Designing Deep Neural Networks Robust to Sensor Failure in Mobile Health Environments

Abdullah Mamun[1], Seyed Iman Mirzadeh[2], and Hassan Ghasemzadeh[3]

***Abstract*— Missing data is a very common challenge in health monitoring systems and one reason for that is that they are largely dependent on different types of sensors. A critical characteristic of the sensor-based prediction systems is their dependency on hardware, which is prone to physical limitations that add another layer of complexity to the algorithmic component of the system. For instance, it might not be realistic to assume that the prediction model has access to all sensors at all times. This can happen in the real-world setup if one or more sensors on a device malfunction or temporarily have to be disabled due to power limitations. The consequence of such a scenario is that the model faces "missing input data" from those unavailable sensors at the deployment time, and as a result, the quality of prediction can degrade significantly. While the missing input data is a very well-known problem, to the best of our knowledge, no study has been done to efficiently minimize the performance drop when one or more sensors may be unavailable for a significant amount of time. The sensor failure problem investigated in this paper can be viewed as a spatial missing data problem, which has not been explored to date. In this work, we show that the naive known methods of dealing with missing input data such as zero-filling or mean-filling are not suitable for senors-based prediction and we propose an algorithm that can reconstruct the missing input data for unavailable sensors. Moreover, we show that on the MobiAct, MotionSense, and MHEALTH activity classification benchmarks, our proposed method can outperform the baselines by large accuracy margins of 8.2%, 15.1%, and 11.6%, respectively.**

I. INTRODUCTION

Compared to many other prediction problems, performing predictions on sensor data can have additional challenges due to the dependence on physical hardware. Notable examples of such limitations can be power consumption or energy budget, compute power, and, more importantly, dealing with hardware malfunction at the deployment time. However, while several works have studied the energy and compute limitations to improve the efficiency and performance of such prediction systems, not much has been done regarding the more severe scenario where one or more sensors are not accessible to the deployed model.

Suppose we have a prediction model that is designed and trained to perform predictions using a specific number of sensors. Then, the deployed model can potentially face one of the following challenging situations:

- One or more sensors are not available due to hardware problems.
- One or more sensors have been temporarily disabled to reduce power consumption.

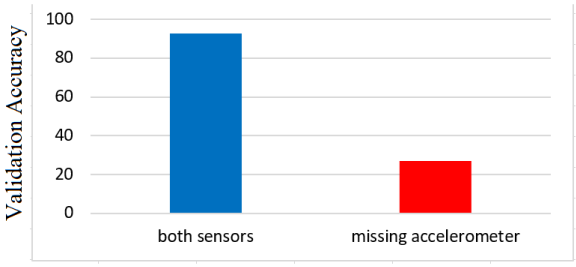

Fig. 1. The consequence of missing a sensor on the MobiAct classification benchmark. Left bar: prediction accuracy when both accelerometer and gyroscope are present. Right bar: when the accelerometer is not available and the data was imputed using the mean-filling method.

The absence of a sensor can significantly hurt the prediction quality. Figure 1 illustrates this problem on the MobiAct activity classification using accelerometer and gyroscope sensors. When the model has access to both sensors, it can reach the validation accuracy of 92.7%.

However, when the data from the accelerometer sensors are missing, a common practice is to fill the data using the average of the previously seen accelerometer data. This is known as the mean-filling method or mean-imputation method. Such a model gives the validation accuracy of 26.8%, which is significantly low, given that the MobiAct benchmark has 6 activities, and a random prediction can give accuracy of almost 16.6%, assuming the input data is balanced.

In this work, we study the implications of the "missing sensor data" for building a robust model that sustains its performance when one or more sensors are not available. More specifically, our contributions can be summarized as follows:

1) We study an important challenge that sensor-based prediction systems can face in the real world, namely, the missing input data from unavailable sensors.
2) We show that the consequence of this problem is that the prediction quality can degrade.
3) We propose a reconstruction algorithm that, at the training time, learns the relationship between various sensors' data distributions and can reconstruct the signal from a missing sensor at the test time.

