TEMPORALLY MULTI-SCALE SPARSE SELF-ATTENTION FOR PHYSICAL ACTIVITY DATA IMPUTATION

Hui Wei[1] **Maxwell A. Xu**[2] **Colin Samplawski**[1] **James M. Rehg**[3] **Santosh Kumar**[4]
Benjamin M. Marlin[1]
[1] Manning College of Information & Computer Sciences, University of Massachusetts Amherst
[2] Georgia Institute of Technology [3] University of Illinois Urbana-Champaign
[4] University of Memphis
{huiwei,csamplawski,marlin}@cs.umass.edu maxxu@gatech.edu
jrehg@illinois.edu santosh.kumar@memphis.edu

ABSTRACT

Wearable sensors enable health researchers to continuously collect data pertaining to the physiological state of individuals in real-world settings. However, such data can be subject to extensive missingness due to a complex combination of factors. In this work, we study the problem of imputation of missing step count data, one of the most ubiquitous forms of wearable sensor data. We construct a novel and large scale data set consisting of a training set with over 3 million hourly step count observations and a test set with over 2.5 million hourly step count observations. We propose an autocorrelation-informed sparse self-attention model for this task that captures the temporally multi-scale nature of step count data. We assess the performance of the proposed model relative to baselines based on different missing rates.

1 INTRODUCTION

Step count data collected by smart watches and activity trackers is one of the most ubiquitous forms of wearable sensor data, which provides valuable information about physical activity patterns and their relationship to other facets of health. However, wearable sensor data are subject to complex missingness patterns that arise from a variety of causes including device non-wear, insecure device attachment and devices running out of battery (Tackney et al., 2022; Lin et al., 2020).

These missingness issues can significantly hinder the utility of wearable sensor data. Indeed, missing step count data is a problem for traditional statistical analyses that aim to relate physical activity levels to other health events and to the effect of interventions (Master et al., 2022; Hall et al., 2020). It is also a problem when practitioners seek to use these data with missingness as input to common supervised, unsupervised and reinforcement learning-based models that require *complete* data (Papathomas et al., 2021; Liao et al., 2020; Zhou et al., 2018).

In this paper, we consider the problem of imputing missing step count data at the hourly level. This problem has a number of significant challenges due to the presence of high variability in patterns of physical activity both through time for a single person and between different people. To make progress on these challenges necessitates both carefully designed models and the availability of large-scale step count datasets.

To address these requirements, we curate a training set consisting of hourly step count data from 100 individuals and a test set from 500 individuals, which results in over 5.5 million hourly step count observations. Also, we introduce a novel sparse self-attention model which is specifically designed to handle the quadratic complexity of the vanilla self-attention mechanism, capture the temporally multi-scale information (e.g. daily, weekly, etc.) as well as combine a time encoding (hour of day, day of week) with a temporally local activity pattern representation.

We compare our proposed model to a broad set of prior models including the state-of-the-art architecture on missing step data imputation in actigraphy data (Jang et al., 2020). The results show that our model achieves statistically significant improvements in average predictive performance. Finally, we report the model performance with respect to various missing data rates.

2 DATA SET DEVELOPMENT

Data Set Extraction Our data set is derived from the *All of Us* Fitbit data set[1] (Mayo et al., 2023) and contains hourly-level data of step count, heart rate and wearing time (i.e. total wear minutes). Each hour aligns with each hour of the day (e.g. 9:00am to 10:00am) and is referred to as "hourly block". We define hourly blocks with zero minutes of wear time as *missing*, and hourly blocks with at least one minute of wear time as *observed*.

Since imputation model training requires holding out *observed* data to use as prediction targets thus increasing the amount of missing data seen by models during training, we form a training set of 100 individuals with low to moderate levels of natural missing data (i.e. $< 20\%$), including over 3 million observed hourly blocks with an average time series length of over $50,000$ hours per participant. Also, since most participants do not wear their devices between 11:00pm and 5:00am and the observed step count data for those who do are almost always 0 (presumably due to sleep), we focus on predicting step counts in the interval of 6:00am to 10:00pm.

To form a test set, we first exclude the training participants. Next, we select a total of 100 participants for each of five missing data level bins [0%, 20%), [20%, 40%), [40%, 60%), [60%, 80%), and [80%, 100%). This yields a total of 500 test participants with a total of approximately 2.5 million observed hourly blocks. We again assess missing data within the 6:00am to 10:00pm time frame.

Data Set Pre-Processing Once the data set is extracted, we apply several pre-processing steps. First, to deal with partially observed hourly blocks (i.e. wear minutes is more than 0 and less than 60), we use step rates as input features instead of step counts. The step rate of an hourly block is defined as the observed step count divided by the observed wear minutes within that hourly block. When making predictions for held-out observed hourly blocks, the model predicts step rates, but the loss is computed between the observed step count and the predicted step count formed by combining the predicted step rate with the observed wear minutes. Further, we normalize each participant's step rates and heart rates using personalized z-normalization (Ulyanov et al., 2016). Also, in order to enable vectorized computations over time series with missing data, we use zeros as the placeholder for missing data values and use an auxiliary response indicator time series to maintain information about which blocks are missing and which are observed. For more details about data pre-processing, please refer to Appendix A.

3 PROPOSED MODEL

Problem Definition We denote by $\mathcal{D} = \{\mathbf{C}_{l,t}^{(n)} | n = 1, \ldots, N, l = 1, \ldots, L, t = 1, \ldots, T_n\}$ a dataset of N participants, $\mathbf{C}^{(n)} \in \mathbb{R}^{L \times T_n}$ is a multivariant time series with L features and T_n hourly blocks. When considering data from a single participant, we drop the (n) superscript for brevity.

