Deep Convolutional Bidirectional LSTM for Complex Activity Recognition with Missing Data

Swapnil Sayan Saha*, Sandeep Singh Sandha* and Mani Srivastava

Abstract Complex activity recognition using multiple on-body sensors is challenging due to missing samples, misaligned data timestamps across sensors, and variations in sampling rates. In this paper, we introduce a robust training pipeline that handles sampling rate variability, missing data, and misaligned data timestamps using intelligent data augmentation techniques. Specifically, we use controlled jitter in window length and add artificial misalignments in data timestamps between sensors, along with masking representations of missing data. We evaluate our pipeline on the *Cooking Activity Dataset with Macro and Micro Activities*, benchmarking the performance of deep convolutional bidirectional long short-term memory (DCBL) classifier. In our evaluations, DCBL achieves test accuracies of 88% and 72% respectively for macro and micro-activity classification, exceeding performance over state-of-the-art vanilla activity classifiers.

1 Introduction

Human activity recognition (HAR) is defined as the process of correlating a sequence of granular human action primitives with similar data from a datastore [1]. The advent of smartphones and wearables has enabled a wide application spectrum for HAR, including healthcare monitoring, human-computer interaction, fitness tracking, and transportation mode recognition [1][2][3]. However, challenges of deploying

Swapnil Sayan Saha
University of California, Los Angeles, e-mail: swapnilsayan@ucla.edu

Sandeep Singh Sandha
University of California, Los Angeles, e-mail: ssandha@ucla.edu

Mani Srivastava
University of California, Los Angeles, e-mail: mbs@ucla.edu

*Both authors contributed equally to the paper

learning-enabled HAR systems in the wild include dealing with data from multiple noisy sensors with variable sampling rates [16], misaligned timestamps [15] and missing data [3]. Traditional machine learning approaches are unable to deal with abnormal data on their own, requiring handcrafted features and domain knowledge [2][3]. Furthermore, complex activities may be composed of a succession of simple activities, whose spatial context might alter with time [4].

The Cooking Activity Recognition Challenge [5][6][7] embraces the aforementioned hurdles, with the goal of building a classifier to classify 3 distinct macro and 10 distinct micro-activities as follows:

- Making a sandwich - cut, wash, take, put, other
- Preparing fruitsalad - cut, take, peel, add, mix, put, other
- Preparing cereal - cut, take, pour, peel, put, open, other

The training dataset consists of 30-second windows from 3 subjects in 288 data frames/files. Sensors include 4 triaxial accelerometers placed at left wrist, right wrist, right arm, and left hip and 29 triaxial motion capture (mo-cap) markers placed at random locations of the body, totaling 99 sensor channels. Each frame corresponds to a single macro-activity, with one-to-multiple possible micro-activities. Mo-cap samples are captured at approximately 100 Hz, while the accelerometers' sampling rates vary between 50Hz - 100Hz. The starting and terminal timestamps for each micro-activity are absent.

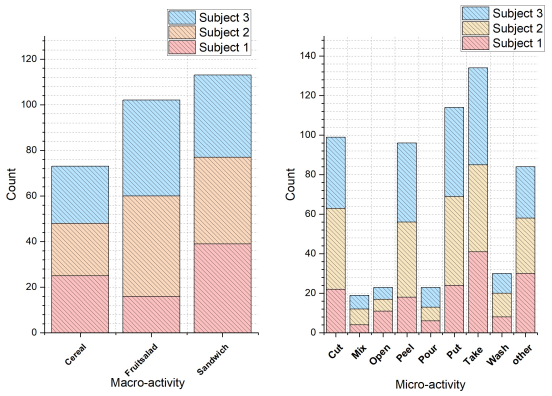

Fig. 1: Distribution of subject-wise class labels in the Cooking Activity Dataset.

Several techniques have been proposed in the literature to handle missing data in temporal streams. Hossain et al. [3][8] proposed extracting handcrafted spatial features to diminish individual effects of sample points, using traditional machine learning algorithms for activity recognition. However, the constituent primitives of complex activities might generate features that evolve with time [4], requiring careful a trade-off between differentiability and generalizability during feature selection

[2], ultimately leading to poor deployment performance for traditional classifiers compared to deep learning approaches [9]. This is evident in [10], where the authors propose using masking and time interval channels as representations of missing patterns to improve the prediction performance of deep recurrent architectures. Their evaluations illustrate that deep learning architectures offer better generalizability in test cases over traditional interpolation (e.g., MICE, MissForest, matrix completion, spline, KNN, forward / statistical imputation) and classification techniques (e.g., logistic regression, support vector machines and random forests) for temporal sequences. Yoon et al. [11] proposed a multi-directional recurrent neural network for missing data imputation that exploits correlation within and across data streams and with conclusions similar to [10], achieving 35-50% improvement in error metric over existing approaches. An alternative approach for multi-modal temporal streams is to intelligently "transfer knowledge" from one sensor to the missing sensor modality via temporal/spatial correlation across sensors or mapping to shared latent space [12][13][14].