[1]Abdullah Mamun is a Ph.D. student in the School of Computing and Augmented Intelligence and a Graduate Research Associate in the College of Health Solutions at Arizona State University, Phoenix, AZ 85054, USA a.mamun@asu.edu

[2]Seyed Iman Mirzadeh is a Ph.D. student in the School of Electrical Engineering and Computer Science at Washington State University, Pullman, WA 99164, USA seyediman.mirzadeh@wsu.edu

[3]Hassan Ghasemzadeh is an Associate Professor in the College of Health Solutions at Arizona State University, Phoenix, AZ 85054, USA hassan.ghasemzadeh@asu.edu

4) We compare our proposed method with the common practical baselines and show that it can outperform those baselines by a large margin.

II. BACKGROUND AND RELATED WORK

Different wearable devices or smartphones may have different sensors. A user may be willing to keep a particular sensor disabled from privacy concerns. If a machine learning model is designed to work with a particular format of input feature set, it will expect all these features to make an inference. It may also be possible that the machine expects triaxial sensor data whereas the input sensor is biaxial in a particular situation or one axis is unavailable for some reason [1]. Although works are being done to create adaptive machine learning models, these are not particularly helpful in our problem domain because they do not discuss the problem of sensor failure or spatial data imputation with time-series sensor data [2] [3].

There can be different natures of missing data, e.g. i) missing completely at random, ii) missing at random, or iii) missing not at random [4]. To deal with them, we can either remove the sample with missing features or we can replace the missing cells with estimated values. Removing the samples is not reasonable when there are a lot of missing values. When estimating values for missing cells, we can either consider a single value (single imputation method) or consider multiple candidates for one cell and keep them all (multiple imputation method) [5]. One such popular multiple imputation method is MICE but it may not be reasonable on massive datasets [6] [7]. Researchers have invented lots of hand-crafted methods and also adopted learning algorithms such as generative adversarial networks (GAN) for data imputation [8] [9].

A challenge somewhat similar to missing data is noisy data. Denoising autoencoders can effectively reduce noise from images and improve prediction accuracy [10]. Convolutional neural networks are also used for denoising images [11] [12]. Convolutional networks have also been studied in the context of adversarial attacks, transfer learning, and the adaptation of wearable sensor-based systems [13] [14]. Moreover, convolutional networks have also been found effective in capturing necessary information for health-related predictions, such as activity recognition and stress classification [15] [16]. Our task of reconstructing missing sensor data is highly inspired by the denoising autoencoders. However, our work is different from all these works in terms of the nature of the problem and the amount of missing data relative to the amount of available data. In our research, we consider situations when the amount of missing data can be as high as 50% of the feature set in a test case and we want to minimize the accuracy drop with signal reconstructing neural networks.

III. METHODOLOGY

In this section, we discuss our methodology by first giving a formal definition of our problem setup and then discussing our proposed approach. In Section V, we evaluate the performance of our proposed method on several activity recognition benchmarks.

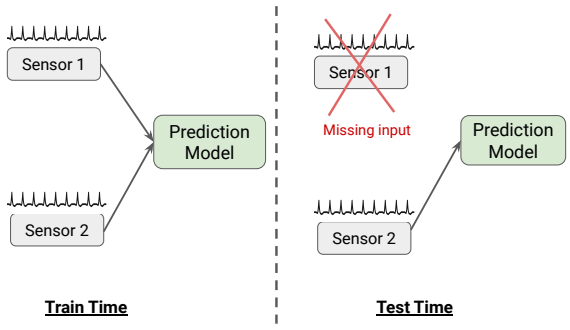

Fig. 2. Schematic illustration of our problem setup where one sensor can be unavailable at the test time.

A. Problem Setup

We assume a supervised learning classification scenario where the goal is to train a model $\mathcal{M}$ using the training dataset $\mathcal{D} = \{(x_1, y_1), \cdots, (x_m, y_m)\}$ that includes m input/output examples where each x_i is a vector that contains data from k sensors. In other words, we can write $x_i = (S_1, S_2, \cdots, S_k)$ where each S_i represents the data from one of the k sensors. In addition, each y_i represents the correct class label for each x_i. After the training phase is finished, we assume that during the test time, any of the k sensors can become unavailable to the model, and since the model assumes a vector of fixed size for input, our system has to deal with the missing values.