For each hourly block t, we let $\mathbf{C}_{s,t}$ be the step count and $\mathbf{C}_{w,t}$ be the wear time, and define a response indicator r_t in Equation 1 to indicate if the data is observed (i.e. with at least one minute of wear time). The problem is thus to impute $\mathbf{C}_{s,t}$ when $r_t = 0$ from all the other observed data.

$$r_t = \begin{cases} 1 & \mathbf{C}_{w,t} > 0 \\ 0 & \text{otherwise} \end{cases} \qquad m(t, t') = \begin{cases} 1 & t' \in \mathcal{A}^{(t)} \text{ and } r_{t'} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Sparse Self-Attention Let $\mathcal{A}^{(t)} \subseteq \mathcal{T}$ be a sub-set of all hourly blocks $\mathcal{T} = \{1, \ldots, T\}$ from a given participant that a query at time t is allowed to attend to, where a query represents a missing hourly block we would like to make the prediction for. To construct a self-attention based model for our long time series which have over $50,000$ hourly blocks on average, we require $|\mathcal{A}^{(t)}| \ll |\mathcal{T}|$ for all

[1] https://www.researchallofus.org

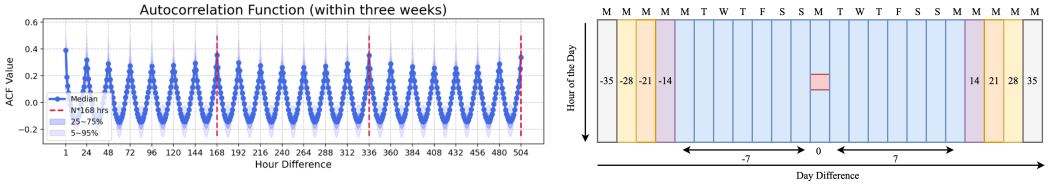


Figure 1: **Left**: Autocorrelation function (ACF) over all the participants of $\Delta t = 1, \ldots, 504$ hrs (within three weeks). Blue line: median ACF. Red line: $\Delta t = 168 \times N$ hrs (i.e. N weeks). **Right**: Multi-timescale context window. The missing hourly block is at the center and indicated as red. Numbers are day differences between each day and the center day (i.e. difference is 0) which contains the target hourly block (red). Letters indicate days of the week. The center day is Monday in this example.

t, where $|\cdot|$ is the size of a set. Additionally, we define a mask function $m(t, t')$ in Equation 1 that indicates both whether time point t can attend to time point t' and whether time point t' is observed.

In order to construct the self-attention set $\mathcal{A}^{(t)}$ which captures useful information to impute step count at time t, we consult the hourly step count autocorrelation function of our data set, which is presented in Figure 1 (Left). We can see that the autocorrelation is highest for the smallest time lags indicating high correlation between nearby hourly blocks. We can also see strong correlations at time lags of 24 hours (1 day) and 168 hours (1 week).

Based on these observations, we propose the multi-time scale context window shown in Figure 1 (Right) as our sparse self-attention set $\mathcal{A}(t)$. Letting d be the day number of the target hourly block t, the context window includes data from day d to day $d \pm 7$ as well as $d \pm 7k$ for $k \in \{2, 3, 4, 5\}$. Given that time points t' with hour of the day closer to the target hour t have higher correlations, we limit the context window to include time points t' with hours of the day close to that of t. Letting h be the hour of the day for the target hourly block t, the context window includes hours h to $h \pm 4$. Of course, the center of the context window, which corresponds to the target hourly block t, is not included in the sparse self-attention set.

Feature Representation Individual hourly blocks in our case are featurized by step count, step rate, average heart rate, wear time minutes, hour of the day, and day of the week. However, the target hourly block has unobserved step count, step rate and heart rate. A self-attention computation based on comparing the observed features of the target hourly block to the corresponding features of blocks in the self-attention set would thus be limited to expressing similarity based on hour of the day and day of the week. To overcome this problem, we augment the representation of an hourly block's step rate data using a window of activity data from $t-W$ to $t+W$, which allows for learning much richer notions of similarity. We refer to this feature as the "local activity profile" representation (LAPR).

Proposed Model The proposed model is summarized in the equations below. s_t is the predicted step rate at time t. $a_{tt'}$ is the attention weight from hourly block t to hourly block t'. $m(t, t')$ is the sparse attention mask function defined in Equation 1.

$$s_t = \sum_{t' \neq t} a_{tt'} v_{t'} \qquad a_{tt'} = \frac{m(t,t') \exp(\mathbf{q}_t^\top \mathbf{k}_{t'} + \boldsymbol{\theta}_{\mathtt{I}(t,t')})}{\sum_{u \neq t} m(t,u) \exp(\mathbf{q}_t^\top \mathbf{k}_u + \boldsymbol{\theta}_{\mathtt{I}(t,u)})} \tag{2}$$

The value $v_{t'}$, the query vector $\mathbf{q}_t$ and the key vector $\mathbf{k}_{t'}$ are produced using distinct neural network-based transformations of the input features for their respective time points. First, the LAPR is processed through an encoder network $Conv \rightarrow LayerNorm \rightarrow ReLU \rightarrow AveragePool$, which extracts more abstract features and prevents the overfitting problem by lowering the input dimension. Then the output of the encoder is concatenated with other observed features from the time point. Lastly, the whole feature set is projected through linear layers to produce the final $v_{t'}$, $\mathbf{q}_t$ and $\mathbf{k}_{t'}$.

A relative time encoding $\boldsymbol{\theta}_{\mathtt{I}(t,t')}$ is employed to encode information based on time difference between the target hourly block t and another block t', which provides similarity measures independent of the input features. The function $\boldsymbol{\theta}_{\mathtt{I}(t,t')}$ returns the value of the relative time encoding bias parameter for time point t' in the context window $\mathcal{A}^{(t)}$ centered at time t. If t' falls outside of the context window, this function returns 0.

