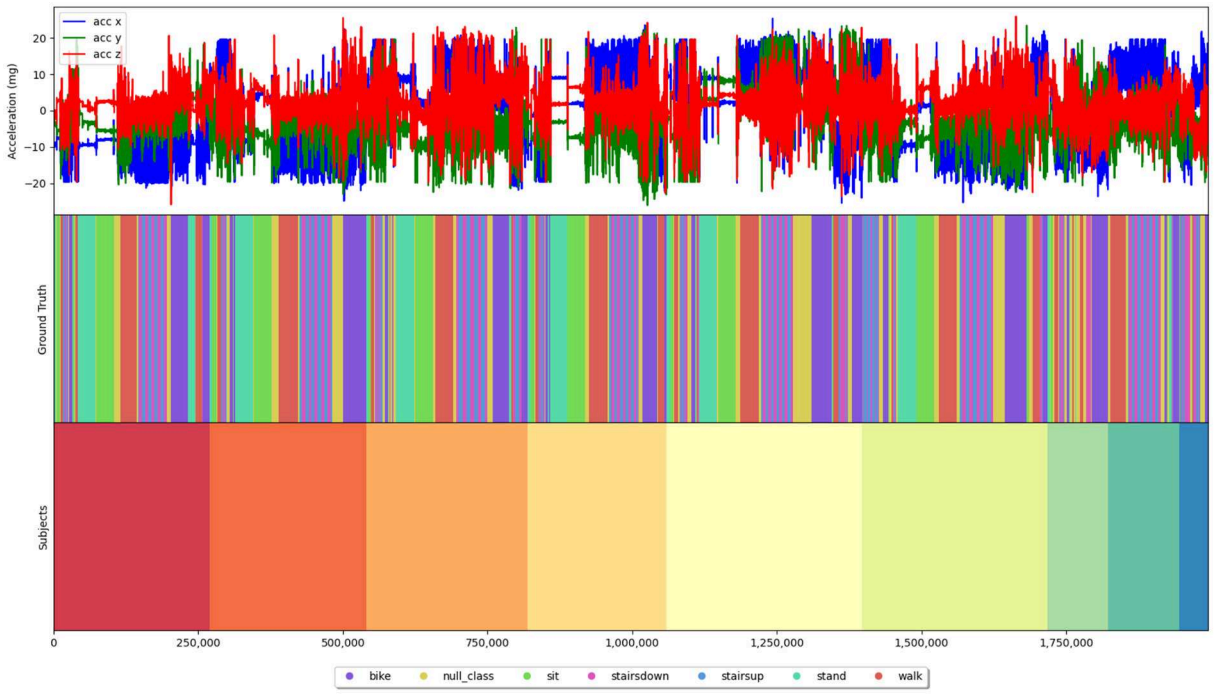

FIGURE 1
Plotted 3D-acceleration data of the HHAR dataset (Stisen et al., 2015) along with color-coded bitmaps indicating which part of the sensor stream belongs to which activity and which subject. This type of plot can be recreated for every dataset using a script we provide in our repository (https://github.com/mariusbock/recurrent_state_of_the_art) and was used as basis for our analysis of each dataset.

further investigate the overall shape of a dataset we created color-coded plots along with the sensor data indicating which parts of a dataset are accredited to which activity and subject (see e.g., Figure 1 for an illustration of the HHAR dataset Stisen et al., 2015). This allows us to better analyze and judge the degree of repetition, rapid changes and distribution of data among subjects within each dataset. We did not include all plots within this article, but advise readers to recreate the plots themselves using a script we provide in our repository[1] as it allows to better zoom in and out without loss of readability.

For the Wetlab (Scholl et al., 2015), RWHAR (Sztyler and Stuckenschmidt, 2016), SBHAR (Reyes-Ortiz et al., 2016), and HHAR dataset (Stisen et al., 2015) we limited ourselves to only use 3D-acceleration data of a sensor—if possible—located on the right wrist of the subject. For the Opportunity dataset (Roggen et al., 2010) many sensors exhibited packet loss. We therefore chose to use the same sensors as Ordóñez and Roggen (2016) to reduce the effect of missing values. In the following, each dataset, its type of activities and limitations will be explained in more detail.

1 https://github.com/mariusbock/recurrent_state_of_the_art

3.1.1. Wetlab

The Wetlab dataset (Scholl et al., 2015) consists of 22 subjects performing two DNA extraction experiments within a wetlab environment and provides two types of annotations namely *tasks* and *actions*. While the former are specific steps within the experimental protocol, the latter represents underlying activities. As the experiments followed an experimental protocol, recording sessions of different subjects contain almost identical sequences of consecutive activities. Nevertheless, it should also be noted that some activities were not performed for some subjects as not all steps in the protocol were not mandatory and were therefore sometimes skipped by subjects. Within our experiments we predicted the annotated actions which left us with 8 different classes, namely *cutting, inverting, peeling, pestling, pipetting, pouring, stirring* and *transfer* as well as a *null* class. In total the dataset consists of 18 h of collected data. As each subject performed the experiment in their own time, execution times per subject differ and average execution times per activity vary greatly, lasting between half a second up to three and a half minutes. Furthermore, the class distribution is imbalanced with the *null* class having almost ten-times as many instances as the longest recorded activity (*pestling*). While performing the activities 3D-accelerometer data of the dominant wrist of each subject was recorded at 50*Hz*. Activities contained in the Wetlab dataset can

be classified as both *simple/ periodical* (e.g., *stirring*) and *complex* activities (e.g., *cutting*).

3.1.2. RWHAR

The RWHAR dataset consists of 15 subjects performing a set of 8 different *simple/ periodical* activities (Sztyler and Stuckenschmidt, 2016). The activities are *climbing_down, climbing_up, jumping, lying, standing, sitting, running* and *walking*. The dataset was collected in a real-world scenario, i.e., with participants not performing the activities in a controlled lab environment, but at different locations like their own home or in public places. In total the dataset consists of roughly 18 h of data (1 h per subject), with activities not always being recorded in the same order for each subject. Due to the fact that each activity was recorded in sessions, i.e., having participants perform a single activity for a certain amount of time, the average activity lengths (except for the activity *jumping, climbing_down* and *climbing_up*) are around 10 min without any recurrences and rapid changes of activities. Though the dataset offers many different sensors, we chose to use 3D-acceleration data captured by a sensor attached to the right wrist, which was set to sample at 50 *Hz*.

3.1.3. SBHAR

The SBHAR dataset consists of 30 subjects performing 6 activities of daily living (*standing, sitting, lying, walking, walking downstairs walking upstairs*) (Reyes-Ortiz et al., 2016). In addition to the 6 simple/ periodical activities, 6 transitional activities (*stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie* and *lie-to-stand*) along with a *null* class are also annotated in the data. The dataset contains roughly 6 h of recorded data (between 10 and 15 min per subject) and was collected in a controlled lab environment. During experiments subjects had to follow an experimental protocol which defined the order and length in which activities were to be performed. Labels are therefore fairly evenly distributed among the 6 simple/ periodical and 6 transitional activities, but are dominated by the (Sztyler and Stuckenschmidt, 2016), the SBHAR dataset (Reyes-Ortiz et al., 2016) has participants more rapidly change activities with the average execution time of the *periodical/ simple* activities being around 10–20 s and the transitional activities being around 2–5 s. Additionally, activities were executed multiple times and thus reooccur within each subject's data. Note that the smartphone, which was used for data collection sampling at 50*Hz*, was attached to the waist of the subjects.

3.1.4. HHAR

The HHAR dataset consists of 9 subjects performing 6 activities of daily living (*biking, sitting, standing, walking, walking upstairs* and *walking downstairs*) (Stisen et al., 2015). During experiments subjects were asked to perform each *simple/ periodical* activity for 5 min following a specific activity sequence, which also makes the class distribution fairly balanced amongst all classes. In total the dataset contains 5 h and 30 min of recorded data with the data being evenly distributed among the first 6 subjects (between 45 and 60 min). For subjects 7, 8, and 9 significantly less data is recorded (17, 20 and 8 min). Similar to the SBHAR dataset (Reyes-Ortiz et al., 2016), activities within the HHAR dataset (Stisen et al., 2015) were not recorded as a whole, but included breaks, i.e., periods of null class, and changes. Nevertheless, on average activities were executed for longer periods being around 20–30 s for the activites *walking upstairs* and *walking downstairs* and the *null* class and around 90–120 s for all other activites. The dataset was recorded in a controlled, real-world scenario with each activity being executed in two environments and routes. For our experiments we use 3D-acceleration data recorded by smartwatches worn by each subject. Unfortunately, the dataset does not state whether sensors were worn on the same wrist for all subjects. Furthermore, due to the fact that two types of smartwatches were used throughout experiments, sampling rates differed across experiments (100 or 200*Hz*). We therefore downsampled the data of the higher sampling smartwatches to be 100*Hz* as well. Additionally, we omitted faulty recording sessions which had only one sample.

3.1.5. Opportunity

The Opportunity dataset (Roggen et al., 2010) consists of 4 subjects performing activities of daily living. Similar to the Wetlab dataset (Roggen et al., 2010), the Opportunity dataset (Scholl et al., 2015) provides two types of annotations namely *modes of locomotion* and *gestures*. We used the latter during our experiments which left us with 18 classes which needed to be predicted. The activities were *opening* and *closing door 1 and 2, fridge, dishwasher* and *drawers 1, 2 and 3, cleaning table, drinking from cup* and *toggling switch* as well as a *null* class. Each subject performed 6 different recording sessions. 5 of these recording sessions were non-scripted (ADL runs), while one required subjects to repeat a sequence of the relevant 17 activities 20 times (Drill runs). The Opportunity dataset was recorded in an controlled, experimental environment. Roggen et al. (2010) state that during the ADL sessions, subjects were asked to follow a higher level protocol, but were allowed to interleave their actions along the way. In total the dataset contains 8 h of recorded activity time with roughly 2 h per subject. For each activity there is around 1–3 min of data per subject. Only the *null* class (1:30 h) and the activity *drinking from cup* (between 5 and 10 min) were recorded longer per subject. Average execution times are short, given that the dataset contains almost solely sporadic activities, being around 2–6 s per activity. Similar to the SBHAR and HHAR dataset, both session types had participants frequently switch between activities, therefore causing activities to reoccur

within each recording session. As the body-worn sensor streams contain missing values due to packet loss, we decided to use the same set of sensor channels as Ordóñez and Roggen (2016) which left us with a dataframe consisting of in total 113 feature channels, each representing an individual sensor axis sampled at 30*Hz*. Unlike Ordóñez and Roggen (2016) we do not apply the train-test split as defined by the Opportunity challenge, but split the data subject-wise in order to perform Leave-One-Subject-Out (LOSO) cross-validation. Gestures contained in the Opportunity dataset can all be classified as *sporadic* activities, except for the activity *cleaning table* which can also be seen as a *simple/ periodical* activity.

3.2. Training

We chose to use the popular DeepConvLSTM architecture as introduced by Ordóñez and Roggen (2016) to base our assessment on. By combining both convolutional and recurrent layers, Ordóñez and Roggen (2016) claim that the network is able to automatically extract discriminative features and model temporal dependencies. We argue that the latter might not be necessary depending on the nature of the activities one tries to predict. We therefore compare predictive performance of the original DeepConvLSTM architecture against two variations of it. The first variation only features a 1-layered LSTM instead of a 2-layered LSTM. We already demonstrated that said change increases the overall predictive performance of the network (Bock et al., 2021). In addition to the 1-layered variant of the network, we now go one step further by completely removing the LSTM from the network, consequently cutting out all recurrent layers in the network. For the two architectures which include LSTM layers, we additionally evaluate settings where we vary the amount of hidden units employed in each layer, i.e., 128, 256, 512 and 1,024, which leaves us with 9 different architectures to be evaluated. Figure 2 illustrates the original DeepConvLSTM as well the two altered versions of the architecture.

In general, we only modify the LSTM within the DeepConvLSTM architecture and other architecture specifications, i.e., dropout rate (0.5), number of convolution layers (4) and kernels per layer (64), are left unchanged to the original architecture (Ordóñez and Roggen, 2016). To minimize the effect of statistical variance, we train each architecture variation 5 times on each dataset, each time employing a different random seed drawn from a predefined set of 5 random seeds. We calculate the predictive performance of each architecture variation as the average performance results of said 5 runs. We chose to use the same hyperparameters as illustrated in Bock et al. (2021) which were obtained by evaluating multiple settings on the Wetlab dataset. As with our previous analysis (Bock et al., 2021) we again did not perform any hypertuning specific to the other datasets as we argue that our analysis is solely focused on the architectural changes and their influence on the predictive performance. We employed a sliding window approach with a window size of 1 s and an overlap of 60%. Furthermore we used the *Adam* optimizer with a weight decay of $1e^{-6}$ and a constant learning rate of $1e^{-4}$ and initialized weights using the Glorot initialization (Glorot and Bengio, 2010). To enable the network to better learn imbalanced class distributions, e.g., as seen in the Wetlab dataset (Scholl et al., 2015), we employed a weighted cross-entropy loss with the weights being set relative to the support of the class. Since the datasets were recorded using different sampling rates we made sure to keep the relation between convolutional filter size and sliding window size consistent across datasets. We therefore set the filter size to be 11 for the Wetlab (Scholl et al., 2015), RWHAR (Sztyler and Stuckenschmidt, 2016), and SBHAR (Reyes-Ortiz et al., 2016), 7 for the Opportunity (Roggen et al., 2010) and 21 for the HHAR dataset (Stisen et al., 2015). That way for each dataset windows and filters are always capturing and analyzing the same amount of temporal information. Lastly, each training session was set to run 30 epochs long.