Fig. 2 shows the overall pipeline for complex activity recognition using DCBL. We use an ensemble of 10 DCBL classifiers with majority voting decision fusion to classify macro-activities and conditionally select some of 20 micro-activity DCBL binary classifiers (one-vs-all, ensemble of 2) based on the macro-activity label. To handle missing & misaligned data and data with variable sampling rate, we introduce controlled time jitter & time-shift data augmentation [15] in the training pipeline, along with masking channels introduced in [10] to inform classifier of missing data.

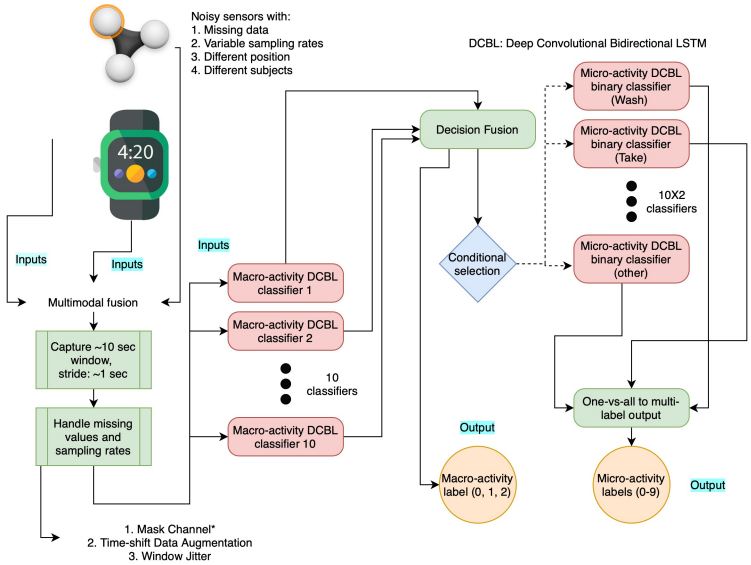

Fig. 2: Overview of handling complex activity recognition using DCBL (dotted lines indicate conditional selection based on macro-activity label).

2 Challenges and Training Pipeline

To handle the challenges of missing and misaligned sensor samples and variable sampling rates of multimodal data, we augment additional sensor channels in the training pipeline through masking representations of missing samples. Additionally, we perform data augmentation through artificial misalignments and jitter during window creation, effectively creating time-aware deep neural architectures.

2.1 Handling missing data

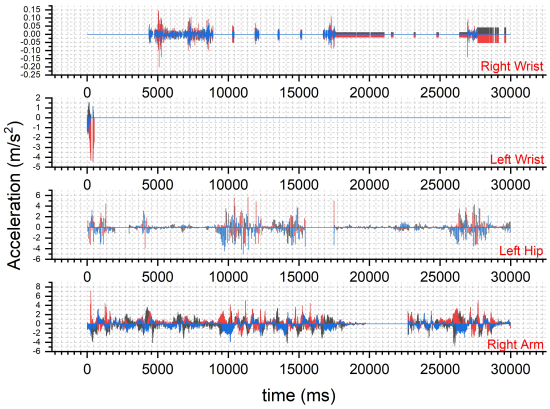

Fig. 3: Accelerometer plots for one of the 288 frames, indicating the presence of missing data and variable sampling rates.

Fig. 3 shows the plots for 12 accelerometer channels for one of the 288 training files, indicating missing samples. For our experiments, we chose to omit the 3 left-wrist channels due to wrongly annotated timestamps and high frequency of missing data in all files. Furthermore, mo-cap supplies absolute position rather than motion signatures unique to each action primitive. In our experiments, the motion capture data did not improve the validation accuracy. Thus, we trained final models using 9 accelerometer channels (right wrist, right arm, and left hip).

We use a masking channel introduced in [10] to signify the presence of missing data for each sensor channel. Instead of aligning the sample points, the individual overlapping windows were aligned based on initial and terminal timestamps of each window, appending zeros for missing samples while choosing an average sampling rate of 100 Hz for each window. The mask vector consists of zeros at timestamps where data is missing and one at all other points.