Method Category	Method	Missing Rate [0%, 20%)	[20%, 40%)	[40%, 60%)	[60%, 80%)	[80%, 100%)	Overall
Basic Fill	Zero Fill	474.03 ± 34.43	408.37 ± 28.70	384.86 ± 33.68	440.94 ± 43.45	422.73 ± 58.58	395.79 ± 18.08
	Forward Fill	416.22 ± 32.09	373.31 ± 29.22	329.69 ± 30.76	381.71 ± 40.83	352.61 ± 47.99	351.87 ± 16.05
	Backward Fill	411.61 ± 30.16	368.52 ± 28.74	324.53 ± 28.92	373.68 ± 39.72	343.54 ± 49.93	346.46 ± 15.98
	Avg. F+B Fill	350.79 ± 26.43	316.96 ± 25.38	278.37 ± 25.75	321.10 ± 34.29	295.56 ± 41.23	306.80 ± 13.86
Micro Mean Fill	Participant	414.57 ± 32.10	375.79 ± 27.87	329.57 ± 27.44	385.16 ± 37.74	367.59 ± 54.93	356.03 ± 16.32
	Day of Week	411.82 ± 31.97	373.30 ± 27.70	327.20 ± 27.36	380.14 ± 37.34	355.83 ± 51.14	351.70 ± 15.74
	Hour of Day	373.93 ± 29.07	343.80 ± 25.07	303.03 ± 24.84	351.89 ± 32.28	311.45 ± 35.07	326.44 ± 12.88
	DW+HD	357.56 ± 27.79	330.69 ± 24.15	288.72 ± 23.93	325.40 ± 29.62	253.56 ± 30.15	304.08 ± 12.12
Mean Fill	Participant	416.01 ± 32.22	378.09 ± 27.99	332.59 ± 27.72	389.68 ± 38.32	370.80 ± 53.86	358.64 ± 16.27
	Day of Week	413.29 ± 32.10	375.67 ± 27.83	330.21 ± 27.64	384.73 ± 37.88	359.88 ± 50.41	354.50 ± 15.74
	Hour of Day	375.40 ± 29.19	346.02 ± 25.17	305.75 ± 25.09	356.46 ± 32.80	317.22 ± 35.80	329.50 ± 13.04
	DW+HD	359.07 ± 27.90	332.97 ± 24.26	291.78 ± 24.21	330.89 ± 29.97	262.46 ± 30.73	308.03 ± 12.21
Median Fill	Participant	369.17 ± 26.48	331.99 ± 23.03	299.41 ± 24.88	341.93 ± 32.48	323.09 ± 46.69	323.69 ± 14.12
	Day of Week	366.66 ± 26.38	329.82 ± 22.84	297.14 ± 24.76	337.97 ± 32.19	314.87 ± 45.49	320.37 ± 13.92
	Hour of Day	335.85 ± 24.41	307.19 ± 21.37	276.57 ± 22.96	316.67 ± 29.11	280.66 ± 32.40	300.62 ± 11.85
	DW+HD	322.03 ± 23.78	295.09 ± 20.67	262.29 ± 22.04	292.01 ± 26.82	**230.54 ± 27.15**	280.39 ± 11.18
*k*NN	Uniform	332.70 ± 25.34	306.68 ± 24.28	270.34 ± 23.74	321.63 ± 36.07	295.93 ± 40.81	305.46 ± 13.93
	Softmax	331.37 ± 25.08	305.90 ± 24.19	269.58 ± 23.65	315.26 ± 33.16	290.78 ± 37.53	302.58 ± 13.22
Model-based	Iterative Imputation (Azur et al., 2011)	313.23 ± 23.26	290.48 ± 21.32	260.61 ± 22.02	304.16 ± 28.86	289.20 ± 37.43	291.54 ± 12.29
	CNN-DAE (Jang et al., 2020)	317.26 ± 23.42	287.35 ± 21.27	256.22 ± 22.38	299.14 ± 30.66	284.27 ± 40.75	288.85 ± 12.93
	Regression Imputation (Little, 1992)	307.96 ± 22.88	284.06 ± 20.69	254.82 ± 21.69	296.01 ± 27.76	282.21 ± 36.75	285.01 ± 12.01
	BRITS (Cao et al., 2018)	299.46 ± 21.59	275.58 ± 19.93	248.21 ± 21.40	289.82 ± 28.34	275.09 ± 36.46	277.63 ± 11.85
	USGAN (Miao et al., 2021)	299.58 ± 21.58	275.58 ± 19.91	248.34 ± 21.43	289.68 ± 28.36	274.56 ± 36.36	277.55 ± 11.84
	MRNN (Yoon et al., 2018)	295.30 ± 20.91	270.38 ± 19.66	243.96 ± 21.21	280.69 ± 27.25	263.42 ± 33.20	270.75 ± 11.27
	Sparse Self-Attention (ours)	**285.76 ± 20.41**	**262.96 ± 19.16**	**239.01 ± 20.58**	**270.32 ± 25.62**	250.36 ± 30.12	**261.68 ± 10.62**

Table 1: Macro MAE ± 95% CI on completely held-out test participants. There are 100 randomly sampled participants in each missing interval. The entire held-out test cohort contains 500 participants in total. Red: overall best performance with statistical significance. Blue: best performance with statistical significance within each method category. Statistical significance level: $p < 0.05$.

Loss Function and Training We use mean absolute error (MAE) between true and predicted step counts as the loss function during model training. We use a stochastic gradient descent-based training approach where each batch contains instances sampled from different participants. We compute the MAE with equal weights on all samples in the batch. Additional hyper-parameter optimization and training details can be found in in Appendix C.

4 Experiments

Baselines We compare our proposed model to several commonly used strategies for imputing missing values in time series data, as well as to the state-of-the-art step count imputation method on the actigraphy data proposed by Jang et al. (2020). We group methods into several categories as shown in Table 1. Please refer to Appendix C for more details about baseline models.

Data Partitioning To set the hyper-parameters, we partition the training data described in Section 2 into training, validation and a in-domain test set with the ratio of 80%, 15% and 5%. Since our prediction target (i.e. step counts) in our data are significantly skewed toward low step count values (refer to Appendix E for the distribution), we apply a 10-fold stratified random sampling validation approach with per-participant uniform density bins, so that on average the validation set and the training set have the same overall and participant-level step count distribution. We emphasize that in this work we focus on the fully held out test set described in Section 2 to provide results covering multiple levels of missing data.

Model Evaluation Results are reported per missing data bin as well as on the overall test set. We report Macro Mean Absolute Error (MAE) and its 95% confident interval. This is the mean over participants in the test set of the mean absolute error per test participant (refer to Appendix D for details). For models where hyper-parameters are selected using the 10 validation splits, we determine the optimal hyper-parameter values using the validation set and average the test predictions of the 10 corresponding models to form a final test prediction.

5 Results

Overall Imputation Results Table 1 shows the overall imputation results (last column) for each method. Methods highlighted in blue have a statistically significantly lower error than other methods in their group ($p < 0.05$). Methods highlighted in red have a statistically significantly lower error

than other methods across all groups ($p < 0.05$). As we can see, our sparse self-attention model achieves the best overall performance and does so with statistical significance relative to all other methods considered.

Imputation Results by Missingness Rates The remaining columns in Table 1 show the imputation results for each missingness rate interval. As we can see, our sparse self-attention model achieves the best performance on all but the highest missing data rate bin. On participants with extremely high missing rates (i.e. $\geq 80\%$), DW+HD Median Fill performs best and is better than our self-attention model with statistical significance ($p < 0.05$). This is likely due to the fact that at over 80% missing data, the context windows for the proposed model will contain relatively few observations. It may be possible to further improve performance for high missing rate bins by using adaptive context window sizes or by adaptively smoothing the model's prediction towards that of simpler models as the volume of observed data in the context window decreases.

6 Conclusions

In this work, we consider the problem of imputing missing step count data collected by wearable devices. To enable this research, we curated a novel dataset consisting of 100 training participants and 500 test participants with more than 5.5 million total hourly step count observations extracted from the *All of Us* dataset. We proposed a customized model for this task based on a novel multi-timescale sparse self attention structure to mitigate the quadratic complexity of the standard dense self-attention mechanism. Our experiments show that the proposed model outperform all of the the baseline approaches and prior state-of-the-art RNN-based models on the fully held-out test data. This includes outperforming all baselines on nearly all missing data rate bins.