For instance, assume we aim to train a neural network on the human activity data using the accelerometer and gyroscope sensors, each with 3 axes. Then during the training, we have access to the training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{m}$ where each input example x_i can be written as $x_i = (a_x, a_y, a_z, g_x, g_y, g_z)$ where a_x, a_y, and a_z represent the values for the x-axis, y-axis and z-axis of the accelerometer. Similarly, g_x, g_y, and g_z represent the axes values for the gyroscope. However, during the test time (i.e., inference time), one sensor may become unavailable due to hardware limitations. For instance, if the accelerometer becomes unavailable, the model sees examples in the form of $x_i = (\text{null}, \text{null}, \text{null}, g_x, g_y, g_z)$, where "null" represents the missing inputs for the unavailable sensors. This has been illustrated in Figure 2.

B. Reconstructing missing signals

We have stated in Section I that the trivial approaches cannot provide us good results in prediction metrics. To find a better method, we consider that among the sensors $(S_1, S_2, \cdots, S_k)$, any one or more may be missing at the testing time. Recall from Subsection III-A that $\mathcal{M}$ is our classifier model. We want to get the highest possible accurate predictions despite having missing data in our test set. To the least, we will have to get a significant improvement over the trivial (mean-fill or zero-fill) approaches.

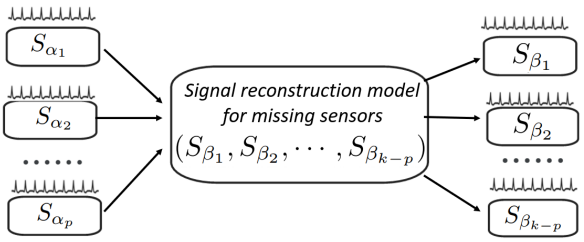

Fig. 3. An illustration of reconstruction of the missing sensors with a reconstructor. If the number of sensors in the system is k, we may need $2^k - 2$ such reconstructors to handle all possible scenarios.

Consider a scenario where p is the number of available sensors ($p \leq k$) and so $k - p$ is the number of missing sensors. Suppose, $(S_{\alpha_1}, S_{\alpha_2}, \cdots, S_{\alpha_p})$ are present and $(S_{\beta_1}, S_{\beta_2}, \cdots, S_{\beta_{k-p}})$ are absent. Here $1 \leq \alpha_i, \beta_j \leq k$ for $i = 1, 2, ..., p$ and $j = 1, 2, .., k-p$. Also $\{\alpha_1, \alpha_2, \cdots, \alpha_p\} \cap \{\beta_1, \beta_2, \cdots, \beta_{k-p}\} = \emptyset$. In this case, we will need to reconstruct the signals of the missing sensors using the available sensors. As the total number of sensors is k, we can have $2^k - 2$ possible situations to deal with. Because we do not need any reconstruction if all sensors are available and we cannot do anything if all the sensors are missing, that is why we are excluding these two situations from our consideration. So, if $k = 5$, we will need to create $2^5 - 2 = 30$ different reconstructor models for recreating the missing signals which is manageable if the reconstructor models are not very deep or complex. But if we were to keep 30 additional different models for prediction, which may be very deep, complex, or memory consuming, it would not be feasible for us in most cases. We present the reconstruction in Figure 3.

IV. EXPERIMENTAL SETUP

We need some datasets and experiments to test our proposed method from Section III. A common prediction task with sensor data is human activity recognition. In this section, we describe our datasets, model architecture, hyperparameters, validation method, etc.

A. Dataset

We choose three common and popular human activity recognition datasets: MotionSense [17], MobiAct [18], and MHEALTH [19] [20]. MHEALTH has time-series data from three accelerometers, two gyroscopes, two magnetometer sensors, and also 2-leads ECG signals. MotionSense and MobiAct have an accelerometer, gyroscope, and a few other device-motion features. However, we use the data from only one tri-axial accelerometer sensor and one tri-axial gyroscope sensor for each of those datasets and ignore the other features for our experiments. So, we have six features for each timestamp: readings of the x, y, and z axes of one accelerometer sensor and one gyroscope sensor.

B. Preprocessing

We cannot always just feed the raw time-series data into any machine learning model or neural network. Moreover, we want to build models that will work in the same way for all three datasets. So, we need to do some preprocessing as described throughout this subsection.

TABLE I
DESCRIPTION OF THE DATASETS AND OUR PREPROCESSING HYPERPARAMETERS

	MobiAct	*MotionSense*	*MHEALTH*
Sampling frequency (Hz)	50[a]	50	50
Number of subjects	67	24	10
Number of activities[b]	6	6	13
Window size used	5.12 s	5.12 s	5.12 s
Window overlap	75%	75%	75%
Size of training set[b]	314 MB	204 MB	195 MB

[a]Downsampled from 200 Hz.
[b]After preprocessing.

