Supplementary Material
Reduction of the Position Bias via Multi-Level Learning for Activity Recognition

#417

Appendix A Experimental Details

Appendix A.1 Datasets Description

We evaluate our proposed approach on three large-scale real-world wearable benchmark datasets featuring multi-location and heterogeneous sensors.

(1) The **SHL** dataset [4] is a highly versatile annotated dataset dedicated to mobility-related human activity recognition. It was recorded over a period of 7 months in 2017 in 8 different modes of transportation in real-life setting in the United Kingdom (*Still*, *Walk*, *Run*, *Bike*, *Car*, *Bus*, *Train*, and *Subway*). The dataset contains multi-modal data from a body-worn camera and from 4 smartphones, carried simultaneously at typical body locations (*Hand*, *Torso*, *Hips*, and *Bag*). The SHL dataset contains 3000 hours of labeled locomotion data in total. It includes 16 modalities such as accelerometer, gyroscope, magnetometer, linear acceleration, orientation, gravity, ambient pressure, cellular networks, etc.

(2) The Heterogeneity Dataset for Human Activity Recognition **(HHAR)** [11] provides data from smartphones and smartwatches built-in sensors specifically devised to investigate sensor-, device- and workload-specific heterogeneities on HAR models. The dataset features 2 types of modalities, i.e., accelerometer and gyroscope, sampled according to the highest sampling rate of the respective devices. A total of 6 activities carried by 9 different users were recorded, including *Biking*, *Sitting*, *Standing*, *Walking*, *Stair Up* and *Stair down*.

(3) **Fusion** dataset [9] containing accelerometer, gyroscope, linear acceleration and magnetometer from smartphones placed on *right upper arm*, *right wrist*, *belt*, *right and left jean pocket*. Data collection from all 5 positions was performed in a synchronized manner at a sampling rate of 50Hz. This dataset considers 8 different activities, including *walking*, *running*, *sitting*, *standing*, *jogging*, *biking*, *walking upstairs* and *walking downstairs*, which were performed by 10 participants.

Appendix A.2 Implementation details

For the closely related baselines, the available implementation are used otherwise we reproduce them. We use Tensorflow [1] for building the architecture of the

VAE used to model the learners in our proposed approach. Figure 1 illustrates the proposed architecture. As a preprocessing step, the annotated input streams are segmented into sequences, e.g., in the case of the SHL dataset, we obtain sequences of 6000 samples which correspond to a duration of 1 min. given a sampling rate of 100 Hz. To model the temporal dependencies in the considered sequences, we use LSTM cells [6]. For weight optimization, we use stochastic gradient descent with Nesterov momentum of 0.9 and a learning rate of 10^{-1}. Weight decay is set to 10^{-5}. The number of update steps τ_p performed by each local learner before the conciliation step is set to 100. We also use Bayesian optimization based on Gaussian processes as surrogate models to select the optimal hyperparameters of the models [10].

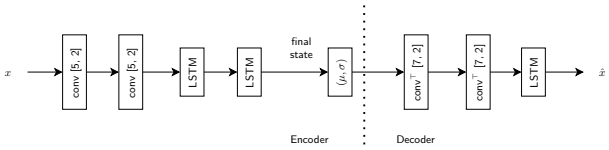

Fig. 1: Example of VAE architectures used to model the learners. All convolutions are 1D with their hyperparameters (kernel size and stride) shown. All layers are preceded by batch normalization and a ReLU activation. conv$^\top$ stands for transposed convolution. LSTM cells are used to capture the temporal dependencies in the considered sequences. The final states generated by the LSTM cell are used to model the latent distribution's mean and variance. The number of layers and their hyperparameters are optimized.

Appendix A.3 Ablation studies

For the ablation study, as mentioned in the main text, we also compare our approach with two baselines which do not perform the separation nor conciliation steps. The architecture of these basic models consists of convolution-based circuits for each position which are then fused together and trained jointly. We implemented two types of fusion schemes: concatenation-based and alignment-based fusion.