7 Acknowledgements

This work was partially supported by National Institutes of Health National Cancer Institute, Office of Behavior and Social Sciences, and National Institute of Biomedical Imaging and Bioengineering through grants U01CA229445 and 1P41EB028242 as well as by a Google Cloud Research Credits Program credit award. The *All of Us* Research Program is supported by the National Institutes of Health, Office of the Director: Regional Medical Centers: 1 OT2 OD026549; 1 OT2 OD026554; 1 OT2 OD026557; 1 OT2 OD026556; 1 OT2 OD026550; 1 OT2 OD 026552; 1 OT2 OD026553; 1 OT2 OD026548; 1 OT2 OD026551; 1 OT2 OD026555; IAA #: AOD 16037; Federally Qualified Health Centers: HHSN 263201600085U; Data and Research Center: 5 U2C OD023196; Biobank: 1 U24 OD023121; The Participant Center: U24 OD023176; Participant Technology Systems Center: 1 U24 OD023163; Communications and Engagement: 3 OT2 OD023205; 3 OT2 OD023206; and Community Partners: 1 OT2 OD025277; 3 OT2 OD025315; 1 OT2 OD025337; 1 OT2 OD025276. In addition, the *All of Us* Research Program would not be possible without the partnership of its participants.

References

Melissa J Azur, Elizabeth A Stuart, Constantine Frangakis, and Philip J Leaf. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research*, 20(1):40–49, 2011.

Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems*, 31, 2018.

Katherine S Hall, Eric T Hyde, David R Bassett, Susan A Carlson, Mercedes R Carnethon, Ulf Ekelund, Kelly R Evenson, Deborah A Galuska, William E Kraus, I-Min Lee, et al. Systematic review of the prospective association of daily step counts with risk of mortality, cardiovascular disease, and dysglycemia. *International Journal of Behavioral Nutrition and Physical Activity*, 17(1):1–14, 2020.

Jong-Hwan Jang, Junggu Choi, Hyun Woong Roh, Sang Joon Son, Chang Hyung Hong, Eun Young Kim, Tae Young Kim, Dukyong Yoon, et al. Deep learning approach for imputation of missing

values in actigraphy data: Algorithm development study. *JMIR mHealth and uHealth*, 8(7): e16113, 2020.

Peng Liao, Kristjan Greenewald, Predrag Klasnja, and Susan Murphy. Personalized heartsteps: A reinforcement learning algorithm for optimizing physical activity. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–22, 2020.

Suwen Lin, Xian Wu, Gonzalo Martinez, and Nitesh V Chawla. Filling missing values on wearable-sensory time series data. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pp. 46–54. SIAM, 2020.

Roderick JA Little. Regression with missing x's: a review. *Journal of the American statistical association*, 87(420):1227–1237, 1992.

Hiral Master, Jeffrey Annis, Shi Huang, Joshua A Beckman, Francis Ratsimbazafy, Kayla Marginean, Robert Carroll, Karthik Natarajan, Frank E Harrell, Dan M Roden, et al. Association of step counts over time with the risk of chronic disease in the all of us research program. *Nature medicine*, 28(11):2301–2308, 2022.

Kelsey R Mayo, Melissa A Basford, Robert J Carroll, Moira Dillon, Heather Fullen, Jesse Leung, Hiral Master, Shimon Rura, Lina Sulieman, Nan Kennedy, et al. The all of us data and research center: creating a secure, scalable, and sustainable ecosystem for biomedical research. *Annual Review of Biomedical Data Science*, 6:443–464, 2023.

Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin. Generative semi-supervised learning for multivariate time series imputation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8983–8991, 2021.

Evangelos Papathomas, Andreas Triantafyllidis, Rafail-Evangelos Mastoras, Dimitrios Giakoumis, Konstantinos Votis, and Dimitrios Tzovaras. A machine learning approach for prediction of sedentary behavior based on daily step counts. In *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 390–394. IEEE, 2021.

Mia S Tackney, Daniel Stahl, Elizabeth Williamson, and James Carpenter. Missing step count data? step away from the expectation–maximization algorithm. *Journal for the Measurement of Physical Behaviour*, 5(4):205–214, 2022.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Jinsung Yoon, William R Zame, and Mihaela van der Schaar. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Transactions on Biomedical Engineering*, 66(5):1477–1490, 2018.

Mo Zhou, Yonatan Mintz, Yoshimi Fukuoka, Ken Goldberg, Elena Flowers, Philip Kaminsky, Alejandro Castillejo, and Anil Aswani. Personalizing mobile fitness apps using reinforcement learning. In *CEUR workshop proceedings*, volume 2068. NIH Public Access, 2018.

A Data Curation and Preprocessing Pipeline

Figure 2 presents the data curation pipeline. Figure 3 presents the data preprocessing pipeline.