1) Choosing the activities: Motionsense has six activities: *walking downstairs, walking upstairs, walking, jogging, sitting, and standing.* The MobiAct dataset has nine activities of daily living but we choose the same six activities as in MotionSense. The MHEALTH dataset has 12 activities: "*standing still, sitting and relaxing, lying down, walking, climbing stairs, waist bends forward, frontal elevation of arms, knee bending, cycling, jogging, running, jump front and back*", and a *null class* activity [20]. We keep them all.

2) Creating windows: Our MobiAct dataset was initially sampled at 200Hz. We downsample it to 50Hz. The other two datasets are already in 50Hz. For every dataset, we pass the input signal through a Butterworth filter to reduce the amount of noise from the input signals. We choose 5.12 seconds as the window size with 75% overlapping between adjacent windows. Table I summarizes the information related to the datasets and preprocessing.

C. Two-sensor classifier

Our research question deals with the situation where the training set has no missing data but the test set may have some missing data. We train a Convolutional Neural Network (CNN) that takes accelerometer and gyroscope signals as input and processes them through 1-dimensional convolutional layers and predicts the correct activity. It is important to note that we do not need a state-of-the-art classifier to test our method. So, we find a model with reasonable accuracy and try to reduce the accuracy drop with reconstructor when a sensor is missing.

D. Resconstructor

As we have mentioned in Subsection III-B, we will need $2^k - 2$ reconstructors for k sensors. For $k = 2$ (accelerometer and gyroscope), we will need $2^2 - 2 = 2$ reconstructors. We can achieve at least 92.7% validation accuracy on all three datasets in human activity recognition problems with CNN, as we can see in Table IV. So, CNN models are capable of capturing information and extracting features from time-series data. For this reason, we use CNN in our reconstructors. We use 16 1-dimensional convolutional filters in the first layer and keep adding four filters of the same filter size in every subsequent convolutional layer. We vary i) the number

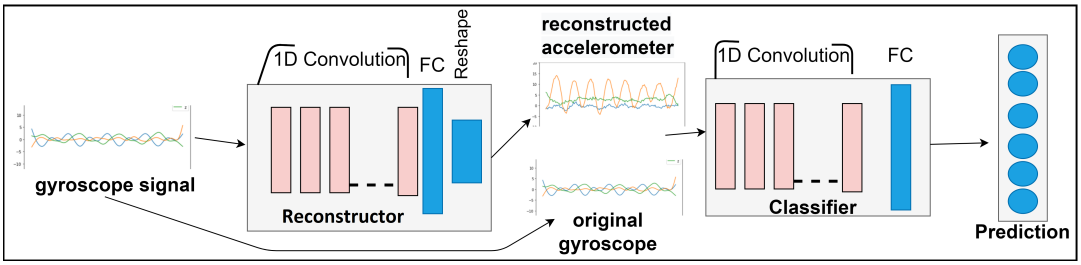

Fig. 4. Pipeline of a two-sensor classifier with reconstructor.

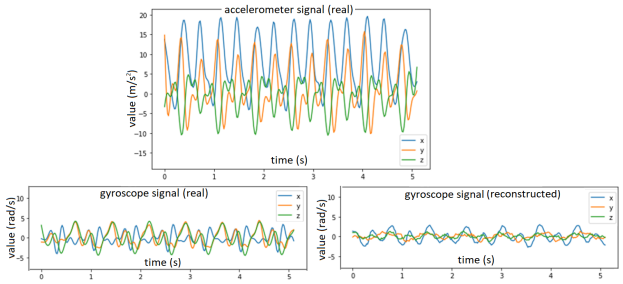

Fig. 5. A sample illustration of reconstruction of a gyroscope signal using an available accelerometer signal on the MobiAct dataset.