2.2 Jitter and data augmentation

We apply intelligent data augmentation techniques when creating sensor windows. Our augmentations exploit the observation that timing characteristics are variable and unstable for the devices capturing data during training and deployment settings [15]. Several factors, including delay in the operating system time stamping and variable instantaneous I/O load, are responsible for this variation. For fixed sampling rates, the general approach is to use a fixed duration for window creation. Aware of the sampling rate instability, we introduce a controlled timing jitter in the window length when creating windows from the training files. We hypothesize that the introduced jitter in the window length explicitly exposes the classifiers to the variable sampling rate, allowing them to generalize on the test data.

In multimodal sensing, Sandha et al. [15] have shown that data timestamps across sensors can have significant timing errors that can misalign the modalities. Causes of misaligned modalities include poor management of timing stack by the operating system and the choice of time synchronization techniques used by edge devices. This can result in overfitting of classifiers on the timing characteristics of data, with poor generalization in the deployment scenarios [16]. To avoid this situation, we use the proposed time-shift data augmentation approach [15] of adding artificial misalignments across the device's data timestamps when creating windows. This artificial misalignment helps the classifier maintain accuracy in the presence of variable timing characteristics in the wild.

3 Implemented Deep Learning Architectures

For evaluation, we applied the training pipeline on three deep neural architectures: CNN, LSTM and DCBL

3.1 Convolutional Neural Networks

CNNs are able to extract differential patterns in activity windows regardless of precise location by enforcing a degree of local connectivity among adjacent and enabling scale invariance (account for noisy feature discrepancy among same activity class), making the architecture suitable as a high-dimensional feature extractor and classifier for inertial complex activity recognition. Table 1 shows the architecture of implemented CNN, motivated from [2][17][18][19][20][21]. Each 2D convolutional layer contains 128 hidden units, with the kernel size set to (1, 10) with a stride of (1, 1) to ensure maximal overlapping (and correlation) among adjacent inertial samples, ensuring effective feature extraction. Batch normalization layers smooth the objective functions to account for noisy and unpredictable multimodal data. Dropout layers improve generalization performance such that the network ignores fine-grained subject-dependent motion signatures, preventing overfitting.

Table 1: Sample architecture of implemented CNN.*

Layer (type)	Param #	Size	Stride	Activation	Output Shape
Conv 1	1408	(1, 10)	(1, 1)	ReLu	(12, 500, 128)
Conv 2	163968	(1, 10)	(1, 1)	ReLu	(12, 500, 128)
B. Norm 1	512				(12,500,128)
Conv 3	163968	(1, 10)	(1, 1)	ReLu	(12, 500, 128)
B. Norm 2	512				(12, 500, 128)
Max. pool 1		(1,3)	(1,1)		(12, 166, 128)
Conv 4	163968	(1, 10)	(1, 1)	ReLu	(12, 166, 128)
Conv 5	163968	(1, 10)	(1, 1)	ReLu	(12, 166, 128)
B. Norm 3	512				(12, 166, 128)
Max. pool 2		(1,2)	(1,1)		(12, 83, 128)
Dropout 1					(12, 83, 128)
Conv 6	163968	(1, 10)	(1, 1)	ReLu	(12, 83, 128)
B. Norm 4	512				(12, 83, 128)
Max. pool 3		(1,2)	(1,1)		(12, 41, 128)
Dropout 2					(12, 41, 128)
Flatten 1					(62976)
Dense 1	3022912			ReLu	(64)
B. Norm 4	256				(64)
Dropout 3					(64)
Dense 2	195			Softmax	(3)

* Output shape and parameter numbers are different for micro-activity classifiers

3.2 LSTM Recurrent Neural Network

In order to infer temporal dependencies of a sequence of common integrant micro-activities across several macro-activities (e.g., "pouring" may arrive after "taking" in cereal preparation but "peeling" may arrive after "taking" in fruitsalad preparation) and translate them differently, it is necessary to store perceived windows, which can be achieved via LSTM and RNN [2][17][20]. LSTMs can handle variable delays between timestamps and events and learn long-term contextual dynamics of the time-series data for each sensor channel. Table 3 shows the architecture of implemented LSTM.