4. Results

For all datasets results were obtained using a LOSO cross-validation. Within this validation method each subject's data becomes the validation set exactly once while all the other subjects' data are used for training. For example, in case of the Opportunity dataset (Roggen et al., 2010) each architectural variation would be trained 4 times as the dataset contains sensor data of 4 subjects. The final prediction performance is then the average across all subjects. Using LOSO cross-validation ensures that results are not a product of overfitting on subject-specific traits as the validation data is always from an unseen subject, whose specific ways of performing the set of activities has not been seen during training. Unlike Bock et al. (2021), we used the epoch which resulted in the highest average F1-score for calculation of the global average. Furthermore, in addition to reporting results on the full Opportunity dataset (Roggen et al., 2010), we also report prediction results obtained solely on the ADL as well as the Drill sessions of the dataset.

Looking at the results of the convolutional architectural variant vs. the 1- or 2-layered LSTM variants, one can see that there are major differences in the performance depending which dataset was used as input. On average, the architecture variant which employs a 1-layered LSTM with 1,024 hidden units delivers the best prediction results. Figure 3 summarizes the average performance difference in F1-score between the 1-layered and 2-layered variants, the 1-layered variants and no LSTM variant and the 2-layered variants and no LSTM variant. For the Wetlab (Scholl et al., 2015), RWHAR (Sztyler and Stuckenschmidt, 2016), SBHAR (Reyes-Ortiz et al., 2016), and HHAR dataset (Stisen et al., 2015) the 1-layered variants on average outperform all other architectural variants. Only

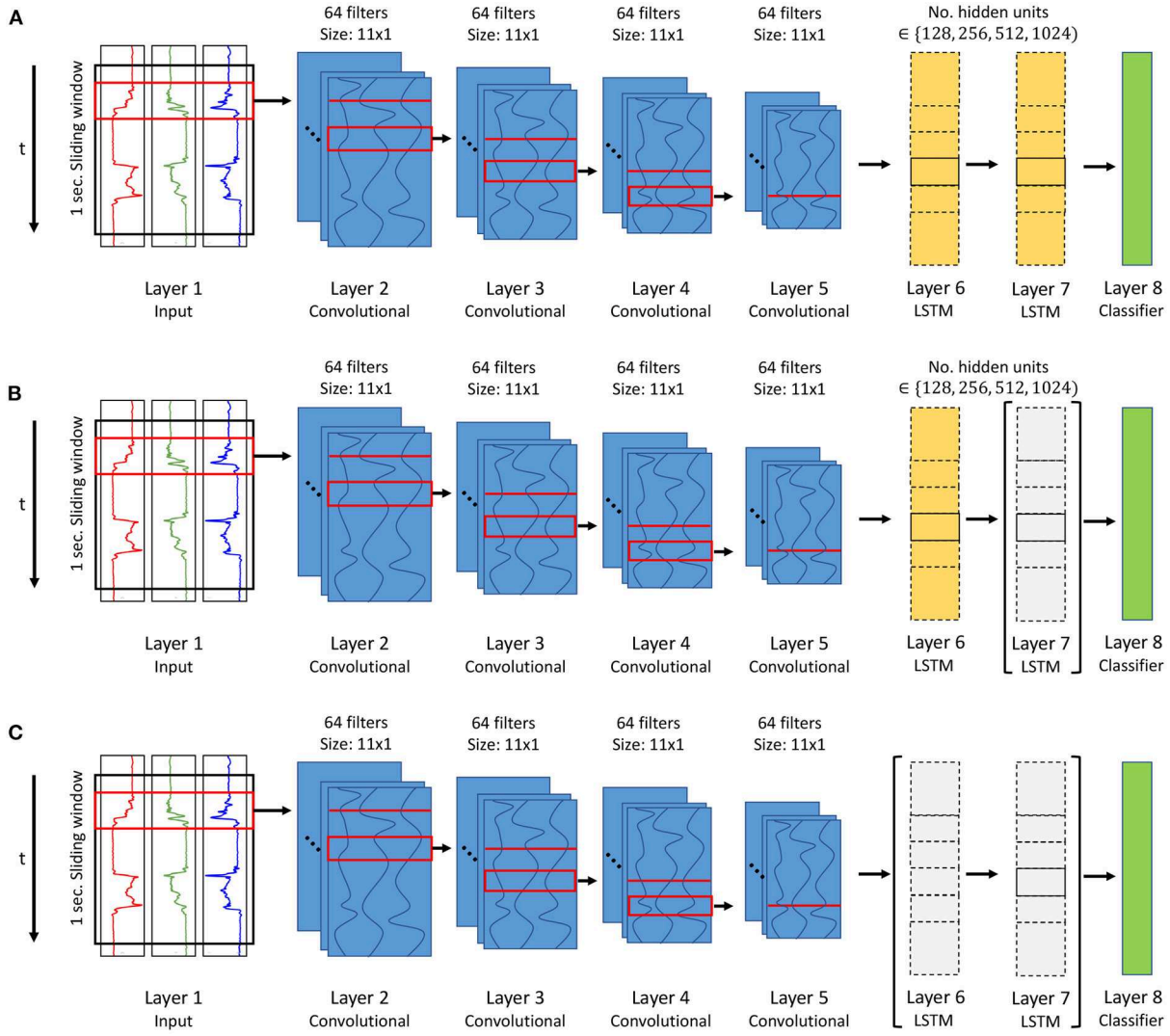

FIGURE 2
Illustration of the 3 evaluated architectures, namely **(A)** the original DeepConvLSTM architecture (Ordóñez and Roggen, 2016), **(B)** an altered version with a 1-layered LSTM (Layer 7 removed) (Bock et al., 2021) and **(C)** one with all LSTM layers (Layer 6 and 7) being removed. For **(A,B)** we vary the amount of hidden units within the LSTM layers to be either 128, 256, 512, or 1,024.

for the Opportunity dataset (Roggen et al., 2010), as well as its two sessions types individually, the 2-layered variants on average achieve a higher F1-score than all other variants. Table 1 outlines the average accuracy, precision, recall and F1-score per architecture variant for each dataset.

As previously mentioned, each dataset contains different types of activities. Analyzing results therefore on a per-class level (see Tables 2–8) reveals which type of network might be a more suited for predicting which type of activity. In their works, Ordóñez and Roggen (2016) and Hammerla et al. (2016) claim that convolutional layers are better at modeling local patterns on a sensor level, while recurrent layers are better at modeling (global) temporal patterns, e.g., of previously occurring activities. One could therefore conclude that in our experiments the convolutional network would have performed best at predicting simple/ periodical activities. Nevertheless, we could only partly exhibit this trend as the convolutional network performed best only for some simple/ periodical activities. We accredit the increased performance of the LSTM-based implementations on the simple/ periodical activities to two factors:

1. Convolutional networks are less efficient in predicting datasets with short activity execution times or rapid changes between activities.

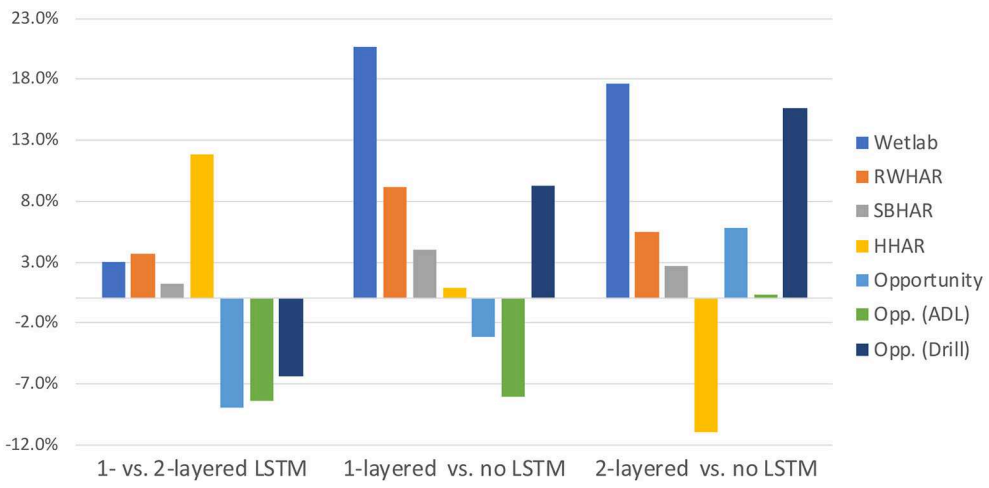

FIGURE 3
Average F1-score performance differences between 1-layered and 2-layered variants (1 vs. 2), 1-layered and no LSTM variants (1 vs. 0) and 2-layered and no LSTM variants (0 vs. 2) of the DeepConvLSTM architecture (Ordóñez and Roggen, 2016). A positive performance difference equates to the former outperforming the latter type of architecture. A negative performance difference equates to the latter outperforming the former type of architecture. For example a positive performance difference between 1 vs. 2 means that on average 1-layered outperformed 2-layered architectures. Contrarily a negative performance difference between 1 vs. 2 means that on average 2-layered outperformed 1-layered architectures. Reported results are averaged across 5 runs using a set of 5 varying seeds and either the Wetlab (Scholl et al., 2015), RWHAR (Sztyler and Stuckenschmidt, 2016), SBHAR (Reyes-Ortiz et al., 2016), HHAR (Stisen et al., 2015) or Opportunity dataset (Roggen et al., 2010) as input. For the Opportunity dataset results are also reported on the ADL and Drill sessions individually.

2. All datasets were recorded following some experimental protocol which consequently introduced some form of temporal schedule amongst activities.

The first factor can particularly be seen comparing results obtained on the RWHAR (Sztyler and Stuckenschmidt, 2016), SBHAR (Reyes-Ortiz et al., 2016), and HHAR dataset (Stisen et al., 2015). Even though said datasets have an overlap in activities, i.e., *sitting, standing, walking, walking_downstairs* and *walking_upstairs*, prediction performance on these activities is significantly worse for the SBHAR dataset (Reyes-Ortiz et al., 2016). Looking at the average execution time of each activity, one can see that they are shorter within the SBHAR dataset (Reyes-Ortiz et al., 2016) compared to the other two datasets, which also causes more rapid changes between activities. At the time of writing this article, we cannot clearly identify a reason to explain this behavior. In fact, as we are using a sliding window, the duration of the activity does not influence what type of data a network sees. Nevertheless, we hypothesize that rapid changes between activities can cause more windows to consist of multiple activity labels. Given that the label of a window is determined by the label of the last sample within said window, as proposed by Ordóñez and Roggen (2016), a shorter average activity execution time can cause windows to be more likely to contain labels of multiple activites. In turn, a convolutional kernel could falsely attribute reoccurring patterns of other activities to the target activity. On the contrary, this exhibited trend could also be something specific to the SBHAR dataset (Reyes-Ortiz et al., 2016) and thus requires further investigation on a larger scale.

The second factor is a phenomenon which comes naturally at the cost of recording datasets in a non-natural environment or pre-scripted recording session. All datasets which were part of our analysis were recorded under experimental conditions. This involved either a predefined length of execution time, repetitions of activities or a set of steps in a predefined workflow. As already mentioned, this consequently introduces some form of temporal schedule amongst activities which can be modeled by a LSTM. Networks which include recurrent layers were thus not only able to use information on preceding activities, but also learn the overall sequence in which the activities were recorded. This type of information could especially help networks differentiate between activities which are very similar in the patterns they consist of. For example *walking, walking down* or *walking up stairs* were more reliably differentiated by networks which contained recurrent layers (see results obtained on the SBHAR, HHAR and RWHAR dataset).