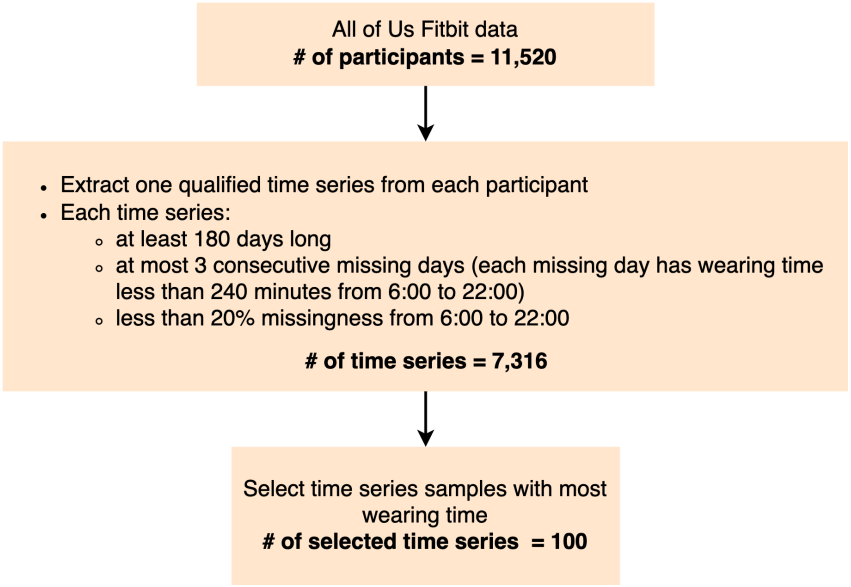


Figure 2: Flow chart for cohort curation

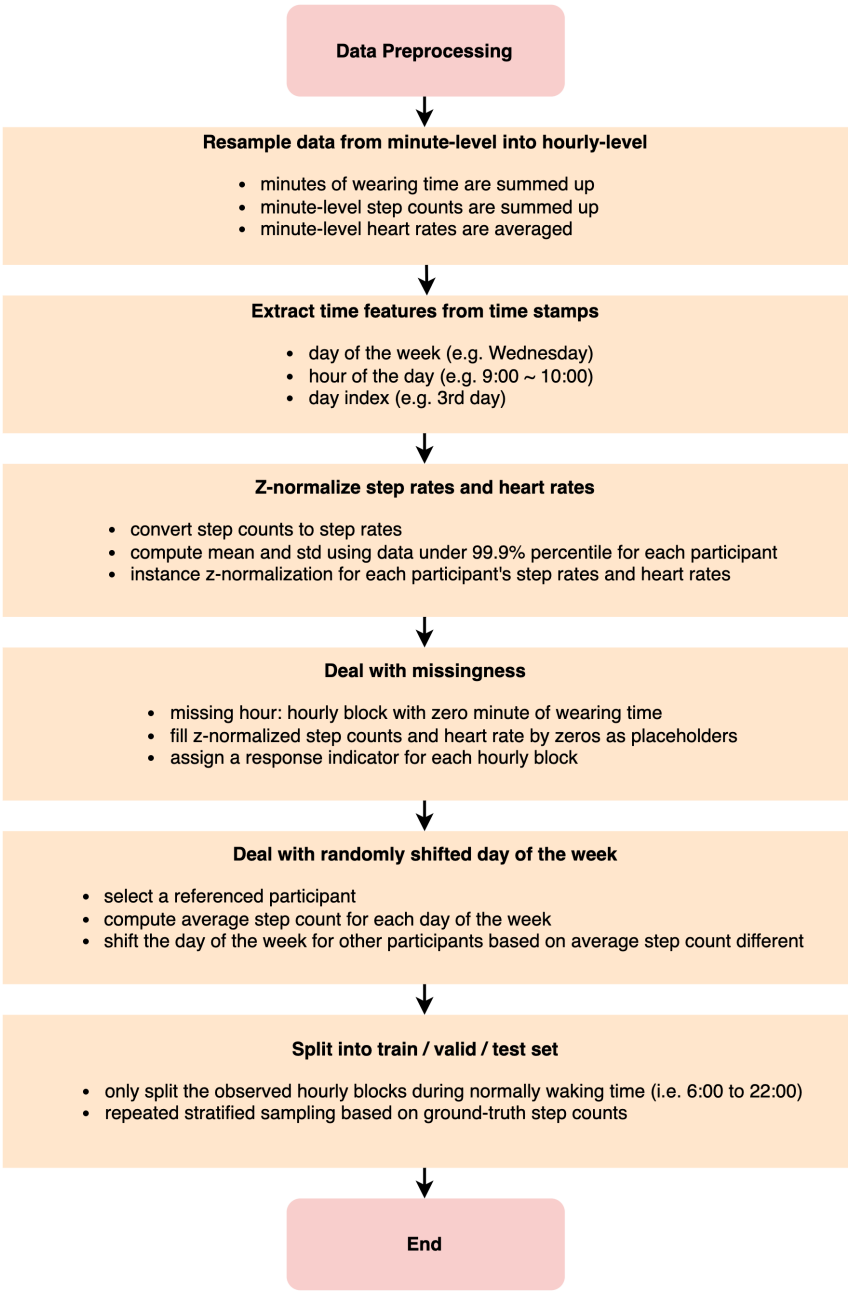


Figure 3: Data preprocessing pipeline

B Comparison between Curated Cohort and All *All of Us* Participants

To compare our curated training cohort with 100 participants and the entire *All of Us* Fitbit dataset with 11,520 participants, we draw the histogram of (1) total number of hourly blocks, (2) total number of valid hourly blocks (i.e. have at least one minute of wearing time), (3) missing rate between 6:00am to 10:00pm in Figure 4, as well as (4) participant level average heart rates and hourly block step rates, (5) hourly step rates for each hourly of the day and (6) missing rate of each hour of the day from 0:00am to 11:00pm in Figure 5.

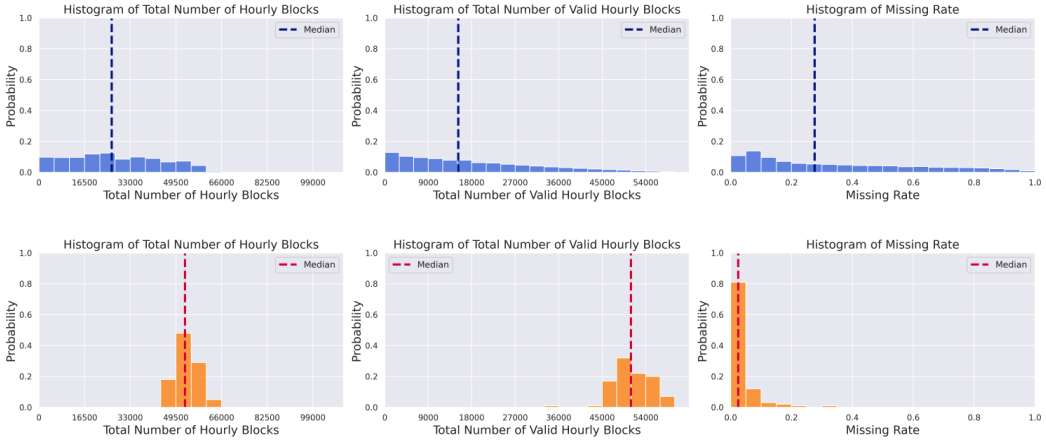


Figure 4: **Blue**: Statistics of the entire *All of Us* Fitbit dataset with 11,520 participants. **Orange**: Statistics of our curated cohort with 100 participants. **First column**: Distribution of the total number of hourly blocks from each participant. **Second column**: The total number of valid hourly blocks (i.e. hourly blocks with non-zero wearing minutes) from each participant. **Third column**: Missing rate (i.e. the number of invalid hourly blocks divided by the total number of hourly blocks), including all hours of the day, not only from 6:00am to 22:00pm, of each participant.