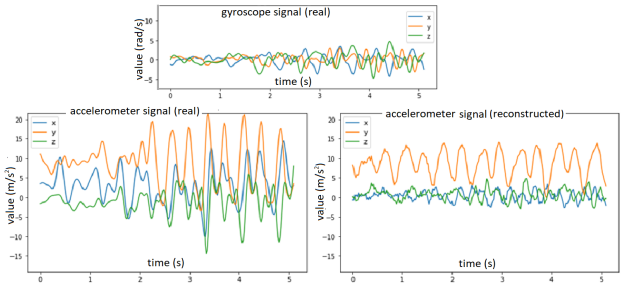

Fig. 6. A sample illustration of reconstruction of an accelerometer signal using an available gyroscope signal on the MobiAct dataset.

of convolutional layers, ii) filter size, iii) the learning rate, and iv) the number of epochs as hyperparameters as shown in Table II to find the optimal reconstructors. The number of convolutional layers has been kept small to minimize the risk of the vanishing gradient problem. The best reconstructor is the one that gets the maximum validation accuracy when the available sensor signal and the reconstructed signal together are fed to our classifier. We present the pipeline using a reconstructor in Figure 4.

We simulate a situation as if we have an existing system that we want to validate on data from unseen subjects. So we separate the training and validation sets based on the subjects as shown in Table III.

TABLE II

HYPERPARAMETERS FOR TRAINING THE RECONSTRUCTORS

Hyperparameter	*Possible values*
number of convolutional layers	7, 9
filter size	10, 30, 40, 50
learning rate	0.001, 0.0001, 0.00001
number of epochs	10, 30, 50

TABLE III

SPLITTING TRAINING AND VALIDATION SETS

Dataset	*Training subject Ids*	*Validation subject Ids*
MotionSense	1 - 20	21 - 24
MobiAct	1 - 64 for sitting 1 - 57 otherwise	65 - 67 for sitting 58 - 67 otherwise
MHEALTH	1 - 9	10

TABLE IV

VALIDATION ACCURACIES(%) OF TWO-SENSOR CLASSIFIER IN DIFFERENT SCENARIOS OF MISSING INPUT

	MobiAct	***MotionSense***	***MHEALTH***
Both sensors are present	92.7	94.5	93.7
Real accel with zero-filled gyro	89.0	35.7	86.9
Real accel with mean-filled gyro	88.9	35.8	86.6
Real accel with reconstructed gyro	**89.4**	**69.5**	**88.5**
Real gyro with zero-filled accel	34.3	76.0	9.9
Real gyro with mean-filled accel	26.8	**77.2**	9.0
Real gyro with reconstructed accel	**50.3**	73.6	**31.4**
Zero-fill (average)	61.7	55.9	48.4
Mean-fill (average)	57.9	56.5	47.8
Reconstructor (avg.)	**69.9**	**71.6**	**60.0**

V. RESULTS

In our experiments, we train classifiers with different sets of hyperparameters and keep the one with the best validation accuracy. Then we train reconstructors and find the reconstructor that gives us the best validation accuracy on data with missing sensors when used with the best classifier. We again train classifiers, keep the best one, and find the zero-fill and mean-fill accuracies on that classifier. Finally, we normalize the validation accuracy of the reconstructor for the best classifier that is used with the zero-fill and mean-fill accuracies. We explained in Table III, how we created the training and validation sets from our datasets. After the training is done, we evaluate our models on the validation sets and report them as the final results.

Our goal was to reduce the accuracy drop with reconstructors when one or more sensors are missing. If we look at Table IV, we notice that on average the reconstructor is giving

us better validation accuracies on all three datasets. In some cases, zero-fill and mean-fill can give us good accuracies too. But the accuracy gain with the reconstructor over zero-fill or mean-fill is visible in most cases. For instance, the average validation accuracy using our reconstructor on the MobiAct dataset is 69.9% whereas its closest competitor in the table is zero-fill (61.7%). We hope that we can do even better with the reconstructor with some fine-tuning. Also note that the zero-fill and mean-fill produce almost similar results in most cases, so we can say that taking the mean of sensor signals instead of filling with zeros does not help that much.

Another important thing to remember is that we want our reconstructors to generate sensor signals of reasonable shapes. We can see in Figure 5 and Figure 6 that our reconstructors are capable of generating realistic sensor signals for the missing sensors. So, we can say that our reconstructors work reasonably well in a two-sensor system and we can extend it for an arbitrary number of sensors as long as it is manageable.

VI. CONCLUSION

In this work, we have studied an important and practical challenge in multi-sensor-based prediction models. More specifically, we have studied the scenario where one or more sensors can become unavailable during the inference time of a deployed model. We have shown that the absence of one sensor in a two-sensor-based prediction system can significantly hurt the quality of predictions, and common practices for dealing with the missing input data do not perform well. To this end, we have proposed an algorithm that learns to reconstruct the missing signals during the training phase and using this algorithm in the test phase, can improve the accuracy of predictions significantly.