Whether to employ a 1-layered or a 2-layered LSTM seems to come down to how much emphasis one wants to put on these temporal information compared to local patterns identified by the convolutional layers. This can be especially seen in the results obtained on the Drill compared to the ADL sessions of the Opportunity dataset. Former had subjects follow a strict experimental protocol with many repetitions of the same activity making temporal relations among them more strong while also introducing an overall sequence in which activities were performed. Latter had subjects not adhere to a strict experimental protocol and had them act naturally which caused temporal relationships amongst activities to be less

TABLE 1 Average cross-participant results employing a removed, 1-layered, 2-layered LSTM within the DeepConvLSTM architecture (Ordóñez and Roggen, 2016) with varying hidden units (i.e., 128, 256, 512 or 1,024) across 5 runs with a set of 5 varying seeds for the Wetlab, RWHAR, SBHAR HHAR and Opportunity dataset (Roggen et al., 2010; Scholl et al., 2015; Stisen et al., 2015; Reyes-Ortiz et al., 2016; Sztyler and Stuckenschmidt, 2016).

Layers	Units	Metric	Wetlab	RWHAR	SBHAR	HHAR	Opportunity	Opp. (ADL)	Opp. (Drill)
0		Accuracy	19.12%	57.34%	40.77%	39.12%	24.05%	17.21%	29.26%
		Precision	35.20%	70.57%	55.28%	53.21%	41.29%	29.07%	47.76%
		Recall	28.18%	70.61%	55.94%	52.21%	36.85%	30.87%	42.80%
		F1	26.00%	68.77%	52.34%	50.25%	35.68%	26.48%	40.56%
1	128	Accuracy	33.23%	65.92%	59.49%	40.00%	17.76%	7.18%	33.23%
		Precision	45.66%	79.31%	71.68%	53.99%	38.04%	15.94%	45.66%
		Recall	52.48%	77.74%	75.68%	53.30%	25.70%	10.70%	41.74%
		F1	44.53%	75.97%	70.89%	50.86%	25.79%	9.73%	44.53%
	256	Accuracy	34.66%	67.84%	61.21%	**40.29%**	21.98%	11.48%	36.86%
		Precision	46.97%	80.60%	73.30%	55.31%	41.44%	25.37%	54.33%
		Recall	**53.52%**	79.36%	76.83%	54.07%	31.81%	18.81%	52.74%
		F1	46.62%	77.72%	72.40%	51.31%	31.73%	16.81%	49.04%
	512	Accuracy	**35.25%**	67.97%	62.09%	39.34%	24.33%	14.99%	39.51%
		Precision	47.68%	80.75%	74.10%	53.61%	42.86%	29.85%	57.55%
		Recall	53.11%	79.34%	77.78%	53.52%	36.03%	25.03%	54.76%
		F1	47.26%	78.29%	73.31%	50.47%	35.01%	22.36%	51.84%
	1,024	Accuracy	35.12%	**70.31%**	63.22%	40.71%	26.18%	16.64%	41.15%
		Precision	**48.04%**	**82.11%**	74.79%	**55.37%**	45.27%	30.98%	59.40%
		Recall	51.98%	**81.10%**	78.64%	**54.28%**	38.11%	29.49%	56.64%
		F1	**48.25%**	**79.79%**	74.30%	**51.80%**	37.49%	24.87%	53.68%
2	128	Accuracy	29.98%	60.85%	54.76%	25.35%	22.70%	12.45%	36.61%
		Precision	41.74%	74.62%	68.14%	39.19%	35.48%	20.60%	49.12%
		Recall	48.53%	73.01%	72.04%	38.96%	42.11%	29.06%	58.37%
		F1	41.06%	71.25%	66.97%	35.39%	34.63%	19.53%	50.42%
	256	Accuracy	31.70%	60.42%	58.59%	25.67%	26.23%	15.33%	40.52%
		Precision	43.67%	74.18%	71.52%	40.56%	39.60%	24.99%	53.74%
		Recall	49.55%	72.81%	75.02%	39.34%	46.68%	36.42%	60.55%
		F1	43.25%	70.87%	70.37%	36.15%	38.85%	24.17%	54.29%
	512	Accuracy	32.86%	67.11%	61.80%	29.16%	30.85%	19.69%	43.74%
		Precision	45.17%	79.71%	73.94%	43.41%	43.88%	29.66%	58.05%
		Recall	50.74%	78.82%	77.76%	42.71%	52.65%	43.19%	62.89%
		F1	44.53%	77.13%	73.24%	39.50%	44.55%	30.49%	57.53%
	1,024	Accuracy	33.74%	68.07%	**64.31%**	35.02%	**33.56%**	**21.52%**	**48.72%**
		Precision	46.48%	80.26%	**75.54%**	49.64%	**46.62%**	**30.89%**	**62.50%**
		Recall	51.53%	79.56%	**79.60%**	49.19%	**56.35%**	**46.33%**	**67.87%**
		F1	45.57%	77.85%	**75.37%**	45.95%	**47.91%**	**32.97%**	**62.72%**

For the Opportunity dataset (Roggen et al., 2010) results are also reported on the ADL and Drill sessions individually. Written in bold font are the best results per dataset and evaluation metric.

prominent. Therefore, networks employing a 2-layered LSTM did not perform notably better than convolutional networks on the ADL sessions, while for the Drill sessions performance differences were as much as 22% in F1-score in favor of the 2-layered variants.

On the contrary, in order to correctly identify sporadic and transitional activities as seen for example in the Opportunity (Roggen et al., 2010) or SBHAR dataset (Reyes-Ortiz et al., 2016) recurrent layers seem to be necessary. We accredit this to the fact that these activities are short in time and do not show periodical, local patterns which could be identified by a convolution kernel. Moreover, similar to what Hammerla et al. (2016) concluded, transitional as well as sporadic activities like *opening* and *closing a door*, have temporal relationships with other preceding

TABLE 2 Per class results for all 9 architectural variations of the DeepConvLSTM architecture (Ordóñez and Roggen, 2016) using the Wetlab dataset (Scholl et al., 2015) as input.

Layers	Units	Metric	null	cutting	inverting	peeling	pestling	pipetting	pouring	stirring	transfer
0		Accuracy	**73.79%**	13.89%	15.89%	0.00%	26.03%	0.93%	3.30%	22.36%	0.00%
		Precision	82.24%	**62.34%**	38.81%	0.00%	34.32%	15.59%	13.13%	34.09%	0.00%
		Recall	**87.89%**	15.35%	26.24%	0.00%	60.73%	1.04%	6.01%	45.21%	0.00%
		F1	**84.85%**	23.46%	25.56%	0.00%	40.18%	1.80%	6.12%	34.64%	0.00%
1	128	Accuracy	72.50%	40.93%	33.66%	11.83%	39.19%	**23.95%**	**10.53%**	**29.02%**	5.82%
		Precision	**89.62%**	55.45%	46.52%	26.01%	50.21%	35.46%	16.35%	**43.70%**	13.32%
		Recall	79.40%	62.00%	57.56%	18.38%	**71.03%**	**47.14%**	**31.83%**	52.97%	11.94%
		F1	83.87%	55.43%	44.94%	18.05%	54.57%	37.30%	17.56%	**42.70%**	9.50%
	256	Accuracy	72.55%	44.61%	**36.44%**	15.24%	40.73%	22.90%	10.17%	28.09%	7.29%
		Precision	89.09%	58.51%	**49.07%**	28.77%	54.11%	33.71%	15.50%	41.93%	14.09%
		Recall	79.88%	65.62%	**59.62%**	26.72%	70.27%	45.15%	27.42%	52.24%	14.89%
		F1	83.98%	59.42%	**48.71%**	24.03%	56.53%	35.98%	17.68%	42.27%	12.01%
	512	Accuracy	73.20%	46.19%	35.72%	**17.50%**	41.31%	23.02%	9.77%	27.64%	**8.31%**
		Precision	88.75%	59.45%	47.39%	33.03%	54.98%	34.74%	15.44%	41.59%	**15.16%**
		Recall	80.91%	66.38%	55.87%	**28.84%**	69.12%	43.76%	26.90%	**53.04%**	**17.53%**
		F1	84.44%	60.98%	47.94%	**27.06%**	57.00%	36.23%	17.21%	41.60%	**13.59%**
	1,024	Accuracy	73.39%	**46.99%**	32.81%	16.47%	**42.24%**	23.66%	10.29%	27.11%	7.84%
		Precision	88.56%	58.85%	46.18%	**34.33%**	55.17%	**35.62%**	**17.58%**	41.23%	13.59%
		Recall	82.76%	**66.66%**	52.11%	28.20%	67.72%	42.41%	24.01%	48.98%	16.32%
		F1	83.99%	**62.66%**	46.78%	25.34%	**60.90%**	**38.60%**	**19.48%**	**42.70%**	**13.59%**
2	128	Accuracy	69.59%	38.16%	30.03%	9.64%	35.04%	19.71%	7.70%	22.19%	4.73%
		Precision	88.48%	55.11%	41.49%	20.91%	45.88%	28.86%	11.42%	38.11%	11.88%
		Recall	76.80%	56.26%	54.80%	14.92%	66.57%	43.56%	26.81%	40.66%	9.84%
		F1	81.92%	52.63%	41.41%	14.99%	50.49%	32.03%	13.96%	34.86%	8.16%
	256	Accuracy	70.63%	40.03%	29.19%	16.28%	37.00%	20.20%	8.55%	23.19%	5.71%
		Precision	87.87%	54.90%	41.06%	29.56%	49.17%	29.71%	13.43%	38.84%	10.22%
		Recall	78.52%	58.99%	51.37%	25.80%	64.93%	42.56%	24.67%	43.50%	12.92%
		F1	82.70%	54.18%	40.49%	25.18%	52.46%	32.50%	15.26%	36.35%	9.58%
	512	Accuracy	71.17%	42.75%	32.23%	15.88%	38.48%	20.70%	8.82%	25.02%	5.41%
		Precision	88.23%	55.62%	44.34%	30.89%	53.10%	29.45%	14.54%	39.21%	10.70%
		Recall	78.93%	63.50%	56.19%	24.71%	64.63%	43.85%	23.38%	47.55%	11.54%
		F1	83.03%	57.23%	43.97%	24.66%	54.11%	33.06%	15.65%	38.34%	8.85%
	1,024	Accuracy	72.26%	44.48%	31.24%	17.38%	40.17%	21.24%	9.39%	26.22%	6.88%
		Precision	88.07%	57.63%	43.25%	32.87%	**56.00%**	31.35%	16.16%	41.07%	13.02%
		Recall	80.38%	66.52%	53.42%	28.25%	60.06%	42.48%	23.59%	47.58%	13.77%
		F1	83.75%	59.45%	42.78%	26.92%	55.90%	33.72%	16.48%	39.54%	11.28%

Results are averaged across 5 runs using a set of 5 different seeds and LOSO cross-validation. Written in bold font are the best results per activity and evaluation metric.

activities and can therefore be identified using the overall temporal context. We also identify this as the reason why the convolutional network is not outperforming the architectural variant with a 2-layered LSTM for the ADL sessions of the Opportunity dataset, as the dataset almost exclusively sporadic activities. Additionally, as for the simple/ periodical activities, rapid changes in activities might cause mixed sliding windows, which contain records of multiple activities. A convolutional network could thus struggle in identifying local patterns belonging to the target activity as it would more frequently see patterns belonging to other activities "incorrectly" labeled as the target activity. Nevertheless, as previously mentioned this hypothesis requires further investigation.

As previously mentioned complex activities are composed of multiple sub-activities which itself can be sporadic, transitional, or short-term periodical. In our experiments only the Wetlab dataset (Scholl et al., 2015) contains complex activities (e.g., *peeling*). Given that participants in the Wetlab dataset

TABLE 3 Per class results for all 9 architectural variations of the DeepConvLSTM architecture (Ordóñez and Roggen, 2016) using the RWHAR dataset (Sztyler and Stuckenschmidt, 2016) as input.