Table 2: Sample architecture of implemented LSTM.*

Layer (type)	Param #	Size	Stride	Activation	Output Shape
LSTM 1	70656			tanh	(500, 128)
Dropout 1					(500,128)
LSTM 2	131584			tanh	(128)
Dropout 2					(128)
Dense 2	387			Softmax	(3)

* Output shape and parameter numbers are different for micro-activity classifiers

3.3 Deep Convolutional Bidirectional LSTM

A special class of LSTM is bidirectional LSTM, which can 'look into' both the past and the future [20], increasing the amount of contextual information available to the network. DCBL is an amalgam of CNN and bidirectional LSTM, capable of extracting patterns while learning temporal dynamics of feature activations. Table 5 illustrates the architecture of proposed DCBL, motivated from [2][17]. The initial layers resemble the CNN architecture except at the end, where a bidirectional LSTM layer maps the highly-correlated spatial features extracted by the convolutional layers to the activity classes using recurrent information from both the past and future states. The bidirectional LSTM layer is added at the end instead of the beginning to prevent long-time windows from causing difficulties in learning for the recurrent layer [22], while at the same time, ensuring that the feature map activations are highly correlated across the samples from a large event duration.

Table 3: Sample architecture of implemented DCBL.*

Layer (type)	Param #	Size	Stride	Activation	Output Shape
Conv 1	1408	(1, 10)	(1, 1)	ReLu	(12, 500, 128)
Conv 2	163968	(1, 10)	(1, 1)	ReLu	(12, 500, 128)
B. Norm 1	512				(12,500,128)
Conv 3	163968	(1, 10)	(1, 1)	ReLu	(12, 500, 128)
B. Norm 2	512				(12, 500, 128)
Max. pool 1		(1,3)	(1,1)		(12, 166, 128)
Conv 4	163968	(1, 10)	(1, 1)	ReLu	(12, 166, 128)
Conv 5	163968	(1, 10)	(1, 1)	ReLu	(12, 166, 128)
B. Norm 3	512				(12, 166, 128)
Max. pool 2		(1,2)	(1,1)		(12, 83, 128)
Dropout 1					(12, 83, 128)
Conv 6	163968	(1, 10)	(1, 1)	ReLu	(12, 83, 128)
B. Norm 4	512				(12, 83, 128)
Max. pool 3		(1,2)	(1,1)		(12, 41, 128)
Dropout 2					(12, 41, 128)
Permute 1					(41,12,128)
Reshape 1					(41,1536)
Bi. LSTM	1704960				(256)
Dense	771			Softmax	(3)

* Output shape and parameter numbers are different for micro-activity classifiers

3.4 Implementation

For classification, the files in the training set were split into 60:20:20 (train:validation:test) ratio randomly for both macro and micro-activity classification. The dataset was split

by files (sessions/trials).

The models were implemented in Jupyter notebook (Python), using Keras and Sklearn via a Tensorflow backend, with MATLAB being used for minor errands such as generating labels and splitting training set. All models were trained on a GPU-machine with 128 GB RAM, 2x 12 GB Nvidia GeForce GTX 1080 Ti, and 3.4GHz AMD Ryzen Threadripper 1950X 16-core CPU. The total training time for the final 10 macroactivity classifiers was approximately 5 hours (including data preprocessing and loading). 40 epochs were used per classifier. The training time for the 20 microactivity binary classifiers was approximately 1 hour and 30 mins, with 10 epochs being used per classifier. The total time required to operate on the test set was approximately 8 minutes.

The right wrist, right arm, and left hip accelerometers were used in training the final models. Sliding overlapping window of length 10 second was used, with a stride of 1 second. No feature extraction or post-processing techniques were applied to the final models.

4 Classification Results

Performance characteristics for vanilla models:

Fig. 4 shows the classification performance of proposed classifiers (window-by-window basis) on the raw dataset without incorporating any interpolation, augmentation, masking, or jitter techniques. For micro-activity classification, a multi-label classifier was used.

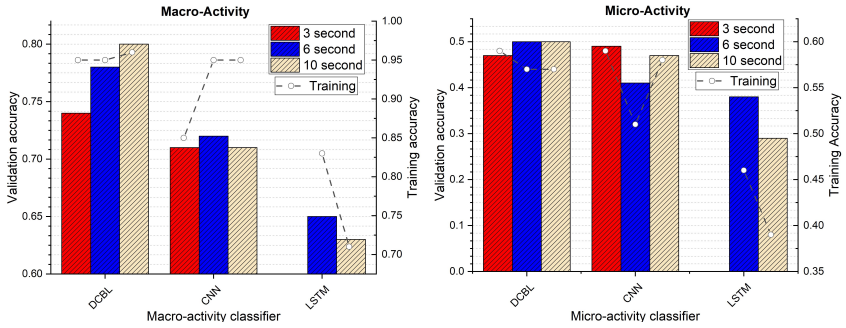

Fig. 4: Average classification performance (window-by-window basis) for proposed architectures without pre-processing applied.