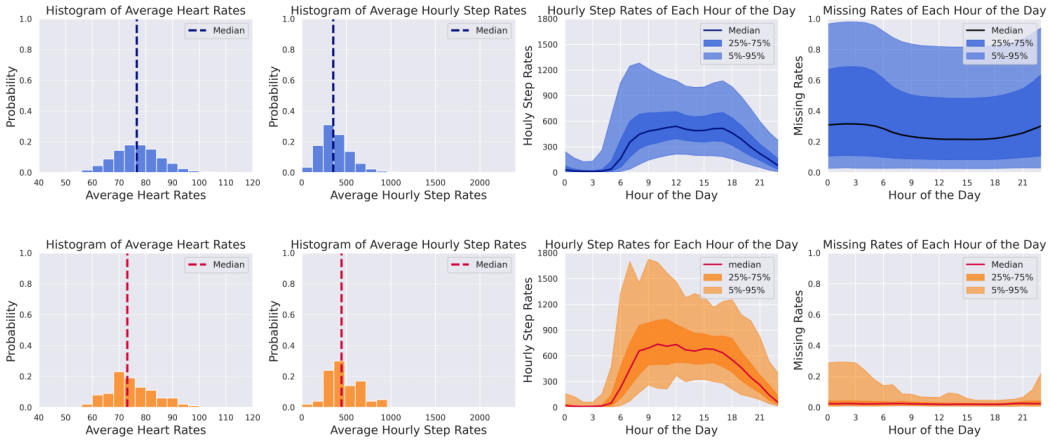


Figure 5: **Blue**: Statistics of the entire *All of Us* Fitbit dataset with 11,520 participants. **Orange**: Statistics of our curated cohort with 100 participants. **First column**: Distribution of average heart rates over all the participants. **Second column**: Distribution of average hourly step rates over all the participants. **Third column**: Distribution of hourly step rates of each hour of the day. **Fourth column**: Missing rate of each hour of the day. The average heart rates and average step rates are computed over all the hourly blocks with non-zero wearing minutes for each participant.

C Model Configurations, Hyperparameters and Training Procedures

In this section, we introduce the details about all the models used in our experiments, including their configurations, hyperparameters and training procedures.

C.1 Multi-Timescale Sparse Self-Attention Model

For the context window, we set the number of hours on each side of the imputing hourly block as 4 and search for the number of weeks on each side within $\{1, 2, 3, 4, 5\}$. We found the best number of weeks is 5, and beyond 5 the performance gain is neglected. Therefore, since the first week is always full, the context window size in our case is $kh = 2 \times 4 + 1 = 9$ and $kw = 2 \times (7 + 4) + 1 = 23$.

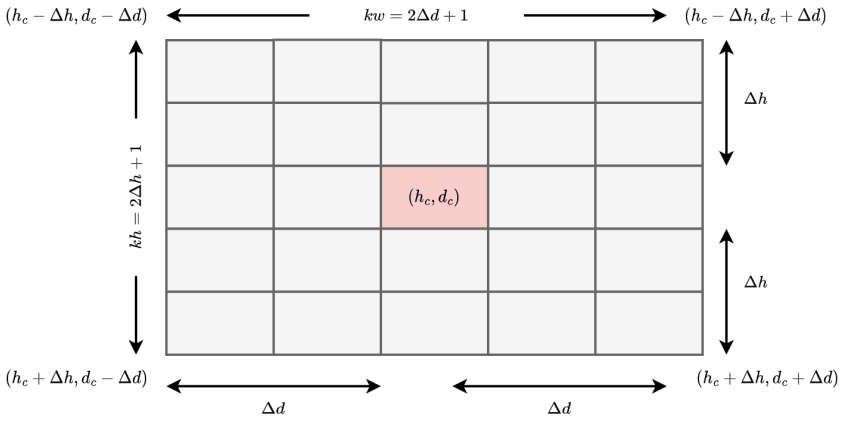


Figure 6: Context window for the multi-timescale sparse self-attention model. In the experiment, we set $\Delta h = 4$ and $\Delta d = 7 + 4 = 11$. Therefore, the context window size is $kh \times kw = (2\Delta h + 1) \times (2\Delta w + 1) = 9 \times 23$.

For local activity profile representations, we set its length as $2W + 1 = 2 \times 72 + 1 = 145$. We fill missing values using the median step rate of day of the week and hour of the day (i.e. DW+HD Median). To prevent data leakage, we set as missing all positions in local activity representations that contain the ground-truth value of the center hourly block of the local context window, which is we are imputing for. Also, the model contains three identical encoding modules: $Conv \rightarrow LayerNorm \rightarrow ReLU \rightarrow AveragePool$, one for each of query, key and value. The configuration of the $Conv$ layer is: out_channels=1, kernel_size=49, stride=1, padding=24 and no bias. The kernel size and stride of $AveragePool$ is set as 7 and 6, respectively.

As for input features, we include one-hot vectors for day of the week and hour of the day indicators in query, key, and value. Furthermore, we observe incorporating normalized step rates of the corresponding hourly block into query and value, along with adding normalized heart rates to value, can enhance the performance. The encoded local activity profile representation is concatenated with other input features. We use three linear layers to map input features to $\mathbf{q}$, $\mathbf{k}$ and v. We search for d_k within $\{4, 8, 16\}$.

During training, we use the Adam optimizer with batch size of 20,000 and search for learning rate within $\{0.1, 0.01, 0.001\}$. The best learning rate is 0.01 and the number of training epochs is set as 30. We conduct early stopping based on validation Micro MAE for each split. Also, average validation Micro MAE over 10 splits is used to choose the best hyperparameters. We train the model using two NVIDIA Tesla T4 GPUs with 32 CPUs and 208 RAM memory within *All of Us* workspace. We implement the model using PyTorch 1.13.1.

C.2 FILLING METHODS

All the filling methods impute missingness on the level of *unnormalized* step rates (i.e. before instance z-normalization). Also, as mentioned in the loss function section, the predicted step rate will go through a non-linearity to limit the upper and lower bound of step rates and will be converted back to step counts using hourly wearing minutes to report the performance. All the filling methods leverage personal statistics, which means all the statistics (e.g. mean and median) are computed specifically from each participant, when filling missing values within that participant. When computing the mean and median w.r.t each factor (e.g. day of the week), all the test hourly blocks are removed to avoid the data leakage. For forward and backward based fillings, during testing, the imputing test hourly block can be filled with the ground-truth value of the other test hourly block if they are contiguous. More details of each filling method are as follows.