Layers	Units	Metric	climbing_down	climbing_up	jumping	lying	running	sitting	standing	walking
0		Accuracy	43.96%	21.85%	79.28%	63.86%	82.41%	58.02%	60.51%	48.81%
		Precision	59.65%	47.60%	85.78%	71.69%	**97.11%**	68.47%	72.45%	61.78%
		Recall	62.21%	28.98%	**91.81%**	73.89%	84.84%	75.80%	78.70%	69.68%
		F1	59.61%	34.17%	87.91%	71.19%	89.18%	70.07%	73.30%	64.69%
1	128	Accuracy	67.45%	39.22%	81.29%	69.78%	82.74%	60.15%	64.34%	62.35%
		Precision	80.74%	59.54%	92.04%	80.60%	95.43%	72.87%	74.30%	78.96%
		Recall	78.61%	54.32%	86.24%	79.53%	87.02%	76.75%	82.52%	76.91%
		F1	78.39%	52.63%	86.62%	77.83%	89.55%	71.29%	76.08%	75.33%
	256	Accuracy	72.09%	46.41%	80.05%	67.68%	81.55%	64.21%	65.24%	65.49%
		Precision	81.73%	65.90%	91.25%	78.86%	93.82%	75.18%	74.24%	83.84%
		Recall	83.29%	61.38%	86.56%	75.36%	86.93%	80.79%	**83.23%**	77.38%
		F1	81.77%	59.98%	86.40%	75.17%	88.66%	**75.31%**	76.52%	77.91%
	512	Accuracy	69.84%	45.77%	80.50%	70.17%	**83.12%**	61.97%	64.94%	67.42%
		Precision	82.22%	66.74%	88.67%	81.75%	95.16%	72.60%	75.56%	83.29%
		Recall	81.22%	58.82%	87.24%	80.08%	**87.39%**	78.28%	81.87%	79.82%
		F1	80.13%	61.35%	87.13%	79.19%	**90.92%**	72.85%	75.45%	79.34%
	1,024	Accuracy	**74.19%**	**51.56%**	**82.93%**	68.61%	81.61%	**65.15%**	**67.21%**	**71.24%**
		Precision	**84.09%**	**70.70%**	**91.53%**	80.69%	93.18%	74.12%	**77.60%**	**84.99%**
		Recall	**85.00%**	63.69%	88.82%	77.83%	87.34%	**80.98%**	83.02%	**82.11%**
		F1	**83.82%**	**64.73%**	**88.70%**	76.65%	88.57%	75.17%	**78.39%**	**82.28%**
2	128	Accuracy	53.65%	31.01%	71.20%	68.96%	81.20%	60.61%	65.59%	54.57%
		Precision	74.19%	47.74%	82.00%	78.27%	93.80%	75.47%	75.98%	69.54%
		Recall	62.75%	45.86%	77.71%	81.51%	86.73%	75.77%	81.47%	72.25%
		F1	65.18%	43.30%	77.89%	77.88%	88.54%	72.30%	76.51%	68.41%
	256	Accuracy	54.26%	33.09%	71.18%	67.50%	81.19%	58.50%	64.72%	52.95%
		Precision	72.88%	49.41%	81.71%	77.85%	93.38%	72.03%	74.60%	71.61%
		Recall	65.51%	50.18%	76.75%	79.37%	86.91%	73.68%	81.48%	68.61%
		F1	66.01%	45.68%	77.88%	76.54%	88.46%	69.51%	75.90%	66.97%
	512	Accuracy	69.58%	43.57%	77.96%	**72.05%**	80.26%	63.71%	64.32%	65.43%
		Precision	80.23%	62.97%	87.84%	**82.25%**	91.67%	**75.96%**	75.27%	81.53%
		Recall	81.04%	57.77%	85.17%	**81.71%**	87.13%	78.00%	81.14%	78.60%
		F1	79.65%	56.86%	84.39%	**80.63%**	87.69%	74.08%	75.86%	77.86%
	1,024	Accuracy	72.03%	48.10%	79.31%	68.29%	82.06%	62.30%	65.44%	67.06%
		Precision	83.16%	64.64%	88.02%	80.13%	94.36%	73.35%	74.97%	83.44%
		Recall	82.15%	**64.49%**	86.11%	77.51%	86.85%	78.50%	82.68%	78.18%
		F1	81.94%	61.29%	85.89%	76.60%	88.89%	72.81%	76.70%	78.64%

Results are averaged across 5 runs using a set of 5 different seeds and LOSO cross-validation. Written in bold font are the best results per activity and evaluation metric.

(Scholl et al., 2015) were asked to perform an experiment which required them to (mostly) follow a step-by-step guide, architectures employing recurrent layers are able to model said sequence and overall achieve better prediction results. Nevertheless, as the complex activities also inherit periodical, local patterns (e.g., *stirring*), employing a 1-layered LSTM seemed to combine information provided by both the recurrent and convolutional layers in the most efficient way and resulted in the highest prediction results. Nevertheless, we deem complex activities to be a case-by-case decision. Whether they are better to be predicted using a convolutional or recurrent network depends on the use case and the type of sub-activities they consist of.

With a few exceptions, performance steadily increases with an increased amount of hidden units for both the 1- and the 2-layered architectural variants. Looking at the generalization gaps within each experiment, i.e., the differences between training and validation performance, we see that generalization

TABLE 4 Per class results for all 9 architectural variations of the DeepConvLSTM architecture (Ordóñez and Roggen, 2016) using the SBHAR dataset (Reyes-Ortiz et al., 2016) as input.

Layers	Units	Metric	null	walking	upstairs	downstairs	sitting	standing	lying	stand-to-sit	sit-to-stand	sit-to-lie	lie-to-sit	stand-to-lie	lie-to-stand
0		Accuracy	40.31%	50.29%	65.12%	74.05%	58.81%	53.32%	85.52%	12.14%	18.48%	27.36%	12.76%	8.93%	22.85%
		Precision	75.24%	57.87%	73.92%	86.10%	86.07%	60.61%	93.38%	22.02%	36.02%	34.92%	32.01%	29.51%	30.98%
		Recall	47.09%	78.65%	87.01%	85.81%	65.06%	80.82%	91.15%	26.14%	30.62%	56.32%	17.91%	11.93%	48.67%
		F1	57.17%	65.31%	77.77%	84.23%	70.13%	68.26%	91.25%	20.88%	30.23%	41.79%	21.04%	15.86%	36.56%
1	128	Accuracy	68.53%	66.05%	80.33%	85.10%	70.56%	72.74%	87.43%	35.05%	45.37%	52.67%	27.22%	49.91%	32.44%
		Precision	**87.71%**	74.98%	83.92%	90.82%	91.54%	82.54%	95.57%	41.80%	58.26%	63.72%	56.62%	64.84%	39.51%
		Recall	76.17%	82.65%	95.34%	93.39%	75.18%	85.25%	91.10%	70.36%	67.93%	71.55%	36.24%	69.75%	68.87%
		F1	81.06%	77.32%	88.28%	91.65%	80.99%	83.16%	92.69%	50.13%	58.72%	65.46%	39.56%	64.94%	47.62%
	256	Accuracy	70.56%	68.07%	82.19%	85.71%	71.53%	73.38%	88.64%	39.21%	47.12%	52.73%	28.46%	53.85%	34.32%
		Precision	88.28%	77.14%	85.48%	91.65%	90.79%	83.85%	95.59%	47.31%	61.09%	63.00%	58.14%	68.32%	42.22%
		Recall	78.19%	83.70%	**95.90%**	93.18%	76.68%	84.89%	92.40%	72.57%	68.05%	72.54%	37.27%	72.20%	71.20%
		F1	82.49%	78.92%	89.41%	92.00%	81.63%	83.69%	93.47%	54.52%	60.70%	65.81%	40.57%	67.98%	50.05%
	512	Accuracy	71.11%	68.28%	82.20%	86.38%	71.30%	73.99%	**89.34%**	41.09%	45.52%	53.88%	32.00%	55.91%	36.17%
		Precision	87.39%	79.08%	86.01%	91.55%	91.60%	84.44%	95.91%	48.86%	58.14%	63.88%	**61.94%**	70.28%	44.25%
		Recall	79.62%	82.72%	95.34%	93.98%	76.17%	85.39%	**92.79%**	74.38%	69.54%	74.94%	41.74%	73.84%	70.74%
		F1	82.86%	79.14%	89.54%	92.42%	81.59%	84.20%	**93.97%**	56.14%	59.45%	67.25%	44.94%	70.04%	51.55%
	1,024	Accuracy	71.59%	69.11%	81.63%	87.36%	72.29%	74.00%	89.25%	42.01%	49.36%	57.01%	33.91%	**56.86%**	37.53%
		Precision	86.85%	81.86%	85.25%	92.34%	**91.70%**	84.11%	96.05%	49.78%	61.29%	**66.83%**	60.19%	70.35%	45.61%
		Recall	80.64%	81.59%	95.55%	**94.32%**	77.06%	85.52%	92.61%	74.29%	71.35%	77.66%	44.85%	**74.85%**	71.99%
		F1	83.18%	80.27%	89.21%	93.02%	82.27%	84.09%	93.92%	56.86%	62.98%	70.05%	46.47%	**70.73%**	52.88%
2	128	Accuracy	64.56%	60.94%	73.30%	79.20%	66.62%	68.60%	81.90%	33.09%	38.20%	45.46%	23.70%	46.20%	30.05%
		Precision	83.70%	75.45%	76.70%	87.37%	90.02%	81.96%	94.23%	41.84%	51.23%	58.67%	48.59%	58.68%	37.39%
		Recall	74.27%	75.85%	95.05%	89.98%	71.70%	79.23%	86.03%	64.43%	62.25%	66.39%	35.23%	69.23%	66.93%
		F1	78.13%	73.73%	83.70%	87.64%	78.02%	79.79%	89.37%	47.73%	52.00%	59.75%	35.12%	61.24%	44.38%
	256	Accuracy	67.51%	64.14%	76.54%	82.97%	67.92%	70.39%	84.34%	38.83%	42.73%	49.77%	28.53%	52.11%	35.91%
		Precision	84.75%	77.72%	79.81%	89.22%	90.83%	82.94%	95.31%	46.32%	55.33%	60.35%	56.95%	66.76%	43.45%
		Recall	77.29%	78.38%	95.53%	92.41%	72.61%	81.63%	87.86%	71.77%	66.41%	72.50%	37.70%	70.60%	70.56%
		F1	80.28%	76.22%	85.91%	90.27%	79.05%	81.41%	90.69%	53.90%	56.22%	63.56%	40.39%	66.20%	50.69%
	512	Accuracy	70.20%	70.30%	81.12%	85.85%	71.84%	73.16%	87.56%	40.73%	47.94%	52.93%	32.50%	53.63%	35.67%
		Precision	85.89%	81.54%	84.81%	91.25%	91.60%	84.77%	96.01%	48.00%	59.44%	63.12%	60.37%	69.58%	44.83%
		Recall	79.80%	84.20%	95.28%	93.29%	76.38%	84.03%	90.90%	74.91%	**71.65%**	74.97%	44.57%	70.75%	70.18%
		F1	82.20%	81.28%	88.97%	91.90%	81.89%	83.52%	92.80%	56.10%	62.01%	66.53%	45.71%	67.86%	51.33%
	1,024	Accuracy	**71.68%**	**73.39%**	**84.30%**	**87.59%**	**72.75%**	**75.20%**	88.70%	**42.99%**	**50.63%**	**57.95%**	**35.66%**	56.20%	**38.97%**
		Precision	86.63%	**83.03%**	**88.75%**	**92.61%**	90.97%	**84.94%**	**96.15%**	**49.90%**	**62.31%**	66.76%	61.57%	**71.08%**	**47.27%**
		Recall	**81.10%**	**86.33%**	94.71%	94.28%	**78.16%**	**86.42%**	91.91%	**75.62%**	71.33%	**80.40%**	**47.57%**	72.32%	**74.71%**
		F1	**83.22%**	**83.76%**	**91.14%**	**93.18%**	**82.72%**	**85.05%**	93.50%	**58.18%**	**64.15%**	**71.38%**	**48.92%**	70.02%	**54.59%**

Results are averaged across 5 runs using a set of 5 different seeds and LOSO cross-validation. Written in bold font are the best results per activity and evaluation metric.

10.3389/fcomp.2022.924954

TABLE 5 Per class results for all 9 architectural variations of the DeepConvLSTM architecture (Ordóñez and Roggen, 2016) using the HHAR dataset (Stisen et al., 2015) as input.