- **Concatenation-based alignment**: the outputs of the convolution-based circuits of each position are fused using a simple concatenation layer.
- **Correlation-based alignment**: the circuits' outputs are fused using a correlation-based conciliation layer [2] which allows the circuits to compensate each other's deficiencies.

The architecture of these basic models consists of convolution-based circuits for each position (see Figure 2) which are then fused together and trained jointly. We implemented two types of fusion schemes from the literature around multi-view representation learning, namely: concatenation-based joint-representation

learning (a basic fusion scheme) and alignment-based representation learning (a more advanced fusion scheme which is specifically devised to leverage a form of collaboration between the different position-specific views or perspectives).

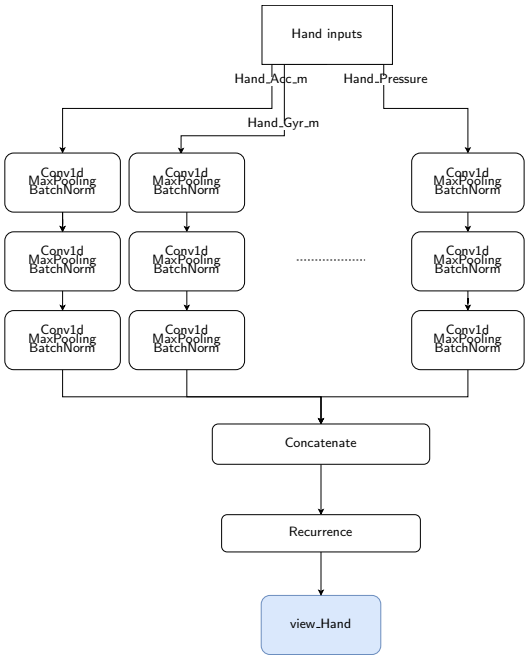

Fig. 2: Architecture of the *hand*-specific convolutional circuit taking as inputs the sensory signals (accelerometer, magnetometer, etc.) and performing fusion locally (sensory signals of a single position). Each phone position architecture is constructed by stacking up to 3 Conv1d/ReLU/MaxPool/BatchNorm blocks for processing each input channel individually. A recurrent layer, e.g., LSTM, is added on top of the fused representation.

Appendix A.4 Concatenation-Based Joint-Representation Learning

Suppose that $\mathbf{x}^a$ and $\mathbf{x}^b$ are two views (or perspective) of the same phenomenon. Concatenation-based fusion $h^{cat} = [\mathbf{x}^a, \mathbf{x}^b]$ is categorized according to [8] into multi-view representation fusion in a similar way with sum-fusion $h^{sum} = \mathbf{x}^a + \mathbf{x}^b$ and max-fusion $h^{max} = \max\{\mathbf{x}^a, \mathbf{x}^b\}$, where h refers, in each case, to the joint-representation learned from the views. In our experimental part, the outputs of the convolution-based circuits of each position are fused using a simple concatenation layer. Figure 3 illustrates how the different position-specific views are merged to form the joint representation. Learning can be performed in a two-step process where, first, the position-specific circuits are learned individually

then, second, fine-tuned in an end-to-end fashion with the presence of the fusion layer. Alternatively, the complete architecture can be trained in an end-to-end fashion.

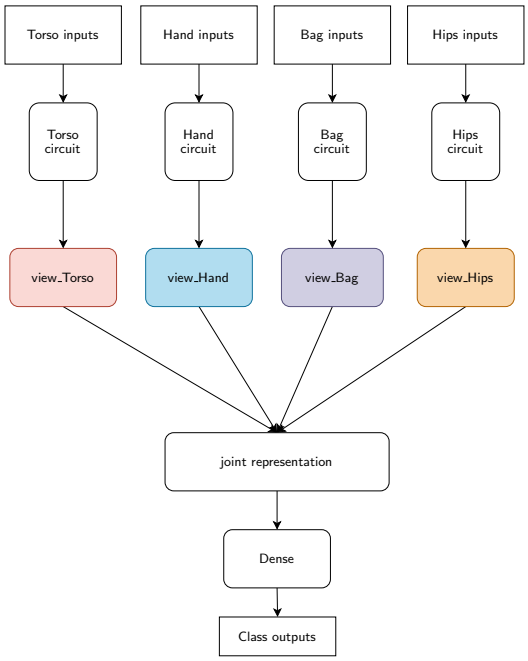

Fig. 3: Architecture of the concatenation-based joint-representation learning baseline illustrated on the inputs featured by the SHL dataset.