We believe that our main contribution is the generic framework and the demonstration that it is possible to employ it to reduce the accuracy drop of a classifier on a two-sensor system. In the future, we want to try our method on more than two sensors and we also intend to test this approach on low-quality datasets, where data are collected in uncontrolled environments. Finally, we note that the challenge of missing input data is very significant in sensor-based systems. While our method outperforms the several standard methods for data imputation, there is still room for improvement. We call for future research on this important topic.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation, under grants CNS-1750679, IIS-1852163, CNS-1932346, CNS-2210133, and IIS-1954372. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

REFERENCES

[1] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 1–17.

[2] M. Pedram, S. A. Rokni, M. Nourollahi, H. Homayoun, and H. Ghasemzadeh, "Resource-efficient wearable computing for real-time reconfigurable machine learning: A cascading binary classification," in *2019 IEEE 16th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, 2019, pp. 1–4.

[3] T. Vu, M. Eder, T. Price, and J.-M. Frahm, "Any-width networks," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Jun. 2020.

[4] R. N. Faizin, M. Riasetiawan, and A. Ashari, "A review of missing sensor data imputation methods," in *2019 5th International Conference on Science and Technology (ICST)*. IEEE, Jul. 2019.

[5] J. Honaker and G. King, "What to do about missing values in time-series cross-section data," *American Journal of Political Science*, vol. 54, no. 2, pp. 561–581, Apr. 2010.

[6] S. van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate imputation by chained equations in R," *Journal of Statistical Software*, vol. 45, no. 3, 2011.

[7] S. I. Khan and A. S. M. L. Hoque, "SICE: an improved missing data imputation technique," *Journal of Big Data*, vol. 7, no. 1, Jun. 2020.

[8] J. Yoon, J. Jordon, and M. van der Schaar, "GAIN: Missing data imputation using generative adversarial nets," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5689–5698.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014.

[10] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning - ICML '08*. ACM Press, 2008.

[11] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, Dec. 2016.

[12] Q. Bolsee and A. Munteanu, "Cnn-based denoising of time-of-flight depth images," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 510–514.

[13] R. K. Sah and H. Ghasemzadeh, "Adversarial transferability in wearable sensor systems," *CoRR*, vol. abs/2003.07982, 2020.

[14] S. A. Rokni, M. Nourollahi, P. Alinia, I. Mirzadeh, M. Pedram, and H. Ghasemzadeh, "TransNet: Minimally Supervised Deep Transfer Learning for Dynamic Adaptation of Wearable Systems," *ACM Transactions on Design Automation of Electronic Systems*, vol. 26, no. 1, pp. 1–31, Jan. 2021.

[15] 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," in *6th International Conference on Mobile Computing, Applications and Services*, 2014, pp. 197–205.

[16] R. K. Sah and H. Ghasemzadeh, "Stress classification and personalization: Getting the most out of the least," *CoRR*, vol. abs/2107.05666, 2021.

[17] M. Malekzadeh, R. G. Clegg, A. Cavallaro, and H. Haddadi, "Protecting sensory data against sensitive inferences," in *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*. ACM, Apr. 2018.

[18] G. Vavoulas, C. Chatzaki, T. Malliotakis, M. Pediaditis, and M. Tsiknakis, "The MobiAct dataset: Recognition of activities of daily living using smartphones," in *Proceedings of the International Conference on Information and Communication Technologies for Ageing Well and eHealth*. SCITEPRESS - Science and and Technology Publications, 2016.

[19] O. Banos, R. Garcia, J. A. Holgado-Terriza, M. Damas, H. Pomares, I. Rojas, A. Saez, and C. Villalonga, "mHealthDroid: A novel framework for agile development of mobile health applications," in *Ambient Assisted Living and Daily Activities*. Springer International Publishing, 2014, pp. 91–98.

[20] O. Banos, C. Villalonga, R. Garcia, A. Saez, M. Damas, J. A. HolgadoTerriza, S. Lee, H. Pomares, and I. Rojas, "Design, implementation and validation of a novel open framework for agile development of mobile health applications," *BioMedical Engineering OnLine*, vol. 14, no. Suppl 2, p. S6, 2015.