Layers	Units	Metric	null	biking	sitting	standing	walking	stair up	stair down
0		Accuracy	15.48%	**78.83%**	**56.23%**	**47.87%**	26.44%	20.45%	28.56%
		Precision	32.87%	**85.09%**	**67.89%**	**62.75%**	51.20%	37.07%	35.58%
		Recall	24.08%	**90.82%**	**66.49%**	55.60%	33.56%	30.88%	64.07%
		F1	26.49%	**87.44%**	**65.57%**	**57.48%**	38.33%	32.43%	44.01%
1	128	Accuracy	22.82%	75.00%	47.87%	47.62%	38.32%	17.01%	31.33%
		Precision	40.91%	82.34%	58.89%	58.14%	61.46%	35.62%	40.55%
		Recall	35.19%	84.59%	58.36%	58.99%	**46.57%**	27.53%	61.87%
		F1	36.30%	82.50%	56.38%	56.50%	**50.40%**	27.57%	46.35%
	256	Accuracy	24.13%	73.50%	46.04%	45.55%	36.28%	**22.48%**	34.05%
		Precision	**41.11%**	79.38%	59.16%	56.62%	68.61%	**37.45%**	44.83%
		Recall	37.62%	86.28%	56.53%	57.97%	43.36%	**35.46%**	61.26%
		F1	37.87%	81.33%	54.52%	54.46%	47.95%	**33.52%**	49.53%
	512	Accuracy	22.73%	72.61%	42.11%	46.68%	36.21%	20.14%	34.90%
		Precision	38.50%	82.66%	54.42%	57.77%	60.47%	37.30%	44.14%
		Recall	37.31%	83.83%	52.52%	**59.98%**	44.02%	32.39%	64.57%
		F1	35.31%	81.01%	50.51%	56.17%	47.61%	31.33%	**50.35%**
	1,024	Accuracy	22.33%	73.10%	50.11%	45.21%	**39.00%**	18.32%	**36.93%**
		Precision	37.73%	82.56%	61.11%	56.77%	**69.35%**	35.10%	**44.95%**
		Recall	36.53%	84.09%	60.50%	55.35%	45.23%	29.05%	**69.19%**
		F1	35.81%	81.46%	58.81%	54.11%	50.75%	28.83%	**52.82%**
2	128	Accuracy	**24.92%**	53.13%	41.74%	5.25%	20.03%	14.91%	17.47%
		Precision	38.17%	65.58%	57.25%	22.45%	31.54%	26.63%	32.72%
		Recall	43.44%	74.43%	53.34%	5.83%	34.20%	29.34%	32.17%
		F1	**39.00%**	65.69%	52.14%	7.55%	31.03%	24.16%	28.15%
	256	Accuracy	24.62%	54.30%	38.58%	10.73%	17.63%	12.56%	21.29%
		Precision	38.29%	65.98%	56.27%	32.33%	32.43%	24.19%	34.46%
		Recall	**41.57%**	74.64%	50.92%	12.70%	30.20%	21.41%	43.94%
		F1	38.29%	67.16%	49.46%	15.94%	28.36%	20.56%	33.26%
	512	Accuracy	22.00%	62.45%	38.07%	24.31%	20.51%	10.91%	25.87%
		Precision	37.98%	73.47%	52.92%	41.91%	40.27%	23.49%	33.79%
		Recall	35.26%	78.87%	49.42%	31.11%	29.01%	19.01%	56.27%
		F1	35.14%	73.37%	47.46%	32.04%	30.72%	18.15%	39.64%
	1,024	Accuracy	22.14%	65.61%	42.77%	39.43%	29.43%	14.78%	31.04%
		Precision	37.55%	76.22%	57.53%	53.05%	52.72%	30.51%	39.87%
		Recall	36.85%	80.26%	53.98%	50.17%	37.14%	23.70%	62.21%
		F1	35.40%	75.80%	51.73%	48.88%	40.67%	23.25%	45.90%

Results are averaged across 5 runs using a set of 5 different seeds and LOSO cross-validation. Written in bold font are the best results per activity and evaluation metric.

gaps of networks employing recurrent layers strongly correlate with the amount of increased performance one gets from recurrent layers. More specifically, this means that the larger the performance increase one gets from an LSTM, the larger the network's generalization gap will be (and vice versa). We hypothesize that this is accredited to larger LSTMs being more prone to overfit on temporal sequences in which activities were performed in the training data. Consequently, datasets who feature a strict sequence of activities, e.g., the Wetlab dataset, overfitting on such temporal patterns generally achieves better validation results than relying on local patterns in the sensor data.

In order to quantify this trend, we propose the correlation coefficient r_{GP}. The coefficient is defined as the vectorized Pearson correlation coefficient between the generalization gap (G) and the difference in performance to the convolutional network (P) of each 1- and 2-layered architectural variant across all evaluation metrics. We report r_{GP} for all datasets in Figure 4.

10.3389/fcomp.2022.924954

TABLE 6 Per class results for all 9 architectural variations of the DeepConvLSTM architecture (Ordóñez and Roggen, 2016) using the Opportunity dataset (Roggen et al., 2010) as input.

Layers	Units	Metric	null	open_d_1	open_d_2	close_d_1	close_d_2	open_f	close_f	open_dw	close_dw	open_dr_1	close_dr_1	open_dr_2	close_dr_2	open_dr_3	close_dr_3	clean_t	drink_c	toggle_sw
0		Accuracy	75.52%	26.41%	26.62%	29.00%	33.34%	17.31%	17.74%	13.91%	14.99%	8.04%	10.33%	8.94%	5.76%	15.88%	20.84%	54.09%	33.32%	20.76%
		Precision	86.05%	49.19%	53.03%	53.15%	57.56%	27.00%	31.81%	24.51%	27.33%	20.51%	26.77%	22.22%	17.31%	31.15%	35.52%	72.57%	73.39%	34.24%
		Recall	86.41%	37.96%	35.12%	42.35%	47.77%	43.38%	40.10%	27.50%	28.92%	12.43%	15.78%	15.03%	10.82%	27.45%	36.66%	74.41%	40.31%	40.90%
		F1	86.00%	41.29%	40.95%	44.57%	49.20%	29.38%	30.01%	24.02%	25.80%	14.35%	18.17%	15.42%	10.33%	26.74%	34.26%	69.52%	48.44%	33.86%
1	128	Accuracy	76.06%	18.89%	20.97%	20.76%	30.92%	8.69%	10.16%	5.13%	6.61%	0.60%	0.48%	1.77%	2.44%	9.83%	14.58%	46.72%	32.56%	12.49%
		Precision	82.91%	57.29%	51.36%	51.86%	60.35%	14.84%	20.51%	15.77%	24.63%	4.33%	12.97%	15.64%	14.44%	25.83%	25.11%	78.20%	86.58%	42.09%
		Recall	**90.84%**	22.49%	27.10%	29.46%	45.61%	25.07%	28.43%	9.35%	14.00%	0.71%	0.58%	2.45%	3.51%	17.01%	32.25%	61.18%	34.48%	18.01%
		F1	86.38%	30.11%	32.36%	32.83%	46.07%	15.86%	18.13%	9.53%	12.10%	1.14%	0.92%	3.13%	4.38%	17.21%	24.99%	60.71%	47.55%	20.75%
	256	Accuracy	76.61%	24.61%	29.97%	29.55%	32.02%	12.37%	11.42%	8.08%	8.81%	2.88%	3.28%	4.75%	4.62%	13.91%	17.60%	55.36%	41.27%	18.52%
		Precision	84.97%	66.71%	57.00%	50.65%	60.12%	21.90%	22.98%	21.16%	17.70%	11.54%	20.07%	25.80%	21.28%	27.41%	30.58%	77.98%	**87.95%**	40.15%
		Recall	89.32%	29.43%	40.72%	44.07%	47.26%	28.24%	32.13%	14.71%	18.99%	4.48%	3.72%	5.44%	5.73%	27.46%	32.42%	70.65%	44.71%	33.01%
		F1	86.71%	38.12%	43.90%	43.77%	46.37%	21.85%	20.24%	14.65%	15.87%	5.14%	5.94%	8.44%	8.45%	23.88%	29.56%	70.47%	57.77%	30.08%
	512	Accuracy	76.91%	28.22%	31.66%	34.72%	34.09%	14.35%	13.26%	10.74%	11.20%	4.43%	3.77%	7.31%	8.13%	14.04%	18.23%	56.57%	45.57%	24.71%
		Precision	86.60%	**70.08%**	58.35%	56.42%	63.24%	23.92%	24.38%	26.28%	22.94%	15.86%	15.93%	23.71%	20.66%	28.31%	27.66%	78.50%	86.99%	41.68%
		Recall	88.01%	33.98%	42.09%	50.51%	50.79%	36.37%	33.55%	20.30%	24.40%	5.72%	6.49%	9.01%	13.52%	25.15%	40.59%	72.99%	49.84%	45.27%
		F1	86.87%	43.15%	46.64%	49.95%	49.18%	25.03%	23.07%	19.02%	19.79%	7.89%	6.76%	12.42%	13.87%	23.96%	30.47%	71.06%	62.02%	39.00%
	1,024	Accuracy	**76.93%**	31.66%	35.29%	37.67%	38.71%	15.48%	13.49%	11.24%	12.70%	6.08%	4.11%	10.28%	9.66%	15.81%	19.42%	**60.32%**	45.93%	**26.39%**
		Precision	87.12%	69.88%	**62.94%**	56.77%	**63.32%**	24.92%	**26.54%**	32.47%	23.94%	14.41%	16.32%	35.17%	26.74%	31.62%	31.26%	**81.26%**	85.68%	**44.51%**
		Recall	87.44%	37.53%	45.50%	56.58%	55.48%	37.90%	38.12%	16.75%	27.26%	9.80%	5.28%	14.34%	15.44%	29.37%	40.39%	74.54%	50.80%	43.51%
		F1	**86.86%**	47.59%	50.97%	54.04%	55.07%	26.68%	23.46%	19.92%	22.25%	10.45%	7.52%	17.35%	16.82%	26.68%	32.07%	**73.88%**	62.27%	**40.94%**
2	128	Accuracy	66.53%	25.06%	25.21%	26.73%	28.11%	18.29%	14.40%	13.93%	15.83%	11.14%	13.94%	11.19%	14.31%	13.51%	20.79%	34.68%	38.46%	16.46%
		Precision	88.82%	42.09%	40.86%	41.66%	45.58%	23.16%	20.94%	25.33%	21.66%	18.35%	22.39%	27.09%	24.76%	29.20%	35.04%	46.23%	64.56%	20.94%
		Recall	72.67%	42.96%	41.77%	49.91%	43.62%	48.22%	46.30%	27.17%	44.51%	25.13%	33.96%	17.10%	30.90%	21.84%	45.74%	63.82%	52.72%	49.61%
		F1	79.66%	39.29%	39.39%	41.45%	42.00%	30.70%	24.96%	23.71%	26.89%	19.15%	23.56%	19.17%	24.11%	22.84%	33.06%	50.39%	55.11%	27.82%
	256	Accuracy	68.29%	32.78%	31.27%	32.17%	34.15%	18.88%	14.94%	20.18%	18.03%	16.95%	15.81%	10.21%	15.81%	19.03%	22.70%	42.59%	39.31%	19.05%
		Precision	88.66%	44.79%	49.57%	45.53%	50.85%	22.89%	20.46%	35.03%	26.21%	25.58%	21.89%	26.93%	22.10%	34.63%	39.53%	57.79%	74.55%	25.90%
		Recall	74.68%	54.31%	49.96%	56.25%	50.27%	51.50%	47.85%	36.46%	46.61%	33.88%	43.01%	14.88%	40.21%	31.43%	44.26%	69.83%	47.74%	47.18%
		F1	80.82%	47.47%	45.57%	47.67%	48.03%	31.36%	25.74%	32.39%	29.97%	27.72%	26.23%	17.47%	26.53%	30.72%	35.82%	59.45%	55.17%	31.15%
	512	Accuracy	70.47%	40.85%	40.12%	**40.12%**	38.89%	21.32%	17.27%	23.21%	19.60%	19.39%	19.81%	14.34%	20.95%	27.98%	25.22%	46.49%	45.55%	23.70%
		Precision	90.79%	55.61%	56.07%	53.29%	55.83%	25.06%	23.16%	36.69%	25.95%	27.67%	28.11%	31.44%	30.79%	46.39%	**40.08%**	57.43%	73.81%	31.68%
		Recall	75.91%	61.85%	57.89%	**61.14%**	56.95%	60.14%	55.64%	40.13%	51.07%	41.19%	44.08%	20.74%	42.52%	43.59%	49.86%	**76.34%**	56.90%	**51.68%**
		F1	82.39%	57.12%	55.90%	**55.92%**	53.33%	34.99%	29.14%	36.42%	32.13%	31.44%	32.05%	23.55%	33.88%	42.69%	38.73%	62.84%	62.09%	37.22%
	1,024	Accuracy	71.72%	**42.41%**	**44.11%**	39.67%	**42.57%**	**24.18%**	**19.03%**	**25.52%**	**22.87%**	**22.60%**	**22.13%**	**21.35%**	**26.32%**	**31.26%**	**27.92%**	47.54%	**48.35%**	24.49%
		Precision	**91.82%**	55.83%	61.10%	**58.54%**	56.61%	**28.90%**	26.49%	**37.91%**	**30.90%**	**32.33%**	**31.51%**	**38.26%**	**35.10%**	**48.24%**	39.91%	60.18%	73.26%	32.33%
		Recall	76.68%	**66.92%**	**61.69%**	57.87%	**61.88%**	**61.29%**	**60.69%**	**46.14%**	**53.49%**	**45.72%**	**45.82%**	**31.76%**	**54.30%**	**47.75%**	**54.07%**	76.00%	**61.39%**	50.92%
		F1	83.30%	**58.79%**	**60.13%**	55.71%	**57.03%**	**38.63%**	**31.60%**	**39.82%**	**36.18%**	**35.74%**	**35.39%**	**33.76%**	**41.07%**	**46.46%**	**42.39%**	63.49%	64.72%	38.10%

Results are averaged across 5 runs using a set of 5 different seeds. Written in bold font are the best results per activity and evaluation metric. To enhance readibility of the table activities are abbreviated: d, door; dw, dishwasher; f, fridge; dr, drawer; t, table; c, cup; sw, switch.