From Fig. 4, we see that vanilla LSTMs perform poorly as they are unable to extract spatially-correlated temporal features over long time-windows. However, since a macro-activity is composed of distinct constituent micro-activities, smaller time-windows may not provide sufficient discriminatory features to distinguish between each activity. This trend is evident for CNNs and DCBLs, where increasing the time-window generally increases the classification performance. DCBL provides better generalization over CNN (although macro-activity classification can largely be attributed to learning transient features over the latent space) by taking into account temporal insights of constituent micro-activities apart from spatial features, which may not be similar for each macro-activity data frame due to distinct constituent micro-activity.

The best classification performance for both macro and micro-activity classification was observed for DCBL operating on 10-second windows, being 0.80 and 0.50, respectively, motivating us to choose DCBL-10 for our endeavors. The same classifier provided test accuracies of 0.73 and 0.37 on macro and micro-activity test splits respectively for window-by-window basis, and 0.77 and 0.48 on macro and micro-activity test splits respectively for file-by-file basis (taking maximum classification output of all windows in a file).

Effect of normalization and vanilla interpolation techniques

To manually account for missing data and smooth the output, pre-processing techniques such as bandpass filtering (using 20 Hz butterworth low-pass filter), normalization (z-normalization, uni-variance, and min-max scaling) and vanilla interpolation (linear, spline, autoregression) were applied but negligible or negative performance improvement was observed over vanilla approach. For example:

- Z-normalization on DCBL-6 provided a validation accuracy of 0.77 (compared to 0.78 for vanilla approach).
- Vanilla interpolation and smoothing techniques yielded validation accuracies as low as 30-40% for DCBL-3, DCBL-6 and DCBL-10. Our hypothesis is that since large sections of data are missing, vanilla interpolation (and smoothing) techniques do not generalize well to repair missing data.

Incorporating mo-cap data

Validation accuracies fluctuated within 30-50% for DCBL, CNN, and LSTM macro-classifiers using all 99 raw sensor channels (including mo-cap). Validation accuracy of 0.33 was observed for the DCBL-10 macro-classifier using only the 87 mo-cap channels. As discussed earlier, raw marker positions do not supply sufficient discriminatory features for each action class, yielding low accuracies. To extract temporal features such as velocity, acceleration, and jerk from the mo-cap channels, we utilized the RT-Mocap toolbox [23]. However, we did not observe significant

performance gains, with validation accuracies remaining within 37-44% for both DCBL micro and macro-classifiers through the extracted features.

Performance with augmentation and jitter (without masking)

Table 4 and Table 5 illustrate the performance of DCBL-10 for macro and micro-activity classification after incorporating jitter and data augmentation. To enhance the performance of micro-activity classification and allow usage of conditional classifier selection, we chose to train one-vs-all binary classifiers instead of multi-label classifiers.

Table 4: Macro-activity classification metrics for 5 random dataset splits.*

Trial	**Test (W-W)**	**Validation (W-W)**	**Train(W-W)**	**Test (F-F)**	**Validation (F-F)**	**Train (F-F)**
Split 1	0.75	0.79	1.00	**0.80**	0.83	1.00
Split 2	0.74	0.82	1.00	**0.78**	0.90	1.00
Split 3	0.84	0.80	0.99	**0.93**	0.85	1.00
Split 4	0.81	0.84	1.00	**0.88**	0.93	1.00
Split 5	0.73	0.73	0.99	**0.78**	0.83	1.00
5-way mean	**0.77**	**0.80**	1.00	**0.83**	**0.87**	1.00

* W-W stands for window-by-window basis. F-F stands for file-by-file basis (taking maximum classification output from all windows in a file).

Table 5: Micro-activity classification metrics (one-vs-all).