Appendix A.5 Alignment-Based Representation Learning

Multi-view Representation Alignment. Let's suppose again the same views as above and consider the problem of multi-view representation alignment which is defined as follows:

$$f(x^a; W_f) \leftrightarrow g(x^b; W_g) \tag{1}$$

where each view has a corresponding transformation (f or g) that takes the original space into a multi-view aligned space with certain constraints and $\leftrightarrow$ denotes the alignment operator. In the case of correlation-based alignment, which relies on the canonical correlation analysis (CCA) [7], this operator is concerned with finding a pair of linear transformations such that one component within each set of transformed variables is correlated with a single component in the

other set. This makes the corresponding examples in the two views maximally correlated in the projected space,

$$\rho = \max_{f,g} corr\big(f(x^a), g(x^b)\big) \tag{2}$$

where $corr(\cdot)$ denotes the sample correlation function. By maximizing the correlations between the projections of the examples, allows to obtain an embedding that tries to compensate for the pairwise deficiencies of the different views. In the problem of activity recognition from on-body sensor deployments, the outputs of the position-specific convolutional circuits (illustrated in Figure 2) are fused using a correlation-based conciliation layer [2] which allows to build non-linear transformations. Figure 4 illustrates the architecture based on representation alignment.

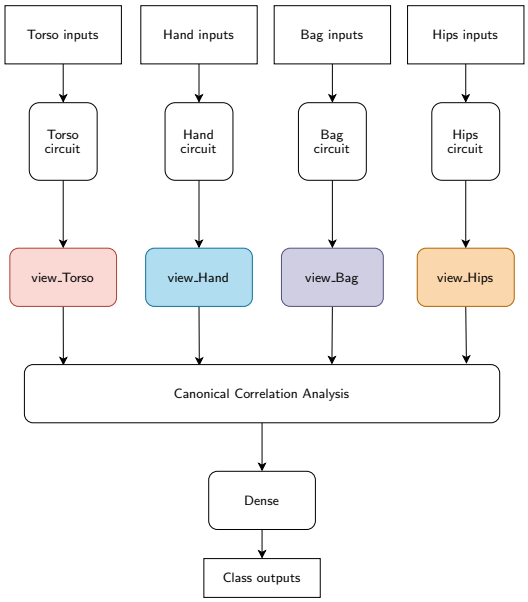

Fig. 4: Architecture of the CCA-based joint-representation learning illustrated on the inputs featured by the SHL dataset.

Appendix B Hyperparameters Optimization

We use Bayesian optimization based on Gaussian processes as surrogate models to select the optimal hyperparameters of the models [10]. Various tools exists in the literature which provide a comprehensive list of exploration strategies.

Among these tools, the Microsoft-NNI (Neural Network Intelligence) [1] constitutes one of the most comprehensive. Table 1 summarizes the hyperparameters being optimized as well as their respective bounds.

Hyperparameter	low	high	prior
Learning rate	0.001	0.1	log
Convolution layers			
├ Kernel size 1^{st}	9	15	-
├ Kernel size 2^{nd}	9	15	-
├ Kernel size 3^{rd}	9	12	-
├ Number of filters	16	28	-
└ Stride	0.5	0.6	log
Recurrent layers			
├ Number of units dense layer	64	2048	-
├ Number of hidden units 1	64	384	-
├ Number of hidden units 2	64	384	-
├ Inputs dropout probability	0.5	1	log
├ Outputs dropout probability	0.5	1	log
└ States dropout probability	0.5	1	log
Alignment layer			
├ View-spec. repr. dimension	10	40	-
├ Joint-repr. dimension	64	384	-
└ Dropout probability	0.3	0.9	log

Table 1: Hyperparameters of the architectural components and their associated ranges used for the construction of the surrogate model.