10.3389/fcomp.2022.924954

TABLE 7 Per class results for all 9 architectural variations of the DeepConvLSTM architecture (Ordóñez and Roggen, 2016) using the ADL sessions of the Opportunity dataset (Roggen et al., 2010) as input.

Layers	Units	Metric	null	open_d_1	open_d_2	close_d_1	close_d_2	open_f	close_f	open_dw	close_dw	open_dr_1	close_dr_1	open_dr_2	close_dr_2	open_dr_3	close_dr_3	clean_t	drink_c	toggle_sw
0		Accuracy	79.69%	16.42%	15.70%	19.63%	24.79%	15.66%	18.59%	9.21%	12.12%	4.26%	7.10%	3.24%	3.87%	15.71%	15.11%	**25.23%**	14.59%	8.82%
		Precision	90.52%	36.76%	35.07%	30.23%	41.45%	**24.62%**	26.04%	**16.21%**	18.62%	9.65%	**16.78%**	9.73%	12.87%	23.22%	20.65%	**49.50%**	46.05%	15.26%
		Recall	87.05%	25.12%	22.56%	38.63%	41.33%	40.45%	43.58%	25.05%	30.86%	7.87%	13.30%	5.27%	5.81%	38.44%	39.49%	**48.52%**	22.51%	19.85%
		F1	88.68%	27.61%	26.69%	32.37%	39.13%	26.77%	31.22%	16.68%	21.35%	8.04%	12.91%	6.13%	7.23%	26.90%	26.11%	**38.69%**	24.13%	16.06%
1	128	Accuracy	**81.41%**	0.61%	2.51%	6.66%	3.47%	8.45%	10.42%	1.50%	3.39%	0.00%	0.00%	0.00%	0.00%	1.21%	3.08%	0.14%	4.68%	1.66%
		Precision	85.91%	16.68%	13.72%	24.22%	29.11%	14.32%	25.33%	2.76%	8.42%	0.00%	0.00%	0.00%	0.00%	11.68%	8.66%	1.43%	34.44%	10.30%
		Recall	**94.17%**	0.67%	3.90%	9.53%	3.74%	27.99%	23.51%	6.31%	7.69%	0.00%	0.00%	0.00%	0.00%	1.44%	5.63%	0.15%	5.80%	1.98%
		F1	**89.73%**	1.14%	4.48%	11.34%	6.00%	15.38%	18.52%	2.85%	6.33%	0.00%	0.00%	0.00%	0.00%	2.27%	5.44%	0.27%	8.35%	3.04%
	256	Accuracy	80.96%	4.76%	13.68%	15.20%	14.20%	10.63%	13.80%	3.41%	5.55%	0.00%	0.00%	0.08%	0.00%	4.24%	8.52%	10.03%	13.35%	8.29%
		Precision	88.37%	31.18%	30.29%	29.87%	**57.40%**	18.11%	23.76%	11.99%	18.70%	0.00%	0.00%	0.42%	0.00%	20.29%	15.22%	29.83%	52.95%	28.31%
		Recall	91.06%	5.38%	20.79%	31.47%	20.99%	30.54%	40.50%	15.09%	13.17%	0.00%	0.00%	0.09%	0.00%	6.87%	18.49%	16.33%	15.83%	11.94%
		F1	89.45%	8.41%	21.44%	25.36%	23.44%	19.05%	24.02%	6.48%	10.35%	0.00%	0.00%	0.15%	0.00%	7.80%	14.98%	15.29%	22.18%	14.14%
	512	Accuracy	81.00%	12.20%	19.52%	23.55%	25.27%	13.65%	15.89%	3.55%	9.26%	0.23%	0.08%	0.00%	0.13%	7.26%	11.15%	15.60%	14.83%	16.62%
		Precision	89.51%	**57.79%**	45.66%	**36.87%**	47.65%	19.56%	22.37%	6.03%	22.74%	4.17%	0.71%	0.00%	5.00%	18.54%	16.82%	51.13%	**56.64%**	**36.11%**
		Recall	89.81%	13.83%	28.07%	47.13%	36.82%	41.32%	44.31%	12.53%	22.21%	0.24%	0.08%	0.00%	0.13%	13.27%	29.52%	28.13%	18.33%	24.86%
		F1	89.47%	21.25%	31.30%	37.50%	38.93%	23.91%	27.16%	6.63%	16.79%	0.46%	0.15%	0.00%	0.26%	13.05%	19.52%	23.78%	24.58%	27.79%
	1,024	Accuracy	80.32%	14.35%	19.77%	23.44%	**36.94%**	14.58%	15.18%	6.11%	9.89%	1.07%	0.24%	0.27%	2.10%	8.90%	14.55%	16.14%	17.68%	**18.00%**
		Precision	90.52%	55.70%	**51.36%**	31.49%	53.20%	20.60%	22.38%	11.38%	19.90%	7.81%	1.43%	7.00%	10.92%	19.87%	19.89%	45.18%	55.90%	33.09%
		Recall	88.08%	17.56%	26.50%	54.76%	**59.98%**	44.44%	43.58%	18.57%	27.39%	1.33%	0.33%	0.28%	2.82%	19.36%	36.39%	33.92%	22.77%	32.75%
		F1	89.03%	24.54%	31.12%	37.20%	**53.08%**	25.39%	26.05%	11.25%	17.80%	1.98%	0.46%	0.53%	3.86%	15.95%	25.01%	25.74%	28.69%	**29.93%**
2	128	Accuracy	70.93%	11.67%	11.40%	12.94%	6.76%	11.45%	15.29%	4.91%	14.31%	1.36%	7.17%	0.17%	4.80%	7.43%	17.07%	8.63%	12.40%	5.42%
		Precision	91.24%	21.96%	23.05%	19.48%	10.78%	15.44%	21.64%	10.32%	18.11%	8.02%	9.60%	0.89%	11.48%	18.15%	23.19%	15.72%	45.51%	6.31%
		Recall	76.48%	23.73%	19.98%	35.05%	18.35%	35.17%	57.20%	14.78%	56.42%	2.44%	24.54%	0.24%	9.91%	12.13%	51.27%	31.12%	17.79%	36.45%
		F1	82.74%	20.06%	19.90%	22.61%	11.75%	20.47%	26.29%	9.17%	24.73%	2.58%	12.83%	0.34%	8.90%	13.55%	28.83%	15.46%	21.17%	10.23%
	256	Accuracy	69.49%	18.52%	15.56%	16.67%	15.89%	13.77%	17.08%	6.10%	15.13%	1.82%	9.37%	2.12%	5.88%	12.06%	19.92%	13.52%	15.75%	7.36%
		Precision	92.86%	29.13%	26.87%	23.53%	24.88%	17.44%	24.27%	9.00%	19.61%	5.76%	11.41%	**14.23%**	13.71%	27.17%	26.98%	36.02%	38.50%	8.52%
		Recall	73.64%	37.11%	32.41%	43.37%	35.93%	43.33%	62.86%	23.75%	54.17%	3.14%	42.16%	3.04%	18.97%	18.75%	54.39%	40.26%	25.71%	**42.65%**
		F1	81.75%	30.34%	26.07%	28.07%	26.58%	24.06%	28.82%	11.19%	26.10%	3.53%	16.81%	4.04%	10.91%	21.09%	32.90%	23.19%	25.98%	13.61%
	512	Accuracy	72.28%	25.44%	23.59%	24.01%	26.29%	16.07%	19.15%	9.93%	**17.82%**	7.14%	**12.46%**	4.16%	10.49%	17.66%	21.48%	19.26%	17.22%	9.90%
		Precision	93.12%	40.48%	38.51%	35.32%	37.34%	19.71%	25.64%	13.97%	**22.78%**	**12.81%**	14.99%	9.93%	**16.10%**	32.49%	26.54%	37.69%	43.04%	13.49%
		Recall	76.52%	45.30%	43.34%	53.86%	51.34%	50.77%	**66.68%**	30.62%	54.83%	15.62%	**50.51%**	7.81%	29.87%	28.65%	62.29%	49.90%	26.99%	32.57%
		F1	83.59%	39.75%	37.00%	38.51%	40.56%	27.59%	31.66%	17.79%	**29.85%**	12.95%	22.01%	7.60%	**18.72%**	29.28%	35.03%	30.86%	28.19%	17.87%
	1,024	Accuracy	72.53%	**31.25%**	**30.81%**	**26.82%**	28.00%	**18.07%**	**19.95%**	**11.14%**	16.71%	**7.76%**	11.59%	**6.41%**	**10.53%**	**20.48%**	**23.32%**	20.45%	**20.23%**	11.34%
		Precision	**93.68%**	45.52%	48.41%	36.60%	39.26%	21.89%	**26.52%**	15.34%	20.47%	11.11%	15.49%	13.92%	14.59%	**30.44%**	**29.25%**	38.20%	40.25%	15.08%
		Recall	76.44%	**55.53%**	**49.42%**	**57.62%**	50.74%	**54.95%**	65.65%	**37.17%**	**56.58%**	**22.49%**	40.98%	**12.34%**	**32.13%**	**38.73%**	**65.14%**	46.13%	**33.24%**	38.70%
		F1	83.87%	**47.25%**	**46.09%**	**41.99%**	42.32%	**30.44%**	**32.69%**	**19.71%**	28.24%	**14.21%**	**20.47%**	**11.68%**	18.68%	**33.01%**	**37.22%**	32.99%	**32.23%**	20.32%

Results are averaged across 5 runs using a set of 5 different seeds. Written in bold font are the best results per activity and evaluation metric. To enhance readibility of the table activities are abbreviated: d, door; dw, dishwasher; f, fridge; dr, drawer; t, table; c, cup; sw, switch.

TABLE 8 Per class results for all 9 architectural variations of the DeepConvLSTM architecture (Ordóñez and Roggen, 2016) using the Drill sessions of the Opportunity dataset (Roggen et al., 2010) as input.