Binary Classifier	**Test (W-W)**	**Validation (W-W)**	**Train(W-W)**	**Test (F-F)**	**Validation (F-F)**	**Train (F-F)**
Cut (0)	0.81	0.78	0.86	**0.88**	0.82	0.99
Wash (1)	0.89	0.91	1.00	**0.90**	0.92	1.00
Take (2)	0.82	0.82	0.99	**0.84**	0.90	1.00
Put (3)	0.83	0.79	0.99	**0.90**	0.84	1.00
Peel (4)	0.87	0.85	0.98	**0.98**	0.96	1.00
Add (5)	0.94	0.97	0.99	**0.94**	0.98	0.99
Mix (6)	0.93	0.94	1.00	**0.92**	0.92	1.00
Pour (7)	0.91	0.92	1.00	**0.90**	0.92	1.00
Open (8)	0.96	0.93	1.00	**0.94**	0.92	1.00
Other (9)	0.84	0.88	0.99	**0.88**	0.86	0.99

For a file-by-file basis, average test accuracy of 0.72 was obtained for micro-activity classification for all 10 labels, yielding a 24% performance improvement over the vanilla approach (0.48). From Table 4, we can see that incorporating window jitter and time-shift augmentation yielded 6% test performance improvement on average over vanilla approach (0.83 vs 0.77) for macro-activity classification (file-by-file basis).

Performance with augmentation, jitter and masking

Table 6 and Table 7 illustrate the performance of DCBL-10 for macro and micro-activity classification after incorporating jitter and data augmentation, as well as masking from [10].

Table 6: Final macro-activity classification metrics (masking included)

Trial	Test (W-W)	Validation (W-W)	Train(W-W)	Test (F-F)	Validation (F-F)	Train (F-F)
Split 1	0.80	0.81	0.99	**0.88**	0.90	1.00
Split 2	0.91	0.97	1.00	**0.90**	1.00	1.00
Split 3	0.87	0.82	1.00	**0.95**	0.90	1.00
Split 4	0.82	0.86	1.00	**0.85**	0.95	1.00
Split 5	0.77	0.77	1.00	**0.80**	0.90	1.00
5-way means	**0.83**	**0.85**	1.00	**0.88**	**0.93**	1.00

Table 7: Final micro-activity classification metrics (masking included)

Binary Classifier	Test (W-W)	Validation (W-W)	Train(W-W)	Test (F-F)	Validation (F-F)	Train (F-F)
Cut (0)	0.80	0.79	0.98	**0.88**	0.87	1.00
Wash (1)	0.87	0.89	0.99	**0.88**	0.92	1.00
Take (2)	0.85	0.82	1.00	**0.90**	0.92	1.00
Put (3)	0.84	0.81	0.99	**0.86**	0.92	0.98
Peel (4)	0.85	0.89	1.00	**0.98**	0.92	1.00
Add (5)	0.94	0.96	0.93	**0.94**	0.96	0.93
Mix (6)	0.94	0.95	1.00	**0.94**	0.96	1.00
Pour (7)	0.91	0.92	0.99	**0.90**	0.92	0.98
Open (8)	0.94	0.93	0.99	**0.94**	0.93	1.00
Other (9)	0.86	0.81	0.99	**0.88**	0.82	0.99

From Table 6, we can see that incorporating masking yielded a 5% test performance improvement on average over the non-masking approach (0.88 vs. 0.83) for macro-activity classification (file-by-file basis). However, for micro-activity classification on a file-by-file basis, average test accuracy of 0.72 was obtained for the binary classifiers for all 10 labels using masking, which is same as the non-masked binary classifiers.

For the test dataset (without labels) provided for the competition, we used an ensemble of the 10 macro-activity classifiers (5 with masking and 5 without masking), while for macro-activity classification, we used 20 micro-activity binary classifiers, with 2 classifiers (masked and non-masked) for each of 10 micro-activities.

5 Analysis and Discussion

From our evaluation, we see that the performance of vanilla classifiers improve significantly when we incorporate our proposed training pipeline in the channel. Classical approaches such as interpolation, filtering, normalization and feature extraction (on mo-cap data) do not yield significant (in some cases lessen accuracy) performance improvement over vanilla classifiers. We hypothesize that classical approaches fail to converge due to the unnatural way the missing data was emulated in the dataset, consisting of long blocks of missing and abnormally aligned samples not normally encountered in the real world, coupled with the innate trade-off and time-evolving issues of classical methods discussed in section I. The uncommon

nature of the dataset actually hurts micro-activity classification performance gains when we incorporate masking in the pipeline along with jitter and augmentation, yielding in negligible performance improvement. This occurs because masking assumes perfect sample alignment among adjacent sensors. Thus, the gains obtained from "space-aware" neural architectures is nullified by the natural misalignments of sensor samples in the dataset. However, our proposed pipeline component, namely augmentation, and jitter, clusters the non-missing samples within a window at the beginning, ultimately yielding 24% performance improvement for complex activity recognition over vanilla approaches and thus exhibiting promising performance characteristics in worst-case multimodal sensing scenarios.