Appendix C Performance Measures

We use the f1-score in order to assess performances of the architectures. We compute this metric following the method recommended in [3] to alleviate bias that could stem from unbalanced class distribution. Given the usual definition of precision $\mathrm{Pr}^{(i)}$ and recall $\mathrm{Re}^{(i)}$ for the ith fold, we compute the f1-score by averaging its different components obtained for each fold as

$$\mathrm{F}_{\texttt{avg}} = \frac{1}{k} \cdot \sum_{i=1}^{k} \mathrm{F}^{(i)},$$

where $\mathrm{F}^{(i)} = 2 \cdot \frac{\mathrm{Pr}^{(i)} \cdot \mathrm{Re}^{(i)}}{\mathrm{Pr}^{(i)} + \mathrm{Re}^{(i)}}$, if both $\mathrm{Pr}^{(i)}$ and $\mathrm{Re}^{(i)}$ are defined. The i-superscripted measures correspond to measures obtained when the ith fold is used as the test set.

[1] https://github.com/microsoft/nni

Appendix D Evaluation and *Neighborhood Bias*

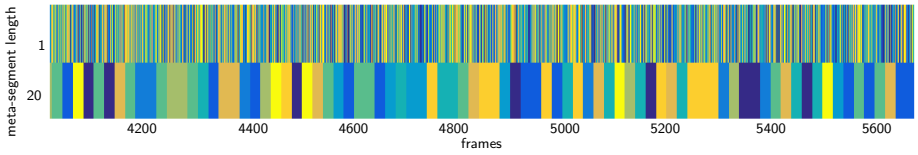

Fig. 5: Partitioning of a portion of the dataset's frames (or segments) over 10 folds using: (top) regular random partitioning used in traditional cross-validation procedures; (bottom) meta-segmented partitioning algorithm proposed in [5]. A segment length of one corresponds to the partitioning produced by the regular cross-validation procedure. The illustrated frames are temporally ordered. Each color corresponds to a different fold.

Model evaluation based on cross-validation usually rely on a random partitioning process. The random partitioning used in the case of segmented time-series introduces a neighborhood bias [5]. This bias consists in the high probability that adjacent and overlapping sequences, which are typically obtained with a segmentation process and that share a great deal of characteristics, fall into training and validation folds in the same time. This leads to an overestimation of the validation results and goes often disregarded in the literature. To alleviate the overestimation problem, we rely in our experiments on the *Meta-segmented partitioning* proposed in [5] which tries to circumvent this bias by, first, grouping adjacent frames into meta-segments of a given size. These meta-segments are then distributed on each fold. Figure 5 illustrates an example of the resulting partitioning obtained using random partitioning (often used with regular cross-validation) vs. meta-segmented partitioning [5].

References

1. Abadi, M., et al.: Tensorflow: a system for large-scale machine learning. In: OSDI. vol. 16, pp. 265–283 (2016)
2. Andrew, G., et al.: Deep canonical correlation analysis. In: ICML (2013)
3. Forman, G., Scholz, M.: Apples-to-apples in cross-validation studies. ACM SIGKDD **12**(1), 49–57 (2010)
4. Gjoreski, H., et al.: The university of sussex-huawei locomotion and transportation dataset for multimodal analytics with mobile devices. IEEE Access (2018)
5. Hammerla, N.Y., Plötz, T.: Let's (not) stick together: pairwise similarity biases cross-validation in activity recognition. In: UbiComp'15. pp. 1041–1051 (2015)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
7. Hotelling, H.: Relations between two sets of variates. In: Breakthroughs in statistics, pp. 162–190. Springer (1992)

8. Li, Y., Yang, M., Zhang, Z.: A survey of multi-view representation learning. IEEE TKDE **31**(10), 1863–1883 (2018)
9. Shoaib, M., Bosch, S., et al.: Fusion of smartphone motion sensors for physical activity recognition. Sensors **14**(6), 10146–10176 (2014)
10. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: NIPS. pp. 2951–2959 (2012)
11. Stisen, A., et al.: Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In: SenSys'15. pp. 127–140 (2015)