Layers	Units	Metric	null	open_d_1	open_d_2	close_d_1	close_d_2	open_f	close_f	open_dw	close_dw	open_dr_1	close_dr_1	open_dr_2	close_dr_2	open_dr_3	close_dr_3	clean_t	drink_c	toggle_sw
0		Accuracy	55.62%	36.31%	35.62%	41.43%	35.54%	6.75%	4.98%	17.79%	15.02%	12.59%	9.92%	24.24%	13.23%	22.23%	24.43%	66.96%	72.90%	31.14%
		Precision	73.73%	60.38%	56.34%	68.89%	53.61%	16.86%	19.78%	31.28%	37.40%	20.47%	38.70%	43.41%	32.86%	46.59%	45.50%	**84.03%**	85.60%	44.30%
		Recall	71.77%	53.32%	55.53%	60.22%	51.41%	17.49%	12.46%	34.35%	25.57%	24.78%	14.20%	33.67%	17.72%	32.57%	39.66%	80.71%	84.05%	60.85%
		F1	71.30%	52.58%	51.32%	56.34%	49.63%	12.19%	9.00%	29.40%	25.39%	20.58%	16.46%	35.77%	21.02%	33.87%	36.29%	79.70%	84.22%	44.96%
1	128	Accuracy	58.27%	46.84%	44.79%	52.84%	42.26%	11.78%	6.91%	19.17%	19.07%	14.37%	12.78%	19.74%	17.02%	23.44%	23.93%	67.84%	77.12%	36.88%
		Precision	83.65%	68.84%	60.67%	72.53%	60.02%	19.80%	22.58%	38.28%	35.88%	32.43%	33.83%	39.65%	34.50%	41.97%	41.12%	72.41%	85.29%	44.30%
		Recall	68.21%	64.08%	68.58%	68.58%	64.04%	36.77%	12.09%	30.26%	35.01%	23.21%	16.65%	27.41%	23.90%	36.89%	36.76%	94.20%	89.00%	78.57%
		F1	73.49%	63.12%	61.24%	67.49%	57.41%	20.25%	11.91%	31.22%	31.16%	23.73%	20.44%	30.20%	26.54%	35.11%	35.55%	79.09%	86.98%	53.01%
	256	Accuracy	60.85%	51.58%	49.33%	57.57%	49.33%	11.96%	8.15%	20.61%	25.21%	18.01%	19.28%	27.81%	20.59%	28.67%	28.65%	71.04%	77.94%	36.96%
		Precision	84.60%	72.67%	70.92%	71.30%	67.25%	22.91%	20.70%	41.59%	46.76%	34.16%	51.76%	45.00%	48.50%	44.41%	48.17%	75.82%	**86.43%**	45.03%
		Recall	70.08%	64.87%	68.70%	77.68%	72.17%	36.79%	14.40%	41.23%	44.04%	27.56%	23.90%	38.69%	27.80%	41.61%	40.02%	**94.41%**	88.99%	76.31%
		F1	75.58%	67.30%	65.04%	72.48%	64.77%	20.47%	13.51%	33.22%	38.73%	27.36%	29.60%	39.94%	31.18%	39.88%	41.09%	82.00%	87.50%	53.02%
	512	Accuracy	61.65%	53.40%	50.68%	59.73%	48.29%	12.35%	8.26%	23.99%	22.62%	25.75%	21.71%	35.61%	25.96%	34.22%	34.01%	**75.06%**	**78.71%**	39.08%
		Precision	83.83%	**75.08%**	72.34%	**74.94%**	66.04%	25.73%	32.18%	35.93%	52.68%	43.10%	53.91%	52.91%	49.28%	52.90%	49.63%	80.28%	86.33%	48.76%
		Recall	71.78%	66.95%	69.44%	78.25%	71.32%	26.36%	20.13%	49.50%	36.10%	36.27%	27.84%	45.38%	35.77%	44.98%	47.00%	93.79%	90.10%	74.68%
		F1	76.16%	68.89%	65.92%	74.00%	62.77%	20.73%	13.76%	38.13%	35.52%	37.82%	32.87%	48.03%	37.54%	46.18%	46.63%	85.19%	**87.98%**	55.01%
	1,024	Accuracy	**62.45%**	**55.30%**	**55.08%**	**60.87%**	48.55%	15.36%	9.78%	27.05%	27.22%	25.09%	19.49%	36.58%	25.91%	37.61%	35.06%	74.93%	78.48%	45.92%
		Precision	83.14%	74.04%	**74.20%**	74.74%	**69.61%**	31.55%	33.71%	43.04%	55.69%	41.72%	**54.16%**	**54.88%**	50.64%	56.26%	51.07%	80.11%	85.68%	54.91%
		Recall	**72.74%**	69.37%	**73.55%**	**78.84%**	71.06%	33.46%	18.51%	48.90%	41.20%	36.42%	24.12%	49.23%	35.09%	51.43%	51.59%	93.88%	90.37%	**79.72%**
		F1	**76.76%**	**70.10%**	**69.95%**	**74.64%**	62.50%	25.15%	15.68%	41.71%	41.10%	37.28%	29.86%	49.42%	37.97%	50.84%	48.14%	**85.24%**	87.78%	62.16%
2	128	Accuracy	36.76%	38.91%	39.61%	43.64%	42.69%	35.56%	22.02%	27.73%	42.62%	36.50%	28.35%	21.63%	23.42%	35.50%	36.86%	53.08%	65.51%	28.56%
		Precision	79.03%	49.09%	49.20%	51.00%	53.70%	48.69%	37.98%	42.26%	58.22%	45.66%	35.30%	32.07%	32.77%	55.46%	45.57%	60.20%	75.89%	32.10%
		Recall	40.96%	61.54%	67.99%	69.35%	67.67%	54.00%	34.42%	47.34%	65.00%	57.21%	46.10%	34.40%	41.08%	48.84%	**66.24%**	86.37%	83.13%	79.02%
		F1	53.63%	53.45%	54.51%	57.66%	56.62%	49.20%	33.77%	41.76%	58.35%	49.58%	39.17%	32.04%	34.42%	50.14%	52.26%	68.53%	78.87%	43.66%
	256	Accuracy	46.88%	40.29%	42.31%	45.38%	46.13%	38.68%	27.98%	35.88%	43.61%	38.26%	30.62%	28.81%	25.73%	35.89%	37.32%	56.53%	69.51%	39.57%
		Precision	82.63%	51.49%	56.01%	53.64%	56.45%	50.47%	51.49%	46.92%	57.96%	51.59%	39.32%	38.04%	38.84%	52.16%	51.86%	67.29%	76.55%	44.65%
		Recall	52.39%	63.50%	58.93%	71.95%	68.68%	58.77%	38.04%	61.87%	71.46%	56.19%	46.36%	44.10%	42.76%	51.33%	55.90%	82.67%	88.57%	76.51%
		F1	63.41%	54.54%	55.58%	58.72%	59.16%	52.39%	40.53%	51.03%	59.49%	51.39%	42.07%	40.19%	37.59%	50.28%	51.83%	71.73%	81.73%	55.51%
	512	Accuracy	52.09%	40.99%	40.89%	47.15%	48.84%	38.78%	26.40%	37.91%	45.05%	43.93%	34.76%	34.15%	33.60%	43.50%	**41.37%**	60.63%	70.21%	46.97%
		Precision	82.23%	55.06%	54.06%	54.86%	59.69%	52.53%	52.66%	51.55%	58.94%	**54.86%**	45.28%	46.01%	50.82%	64.57%	58.60%	71.97%	76.88%	54.26%
		Recall	59.18%	61.63%	56.57%	70.70%	70.36%	56.72%	38.50%	59.65%	71.81%	63.65%	49.38%	52.99%	54.04%	56.99%	60.66%	83.81%	89.54%	75.85%
		F1	68.25%	54.15%	52.00%	59.84%	61.32%	52.31%	39.36%	53.49%	61.03%	56.68%	46.52%	47.38%	48.19%	59.12%	**56.46%**	75.06%	82.29%	62.12%
	1,024	Accuracy	56.19%	51.92%	53.07%	52.96%	**57.37%**	**41.60%**	**32.99%**	**45.28%**	**50.12%**	**46.99%**	**38.55%**	**38.16%**	**36.98%**	**45.33%**	41.08%	64.04%	72.48%	**51.82%**
		Precision	**86.30%**	66.13%	69.28%	59.84%	67.03%	**53.33%**	**55.02%**	**58.18%**	**62.34%**	54.40%	50.69%	48.69%	**52.12%**	**68.42%**	**62.53%**	75.34%	75.63%	**59.70%**
		Recall	62.13%	**71.02%**	68.59%	77.31%	**77.47%**	**61.91%**	**49.90%**	**68.77%**	**77.34%**	**71.20%**	**55.33%**	**55.26%**	**54.70%**	**59.10%**	52.12%	85.54%	**94.87%**	79.11%
		F1	71.71%	66.43%	65.96%	66.38%	**70.34%**	**55.87%**	**48.17%**	**60.68%**	**64.78%**	**60.67%**	**51.98%**	**51.36%**	**51.27%**	**60.72%**	54.33%	77.38%	83.93%	**67.04%**

Results are averaged across 5 runs using a set of 5 different seeds. Written in bold font are the best results per activity and evaluation metric. To enhance readibility of the table activities are abbreviated: d, door; dw, dishwasher; f, fridge; dr, drawer; t, table; c, cup; sw, switch.

10.3389/fcomp.2022.924954

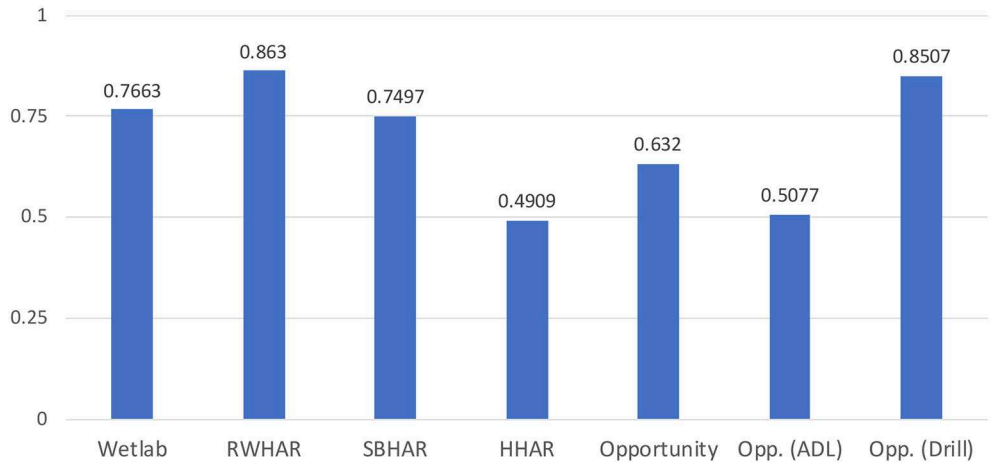

FIGURE 4
Calculated correlation coefficient, r_{GP}, for each dataset. The coefficient is calculated as the vectorized Pearson correlation between the generalization gap scores (G), i.e., difference between train and validation performance, and the performance differences between the variants including recurrent layers and the convolutional network. The closer r_{GP} is to 1, the more the network profits from including recurrent layers for the given dataset. According to r_{GP} the RWHAR dataset (Sztyler and Stuckenschmidt, 2016) profited the most and the HHAR dataset (Stisen et al., 2015) the least from recurrent layers.

Putting r_{GP} further into context with our previous analysis, a low r_{GP} value indicates that the learned temporal patterns are not applicable to the validation dataset. On the contrary, a high r_{GP} value indicates the opposite, i.e., that general temporal patterns are learned. This relationship is nicely visible when dividing the Opportunity dataset into the two session types. On the one hand, when only predicting the Drill sessions r_{GP} increases compared to its original value on the complete dataset as each subject was acting according to the same experimental protocol. On the other hand, when applied on the ADL sessions, r_{GP} significantly decreases as the dataset does not offer prominent temporal patterns which could be learned. Our metric can thus be used as an additional indicator on the importance learned temporal patterns during the prediction process and thus how beneficial recurrent layers are to the network.

In general, our analysis suggests that larger LSTMs are more prone to overfit and rely on temporal patterns during the prediction process. Nevertheless, this can only be beneficial for datasets which contain temporal patterns and relationships among activities, and can, as seen with the HHAR dataset (Stisen et al., 2015), also lead to worse validation results.

5. Conclusion and future directions

In this article we investigated the overall necessity of recurrent layers in HAR based on results we obtained on five popular HAR datasets (Roggen et al., 2010; Scholl et al., 2015; Stisen et al., 2015; Reyes-Ortiz et al., 2016; Sztyler and Stuckenschmidt, 2016). We chose to use the DeepConvLSTM (Ordóñez and Roggen, 2016) as our architecture of choice and modified it so that it either employs 0, 1, or 2 LSTM layers. We further varied the size of the LSTM layers by employing different amounts of hidden units, i.e., 128, 256, 512, or 1,024. During analysis we tried to map characteristics of the datasets to the performance of the individual architectures, trying to identify a "rule-of-thumb" for which types of dataset and activities a convolutional network can compete with a network also including recurrent layers.

Overall, in line with what we proposed in Bock et al. (2021), employing a 1-layered LSTM delivers the best prediction results for 4 out of 5 datasets. Nevertheless, we saw large discrepancies amongst different types of activities, depending on the amount of recurrent layers. Sporadic and transitional activities, which are short in time and do not show characteristic local patterns, were most reliably predicted when using two recurrent layers. Especially the latter type of activities inherit dependencies with preceding activities and are thus more reliably predicted using the overall temporal context. Contrarily, simple/ periodical activities, which show local, reoccurring patterns, were most consistently predicted when removing the second recurrent layer. Contradicting to our expectations, architectures employing a 1-layered LSTM were more performant on simple, periodical activities than solely convolutional networks. Furthermore, we witnessed an overall trend that the performance difference between LSTM-based networks and convolutional networks grows larger for datasets which feature rapid changes and overall shorter execution times of activities.

Our results showed that bigger LSTMs are more likely to overfit and rely on learned time sequences We further noticed that the correlation between the generalization gap of a recurrent

architecture and the difference in performance between the recurrent and convolutional architecture can be used to measure the effectiveness of the learned temporal patterns and thus the recurrent layers in general. Said correlation, which we defined as r_{GP}, is high, i.e.„ close to 1, if the temporal sequences learned by the LSTM help in predicting the validation data. In contrast, said correlation is low, i.e.„ close to 0, if recurrent layers are not beneficial for the given dataset and should be omitted from the architecture. We therefore also argue that claims presented by Karpathy et al. (2015) are not applicable to sensor-based HAR, as 1-layered LSTMs were also able to learn temporal patterns. We rather argue that the depth of an effective LSTM-based RNN comes down to how much relative importance one wants to put on temporal structures compared to local structures.

To give an answer on the necessity of recurrent layers in HAR, one has to consider the type of activities and overall use case. If one mostly tries to predict sporadic and transitional activities, convolutional kernels will struggle to identify local patterns in the data. For said activities temporal context and thus recurrent layers seem to be the only effective way to reliably identify them. In general, if the goal is to train a network which predicts activities within a predefined workflow, e.g., an experiment or production process, recurrent layers are beneficial and should be included in the final architecture. More specifically our experiments showed that a 1-layered LSTM deemed to be most effective in predicting activities within a predefined workflow, suggesting that it most efficiently combines both temporal and local information and is less prone to overfitting solely on the former. On the contrary, our results also suggest that in order to train a general system which predicts simple/ periodical activities, one should not include recurrent layers as they increase the risk of relying too much on temporal patterns which will not be present in the real-world application scenario. Unlike complex activities, simple/ periodical activities do not consist of in-activity sequences. Therefore, having the 1-layered variants outperform architectures employing no LSTM layer, suggests that said performance increase might be due to temporal dependencies amongst activities which were introduced during recording of the datasets. Given that a network is more likely to overfit on temporal patterns when employing two LSTM layers and that datasets like the RWHAR dataset (Sztyler and Stuckenschmidt, 2016) do not intend to model a certain activity workflow, having a 1-layered LSTM still produce the best prediction results along with a r_{GP} coefficient close to 1, makes us assume that models indeed learned (unwanted) temporal patterns. We also saw that for datasets which featured rapid changes between activities, convolutional networks were not as performant as recurrent networks in predicting simple/ periodical activities, causes us to think that latter networks took advantage of temporal patterns within said datasets.