From Fig. 1, we see that the dataset is not balanced, with some classes having fewer occurrences than others. This affected the output on the test dataset, with some of the rarely occurring labels being missed by our classifiers. To minimize the penalty, we chose one-vs-all classifiers for micro-activity recognition to attempt to provide sufficient "positive and negative" examples to the classifiers. Additionally, we observed that subject-dependent training (e.g., use subjects 1 and 2's files for training and test on subject 3) yields inferior generalization performance due to the class imbalance as well as limited training examples. This also limits the direct usage of information-rich high sampling rate sensors (e.g., mo-cap), which translates to more model parameters and requiring more training examples than currently provided.

6 Conclusion and Future Work

In this paper, we introduce a robust learning pipeline for deep neural architectures to handle data from multiple sensors in the presence of missing data, misaligned samples, and variable sampling rates. We evaluate the applicability of the time-shift data augmentation, controlled window jitter, and masks to handle the hurdles mentioned above, benchmarking our pipeline using state-of-the-art DCBL architecture. Our evaluations yield 11% and 24% performance improvement for macro and micro-activity recognition over vanilla architectures.

There are several future directions for our work. While we have focused on online learning from missing data, an alternative approach is to impute the missing samples synthetically using generative adversarial models (GAN) [24][25][26], followed by training. Synthetic data generation may be useful when analyzing microscopic granularities in temporal streams as well as data augmentation. In particular, incorporation of information-rich and naturally & temporally aligned mo-cap data [27] requires data generation. In addition, the proposed training pipeline can be benchmarked on other temporal HAR modalities other than inertial samples such as mmWave ambient sensing [28] as well as sensor placement and availability constraints requiring spatially independent approaches [13][14][29].

Acknowledgements

The research reported in this paper was sponsored in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, the Army Research Laboratory (ARL) under Cooperative Agreement W911NF-17-2-0196 and the King Abdullah University of Science and Technology (KAUST) through its Sensor Innovation research program. Any findings in this material are those of the author(s) and do not reflect the views of any of the above funding agencies. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. S. S. Saha, S. Rahman, M. J. Rasna, A. K. M. Mahfuzul Islam and M. A. Rahman Ahad, *DU-MD: An Open-Source Human Action Dataset for Ubiquitous Wearable Sensors,* 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, 2018, pp. 567-572.
2. J. V. Jeyakumar, E.S. Lee, Z. Xia, S. S. Sandha, N. Tausik and M. Srivastava, *Deep convolutional bidirectional LSTM based transportation mode recognition.* Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers. 2018.
3. T. Hossain and S. Inoue, *A Comparative Study on Missing Data Handling Using Machine Learning for Human Activity Recognition,* 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Spokane, WA, USA, 2019, pp. 124-129.
4. T. Xing, M. Roig Vilamala, L. Garcia, F. Cerutti, L. Kaplan, A. Preece and M. Srivastava, *DeepCEP: Deep Complex Event Processing Using Distributed Multimodal Information,* 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 2019, pp. 87-92.
5. P. Lago, S. Takeda, K. Adachi, S. S. Alia, M. Matsuki, B. Benaissa, S. Inoue and F. Charpillet, *Cooking activity dataset with macro and micro activities*, IEEE Dataport, 2020. [Online]. Available: https://ieee-dataport.org/open-access/cooking-activity-dataset-macro-and-micro-activities. DOI: 10.21227/hyzg-9m49. Accessed: Mar. 15, 2020
6. P. Lago, S. Takeda, S. S. Alia, K. Adachi, B. Benaissa, F. Charpillet and S. Inoue, *A Dataset for Complex Activity Recognition with Micro and Macro Activities in a Cooking Scenario*, Preprint, 2020.
7. S. S. Alia, P. Lago, S. Takeda, K. Adachi, B. Benaissa, M. A. Rahman Ahad and S. Inoue, *Summary of the Cooking Activity Recognition Challenge*, Human Activity Recognition Challenge, Smart Innovation, Systems and Technologies, Springer Nature, 2020.
8. T. Hossain, H. Goto, M. A. Rahman Ahad and S. Inoue, *A Study on Sensor-based Activity Recognition Having Missing Data,* 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, 2018, pp. 556-561.
9. L. Wang, H. Gjoreski, M. Ciliberto, P. Lago, K. Murao, T. Okita and D. Roggen, *Summary of the Sussex-Huawei locomotion-transportation recognition challenge 2019.* Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers. 2019.
10. Z. Che, S. Purushotham, K. Cho, D. Sontag and Y. Liu, *Recurrent Neural Networks for Multivariate Time Series with Missing Values.* Sci Rep 8, 6085. 2018.