- **Zero Fill**: fill missing step rates as zeros. This means when a hourly block has zero wearing minutes, we consider there is no physical activity.
- **Micro Mean Fill**: we define the *micro mean* step rate as the total number of step counts divided by the total number of wearing minutes in all the hourly blocks regarding: each participant, each day of the week, each hour of the day, or each combination of day of the week and hour of the day (e.g. 8am on Mondays). The factors are shared with mean fill and median fill methods. Also, since we only care about 6:00 to 22:00, all the statistics are calculated using data within this period.
- **Mean Fill**: we compute the mean step rate by computing the uniform mean over all the step rates from hourly blocks with non-zero wearing minutes.
- **Median Fill**: we compute the median step rate by calculating the median over all the step rates from hourly blocks with non-zero wearing minutes.
- **Forward and Backward Fill**: different from above filling methods, all the data out of 6:00 to 22:00 are utilized when conducting forward and backward fillings. If there are still missing steps after fillings (e.g. all the hourly blocks are missing before 10:00, then after forward fill, they still remain missing), then they will be filled by the median step rate computed from all the participant non-missing hourly blocks. When computing the average of the forward and backward fill (Avg.F+B), if the results after either forward or backward fill still have the missing step rates, we use the results from the other method to fill.

C.3 REGRESSION IMPUTATION

We use the linear regression model to impute missing step counts for this method. Input features of the linear regression model for each missing hourly block consists of: (1) normalized step rates and heart rate from all the hourly blocks in the context window, except for the center one, which is under imputation, (2) one-hot vectors of day of the week and hour of the day from the currently imputing center hourly block. We fill missing step rates and heart rates in the context window by zeros. The linear regression model has the same context window size, training protocol and loss function as the sparse self-attention model. We set the batch size as 50,000 and search for the learning rate in $\{0.1, 0.01, 0.001, 0.0001\}$ and use Adam optimizer to train the model for 20 epochs with the learning rate of 0.001.

C.4 k-NEAREST NEIGHBORS (kNN) IMPUTATIONS

We search for nearest neighbors for each missing hourly block within all the non-missing hourly data, including those outside 6:00 to 22:00, from the *same* participant data containing the missing hourly block. The feature of hourly blocks is all the step rates within W hours on each side (same as the local activity profile representation in the self-attention model), resulting in the total dimension of $2W + 1$. Missing hourly blocks and the center hourly block within this $2W + 1$ window are filled with the median step rate w.r.t the day of the week and hour of the day (DW+HD median).

Two k-NN variations are tested: (1) uniform averaging (kNN-Uniform) and (2) weighted averaging (kNN-Softmax) based on RBF kernel and the square distance between the imputing hourly block and its neighbors in the $2W + 1$ feature space. Mathematically, let $\mathcal{N}$ denote the set of nearest neighbors of the missing hourly block t' where $|\mathcal{N}| = K$. Let $\mathtt{B}(t', t) : \mathcal{T} \times \mathcal{T} \to \mathbb{R}^{\geq 0}$ be the

function to return the square distance in the feature space between t' and t, then two variations are defined in the following equations:

$$s_{t'} = \frac{1}{K} \sum_{t \in \mathcal{N}} s_t, \qquad s_{t'} = \sum_{t \in \mathcal{N}} \frac{\exp(\mathtt{B}(t', t)/\tau)}{\sum_{u \in \mathcal{N}} \exp(\mathtt{B}(t', u)/\tau)} s_t \tag{3}$$

where s_t is the step rate at time t. Then we use the same procedure to convert it back to the step count. In experiments, we set $W = 72$, which is the same as the local physical activity profile representation dimension in the self attention model. We search for K in {1, 7, 14, 21, 28, 35} for both variations and search for the RBF parameter τ in {0.1, 0.01, 0.001, 0.0001, 0.00001} for kNN-Softmax.

C.5 Multiple Iterative Imputation Methods (Iterative Imputation)

We implement a similar idea as Multiple Imputation with Chained Equations (MICE) method, which uses chained equations to impute every variable conditioned on the others for multiple iterations. However, during the training phase, the algorithm performs a deterministic imputation for each variable instead of probabilistic sampling.

When training the model, we create the same local context window as used in the self-attention model. Each window is centered at each training instance and includes all the z-normalized step rates within it, which serve as the input and output variables for chained equations. Additionally, we incorporate one-hot vectors representing day of the week and hour of the day for the center imputation instance as observed variables, which can only be conditioned on during training. Mathematically, we denote x_i as the normalized step rate of some position i in the context window, and let x_{-i} be normalized step rates from all the other positions in the same context window. Also, θ is the concatenation of two one-hot vectors for day of the week and hour of the day. Therefore, the prediction for x_i is defined as: $\hat{x}_i = \mathtt{g}(x_{-i}, \theta), i \in \{1, \ldots, I\}$, where I is the number of positions in the context window and g is the linear regression in our case: $\mathtt{g}^i(x_i) = w^i x_{-i} + b^i$, where θ is ignored for simplicity.

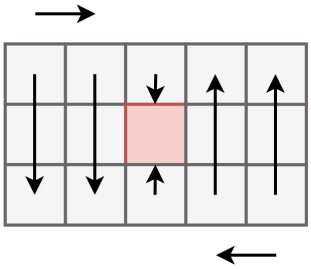
To train our iterative imputation model, we set the initial normalized step rate value belonging to originally missing hourly blocks and test hourly blocks as zeros. We use *sklean.linear_model.SGDRegressor* with *partial_fit* method to train each linear regression model in the chained equation using mini-batch SGD with the batch size of 10,000 for 2 epochs. We initialize the learning rate as 0.001 and use the *invscaling* scheduler implemented by sklean to conduct the learning rate decay. The loss function is epsilon insensitive loss with $\epsilon = 0.01$. The number of imputation iterations is set as 2, so that before the second iteration, the missing values have already been filled by the first iteration. Also, weight decay with the coefficient 0.0001 is leveraged to regularize each linear regression model. We specify the imputation order of context window positions as alternating between the positions from the start and from the end, so that before filling the center hourly block, missing values of all the other positions in the context window have already been imputed. Figure 7 provides an example.

During inference, we perform multiple imputations for each position by sampling from a Gaussian distribution. The mean for the distribution is derived from a trained linear regression, and the variance is estimated from the average squared error between the predictions and the ground-truth values. Also, the sampled step rate will go through the same non-linearity used in other methods to limit its range within zero and 1.5 times maximum step rate from the same participant. In our experiment, we configure the number of samples as 5, and the final prediction for each position is based on the average of them. Algorithm 1 illustrates the inference procedure with multiple imputation.

Step-by-step explanation of Algorithm 1:

Line 0: before illustration, we need to reiterate: direct outputs from the model are *z-normalized step rates* and we evaluate the model on *step counts*. In the following, SC means step counts, sr or SR means *unnormalized* step rates, and x is *normalized* step rate.

Line 1: initialize a variable to record average *step counts* sampled by the algorithm for the center position I in a context window from participant n.