Generally speaking, even though results we obtained during our experiments on paper suggest that LSTMs should be included at all times, we argue that, depending on the underlying use case, high benchmark scores on currently available HAR datasets can give a false sense of security and do not automatically ensure overall generability of trained models. We notice that for datasets which do not try to model a temporal process, e.g., the RWHAR dataset (Sztyler and Stuckenschmidt, 2016), including recurrent layers made networks learn the overall order in which activities were recorded, which, though increasing the benchmark score, would not benefit a model in a real-world setting and could even end up hurting the predictive performance as the model would rely on unnatural temporal patterns. Furthermore, we witnessed a significant decrease in performance for convolutional networks once simple/ periodical activities are changing more rapidly along with the fact that sporadic activities were not at all reliably detected using convolutional kernels. As in said cases temporal context deemed necessary, one has to question whether they would be able to be detected if naturally said context does not exist, but either only exists because of existing "partner"-relations among activities (e.g.„ *opening* and *closing a cupboard*), preceding activities (e.g., a person was *sitting*, thus *sit-to-stand* is very likely) or an experimental protocol. To conclude, the metric we introduced in this article, r_{GP}, be used to measure the effectiveness and applicability of learned temporal patterns and thus also be used as a decision metric whether to employ recurrent layers within the network.

To further examine the applicability of the introduced metric r_{GP}, our next steps within this research are four-fold. First, in order to prove that networks featuring recurrent layers are prone to overfitting on unwanted temporal patterns, we will explore regularization techniques which omit said temporal patterns and only keep those which can be expected to be witnessed in a real-world setting. Especially the RWHAR dataset offers a basis for such an analysis as it intends to model "independent" activities. We expect that putting such regularization techniques in place will end up hurting the performance of the networks which include recurrent layers, while leaving convolutional networks mostly unaffected. Second, we will further investigate the reason for why convolutional networks are less performant, especially for simple/ periodical activities, for datasets which contain rapid changes and thus short average execution times of activities. We already hypothesized that due to the nature how we define sliding windows, i.e., by the label of the last sample, a dataset which features rapid changes thus also contains more likely "mixed" windows, i.e., ones which contain multiple labels. In order to explore whether this is true, we plan to modify datasets to omit said "mixed" windows and see whether a convolutional network is more performant when only being trained on windows which only contain local patterns of the window label. Thirdly, we plan to expand our choice of datasets included in our analysis. We like to include datasets which (1) have subjects perform simple/ periodical activities without any restrictions or

underlying protocol, (2) consist of data obtained from a larger number of participants than our current choice of datasets and (3) include complex activities. Lastly, we plan to apply similar architectural changes to recurrent parts of other popular HAR networks, for example Abedin et al. (2021) or Dirgová Luptáková et al. (2022).

Data availability statement

Solely publicly available datasets were analyzed in this study. The code used for conducting experiments, links to the datasets as well as log files of all experiments can be found at: https://github.com/mariusbock/recurrent_state_of_the_art.

Author contributions

MB conducted the experiments and wrote the manuscript. AH performed the analysis of each dataset along with the color-coded visualization of the sensor data. AH, KV, and MM also contributed to the writing. All authors contributed equally in the analysis of the results.

Funding

This work received support from the House of Young Talents at the University of Siegen.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Abedin, A., Ehsanpour, M., Shi, Q., Rezatofighi, H., and Ranasinghe, D. C. (2021). "Attend and discriminate: beyond the state-of-the-art for human activity recognition using wearable sensors," in *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, Vol. 5* (New York, NY).

Bachlin, M., Roggen, D., Troster, G., Plotnik, M., Inbar, N., Meidan, I., et al. (2009). "Potentials of enhanced context awareness in wearable assistants for Parkinson's disease patients with the freezing of gait syndrome," in *International Symposium on Wearable Computers* (Linz), 123–130.

Bock, M., Hölzemann, A., Moeller, M., and Van Laerhoven, K. (2021). "Improving deep learning for HAR with shallow LSTMs," in *International Symposium on Wearable Computers* (New York, NY), 7–12.

Bordes, A., Chopra, S., and Weston, J. (2014). "Question answering with subgraph embeddings," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (Doha), 615–620.

Chen, K., Zhang, D., Yao, L., Guo, B., Yu, Z., and Liu, Y. (2021). Deep learning for sensor-based human activity recognition: overview, challenges, and opportunities. *ACM Comput. Surveys* 54, 1–40. doi: 10.1145/3447744

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (Almost) from scratch. *J. Mach. Learn. Res.* 12, 2493–2537. Available online at: http://jmlr.org/papers/v12/collobert11a.html

Dirgová Luptákov,á, I., Kubovčík, M., and Pospíchal, J. (2022). Wearable Sensor-Based Human Activity Recognition With Transformer Model. *Sensors* 22. doi: 10.3390/s22051911

Dua, N., Singh, S. N., and Semwal, V. B. (2021). Multi-input CNN-GRU based human activity recognition using wearable sensors. *Computing* 103, 1461–1478. doi: 10.1007/s00607-021-00928-8

Edel, M., and Köppe, E. (2016). "Binarized-BLSTM-RNN based human activity recognition," in *International Conference on Indoor Positioning and Indoor Navigation* (Alcala de Henares), 1–7.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE Trans. Pattern. Anal. Mach. Intell.* 35, 1915–1929. doi: 10.1109/TPAMI.2012.231

Glorot, X., and Bengio, Y. (2010). "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, volume 9 of Proceedings of Machine Learning Research*, eds Y. W. The and M. Titterington (Sardinia), 249–256.

Guan, Y., and Plötz, T. (2017). "Ensembles of deep LSTM learners for activity recognition using wearables," in *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, Vol. 1* (New York, NY).

Hammerla, N. Y., Halloran, S., and Ploetz, T. (2016). "Deep, convolutional, and recurrent models for human activity recognition using wearables," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (New York, NY), 1533–1540.

Haresamudram, H., Beedu, A., Agrawal, V., Grady, P. L., Essa, I., Hoffman, J., et al. (2020). "Masked reconstruction based self-supervision for human activity recognition," in *Proceedings of the International Symposium on Wearable Computers* (New York, NY), 45–49.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-,r., Jaitly, N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process. Mag.* 29, 82–97. doi: 10.1109/MSP.2012.2205597

Inoue, M., Inoue, S., and Nishida, T. (2018). Deep recurrent neural network for mobile human activity recognition with high throughput. *Artif. Life Rob.* 23, 173–185. doi: 10.1007/s10015-017-0422-x

Jaakkola, T., and Haussler, D. (1998). "Exploiting generative models in discriminative classifiers," in *Advances in Neural Information Processing Systems, Vol. 11* (Denver, CO).

Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2014). "On using very large target vocabulary for neural machine translation," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (Beijing), 1–10.

Karpathy, A., Johnson, J., and Li, F.-F. (2015). Visualizing and understanding recurrent networks. *CoRR, abs/1506.02078*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems, Vol. 1* (Lake Tahoe, NV), 1097–1105.

Lester, J., Choudhury, T., and Borriello, G. (2006). "A practical approach to recognizing physical activities," in *International Conference on Pervasive Computing* (Dublin), 1–16.

Lester, J., Choudhury, T., Kern, N., Borriello, G., and Hannaford, B. (2005). "A hybrid discriminative/generative approach for modeling human activities," in *19th International Joint Conference on Artificial Intelligence* (Edinburgh), 766–772.

Liao, L., Fox, D., and Kautz, H. (2005). "Location-based activity recognition using relational markov networks," in *19th International Joint Conference on Artificial Intelligence, Vol. 5* (Edinburgh), 773–778.

Mikolov, T., Deoras, A., Povey, D., Burget, L., and Černocký, J. (2011). "Strategies for training large scale neural network language models," in *IEEE Workshop on Automatic Speech Recognition Understanding* (Waikoloa, HI: IEEE), 196–201.

Murahari, V. S., and Plötz, T. (2018). "On attention models for human activity recognition," in *Proceedings of the 2018 ACM International Symposium on Wearable Computers, ISWC '18* (Singapore), 100–103.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., and Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *J. Big Data* 2, 1. doi: 10.1186/s40537-014-0007-7

Ordóñez, F. J., and Roggen, D. (2016). Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16, 115. doi: 10.3390/s16010115

Patterson, D. J., Fox, D., Kautz, H., and Philipose, M. (2005). "Fine-grained activity recognition by aggregating abstract object usage," in *9th International Symposium on Wearable Computers* (Osaka), 44–51.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., et al. (2018). A survey on deep learning: algorithms, techniques, and applications. *ACM Comput. Surveys* 51, 1–36. doi: 10.1145/3234150

Reddy, S., Mun, M., Burke, J., Estrin, D., Hansen, M., and Srivastava, M. (2010). Using mobile phones to determine transportation modes. *Trans. Sensor Networks* 6, 1–27. doi: 10.1145/1689239.1689243

Reiss, A., and Stricker, D. (2012). "Introducing a new benchmarked dataset for activity monitoring," in *2012 16th International Symposium on Wearable Computers* (Newcastle: IEEE).

Reyes-Ortiz, J.-L., Oneto, L., Sam,à, A., Parra, X., and Anguita, D. (2016). Transition-aware human activity recognition using smartphoneson-body localization of wearable devices: an investigation of position-aware activity recognition. *Neurocomputing* 171, 754–767. doi: 10.1016/j.neucom.2015.07.085

Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Förster, K., Tröster, G., et al. (2010). "Collecting complex activity datasets in highly rich networked sensor environments," in *7th International Conference on Networked Sensing Systems* (Kassel: IEEE), 233–240.

Sainath, T. N., Mohamed, A.-R., Kingsbury, B., and Ramabhadran, B. (2013). "Deep convolutional neural networks for LVCSR," in *IEEE International Conference on Acoustics, Speech and Signal* Processing (Vancouver, BC), 8614–8618.

Scholl, P. M., Wille, M., and Van Laerhoven, K. (2015). "Wearables in the wet lab: a laboratory system for capturing and guiding experiments," in *UbiComp '15: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Osaka), 589–599.

Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T. S., Kjærgaard, M. B., Dey, A., et al. (2015). "Smart devices are different: assessing and mitigatingmobile sensing heterogeneities for activity recognition," in *13th Conference on Embedded Networked Sensor Systems* (Seoul: ACM), 127–140.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems, Vol. 2* (Montreal, QC), 3104–3112.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., et al. (2015). "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (Boston, MA: IEEE).

Sztyler, T., and Stuckenschmidt, H. (2016). "On-Body localization of wearable devices: an investigation of position-aware activity recognition," in *International Conference on Pervasive Computing and Communications* (Sydney, NSW: IEEE), 1–9.

Tompson, J., Jain, A., LeCun, Y., and Bregler, C. (2014). "Joint training of a convolutional network and a graphical model for human pose estimation," in Proceedings of the *27th International Conference on Neural Information Processing Systems* (Montreal, QC), 1799–1807.

van Kasteren, T., Noulas, A., Englebienne, G., and Kröse, B. (2008). "Accurate activity recognition in a home setting," in *10th International Conference on Ubiquitous Computing* (Seoul), 1–9.

Xi, R., Hou, M., Fu, M., Qu, H., and Liu, D. (2018). "Deep dilated convolution on multimodality time series for human activity recognition," in *International Joint Conference on Neural Networks* (Rio de Janeiro), 1–8.

Xu, C., Chai, D., He, J., Zhang, X., and Duan, S. (2019). InnoHAR: a deep neural network for complex human activity recognition. *IEEE Access* 7, 9893–9902. doi: 10.1109/ACCESS.2018.2890675

Yuki, Y., Nozaki, J., Hiroi, K., Kaji, K., and Kawaguchi, N. (2018). "Activity recognition using dual-ConvLSTM extracting local and global features for SHL recognition challenge," in *International Joint Conference and International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers* (Singapore), 1643–1651.

Zappi, P., Lombriser, C., Stiefmeier, T., Farella, E., Roggen, D., Benini, L., et al. (2008). "Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection," in *Wireless Sensor Networks*, ed R. Verdone, 17–33.