11. J. Yoon, W. R. Zame and M. van der Schaar, *Estimating Missing Data in Temporal Data Streams Using Multi-Directional Recurrent Neural Networks,* in IEEE Transactions on Biomedical Engineering, vol. 66, no. 5, pp. 1477-1490, May 2019.
12. R. Ohmura and R. Uchida. *Exploring combinations of missing data complement for fault tolerant activity recognition.* Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication. 2014.
13. J.V. Jeyakumar, L. Lai, N. Suda and M. Srivastava. *SenseHAR: a robust virtual activity sensor for smartphones and wearables.* Proceedings of the 17th Conference on Embedded Networked Sensor Systems (SenSys). 2019.
14. T. Xing, S. S. Sandha, B. Balaji, S. Chakraborty, and M. Srivastava. *Enabling edge devices that learn from each other: Cross modal training for activity recognition.* Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking, pp. 37-42. 2018.
15. S. S. Sandha, J. Noor, F. Anwar and M. Srivastava, *Time Awareness in Deep Learning-Based Multimodal Fusion Across Smartphone Platforms.* 5th ACM/IEEE Conference on Internet of Things Design and Implementation (IoTDI). 2020.
16. A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. *Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition.* In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys), pp. 127-140. 2015.
17. F. J. Ordóñez, and D. Roggen. *Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition.* Sensors 16.1: 115. 2016.
18. S. Ha, J. Yun and S. Choi, *Multi-modal Convolutional Neural Networks for Activity Recognition,* 2015 IEEE International Conference on Systems, Man, and Cybernetics, Kowloon, 2015, pp. 3017-3022.
19. M. Panwar, S. R. Dyuthi, K. C. Prakash, D. Biswas, A. Acharyya, K. Maharatna, A. Gautam and G. R. Naik, *CNN based approach for activity recognition using a wrist-worn accelerometer,* 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Seogwipo, 2017, pp. 2438-2441.
20. N. Y. Hammerla, S. Halloran, and T. Plötz. *Deep, convolutional, and recurrent models for human activity recognition using wearables.* In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16). AAAI Press, 1533–1540. 2016.
21. M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu and J. Zhang, *Convolutional Neural Networks for human activity recognition using mobile sensors,* 6th International Conference on Mobile Computing, Applications and Services, Austin, TX, 2014, pp. 197-205.
22. A. Graves and J. Schmidhuber. *Framewise phoneme classification with bidirectional LSTM and other neural network architectures.* Neural networks 18.5-6: 602-610. 2005.
23. D. Lewkowicz and Y. Delevoye-Turrell. *Real-Time Motion Capture Toolbox (RTMocap): an open-source code for recording 3-D motion kinematics to study action–effect anticipations during motor and social interactions.* Behavior research methods 48.1: 366-380. 2016.
24. J. Yoon, J. Jordon and M. van der Schaar. *GAIN: Missing Data Imputation using Generative Adversarial Nets*. Proceedings of the 35th International Conference on Machine Learning, in PMLR 80:5689-5698. 2018.
25. S. C. X. Li, B. Jiang and B. M. Marlin. *MisGAN: Learning from Incomplete Data with Generative Adversarial Networks*. 7th International Conference on Learning Representations (ICLR). 2019.
26. M. Alzantot, S. Chakraborty and M. Srivastava, *SenseGen: A deep learning architecture for synthetic sensor data generation*, 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, 2017, pp. 188-193.
27. J. K. Aggarwal,, and L. Xia. *Human activity recognition from 3d data: A review.* Pattern Recognition Letters 48: 70-80. 2014.
28. A. D. Singh, S. S. Sandha, L. Garcia, and M. Srivastava. *RadHAR: Human Activity Recognition from Point Clouds Generated through a Millimeter-wave Radar*. In Proceedings of the 3rd ACM Workshop on Millimeter-wave Networks and Sensing Systems (mmNets'19). Association for Computing Machinery, New York, NY, USA, 51–56. 2019.

29. S. S. Saha, S. Rahman, Z. R. R. Haque, T. Hossain, S. Inoue, and M. A. Rahman Ahad. *Position Independent Activity Recognition using Shallow Neural Architecture and Empirical Modeling*. In Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers (UbiComp/ISWC '19 Adjunct). Association for Computing Machinery, New York, NY, USA, 808–813. 2019.