0	6	12	11	5
2	8	14	9	3
4	10	13	7	1

Figure 7: The order of prediction positions within the local context window using iterative imputations. The order is alternating between the start (i.e. upper left) and the end (i.e. lower right). The direction from the start is from top to bottom from left to right, while the direction from the end is from bottom to top, from right to left. The plot on the right gives an example. The number means the order of computation and the same color means the same relative position from the start and from the end.

Line 2 and 3: the outer for-loop is for sampling, the inner for-loop is to iterate every position in the context window with the order in Figure 7. I is the last position, which is the center of the local window.

Line 4: draw a sample for position i from Gaussian distribution with zero mean. The variance is estimated by the mean square error between predicted values from the linear regression g and true z-normalized step rate values at position i over the entire validation or test set. The variance is shared across position i in local context windows from all the participants.

Line 5: use the linear regression and the sampled noise to get the normalized step rate at position i in a context window from participant n. We limit the range of predicted *unnormalized* step rates between zero and 1.5 fold maximum step rate in the training dataset of participant n. Here, we convert this range into the *normalized* step rate level using personalized mean $\mu_{SR}^{(n)}$ and standard deviation $\sigma_{SR}^{(n)}$ derived from all the *unnormalized* step rates from participant n.

Line 7: after predicting the normalized step rate for poisition I, we convert it back to the unnormalized step rate using the same personalized mean and standard deviation as above.

Line 8: Convert the normalized step rate into the step count within the same hourly block using wearing minutes VM. Record this sampled step count in the current iteration for later averaging.

Algorithm 1: Inference with Multiple Imputations

1 $SC_I^{(n)} \leftarrow 0$;
2 **for** $s \leftarrow 1$ **to** S **do**
3 **for** $i \leftarrow 1$ **to** I **do**
4 $\epsilon_{i,s} \sim \mathcal{N}(0, \sigma_i^2)$; /* $\sigma_i^2 \approx \frac{1}{m_i} \sum_{m=1}^{M_i} \|\hat{x}_{i,m} - x_{i,m}^{true}\|^2$ */
5 $\hat{x}_{i,s}^{(n)} \leftarrow \mathtt{P}(w^i x_{-i}^{(n)} + b^i + \epsilon_{i,s})$;
 /* $\mathtt{P}(x^{(n)}) = \max(-\frac{\mu_{SR}^{(n)}}{\sigma_{SR}^{(n)}}, \min(x, \frac{1.5 \max sr^{(n),train} - \mu_{SR}^{(n)}}{\sigma_{SR}^{(n)}}))$ */
6 **end**
7 $\hat{sr}_I^{(n)} = \hat{x}_I^{(n)} \times \sigma_{SR}^{(n)} + \mu_{SR}^{(n)}$;
8 $SC_I \leftarrow SC_I + \frac{1}{S} \hat{sr}_I \times VM_I$
9 **end**

C.6 Convolutional Denoise Autoencoder (CNN-DAE)

We use the symmetric encoder-decoder architecture to implement CNN-DAE. The encoder consists of three 1D convolutional layers, each followed by BatchNorm and ReLU activation. Similarly, the decoder includes three 1D transposed convolutional layers, with the first two layers being followed

by Batch Normalization and ReLU activation. The configurations of convolutional and transposed convolutional layers are in Table 2.

Table 2: Configurations for Convolutional and Transpose Convolutional Layers in CNN-DAE

Layer	Input Channels	Output Channels	Kernel Size	Stride	Padding
Conv1	2	4	31	2	11
Conv2	4	8	20	2	9
Conv3	8	16	10	2	4
TransConv1	16	8	10	2	4
TransConv2	8	4	20	2	9
TransConv3	4	2	31	2	11

For the input, we include both z-normalized step rates and heart rates within the same local context window used in other model-based methods. To represent the data in a 1D format with two variables, we flatten the context window column by column.

During training, we set as zeros all the hourly blocks belonging to (1) the center of each context window (i.e. what we are actually imputing for) (2) the test set (3) positions which are originally missing in the participant data. We calculate the MAE loss between reconstructed and ground-truth z-normalized step rates of all the hourly blocks belonging to the training set in the context window.

Also, we consider the heart rate reconstruction as the auxiliary task and minimize the combined reconstruction MAE loss on step rates and heart rates as total loss $=$ MAE(step rates) $+$ λMAE(heart rates). We search for λ within $\{0.0, 0.5, 1.0\}$. However, since our final goal is to reconstruct the step count of the center hourly block of each local context window, we compute the MAE on step counts over all the center hourly blocks belonging to the validation set to select the best hyperparameters and conduct the early stopping.

We use Adam optimizer and set the batch size as 50,000 to train the model of each split for 20 epochs. The learning rate is searched within $\{0.1, 0.01, 0.001\}$.

C.7 BRITS

We adhere to the settings outlined in the original paper, utilizing LSTM as the RNN architecture. Input features are derived from hourly blocks within the same multi-timescale context window, ensuring a fair comparison with other models. The context window is chronologically flattened, enabling the RNN model to process information sequentially.

We impute both heart rates and step rates at each time step, utilizing the BRITS loss, which is a combination of the reconstruction loss (i.e., MAE) and the consistency loss (i.e., ensuring forward and backward LSTM predictions consistent at each step). Notably, our BRITS loss is at the *normalized step rate* and *normalized heart rate* level.

Validation involves monitoring the Micro MAE, computed on the *step count* level across all the validation hourly blocks. This choice is reflective of our ultimate task, which centers on imputing step counts rather than step rates or normalized step rates. Early stopping is implemented, and the best hyperparameters are selected based on the optimal validation loss.

Training the BRITS model spans 30 epochs with a batch size of 10,000 and a learning rate of 0.01. The LSTM hidden state dimension is searched within $\{4, 8, 16, 32\}$.

C.8 MRNN

MRNN comprises an interpolation block and an imputation block. In the interpolation block, we employ two bidirectional-GRU models to fill in missing values in step rates and heart rates. Given that the interpolation block calculates missing values within each data stream, day of the week and hour of the day one-hot vectors, which are always observed, are excluded from the model input. In the imputation block, as MRNN handles missing values based on information across different variables, we include day of the week and hour of the day features in the input features.

During training, the visibility of train/valid and test sets, along with the context window, aligns with that in the self-attention model. Like BRITS, the context window is flattened in chronological order for RNN processing. To keep consistent with other models, we employ Mean Absolute Error (MAE) instead of Mean Squared Error (MSE) for model training, different from the original paper.

We train MRNN for 40 epochs, utilizing a batch size of 20,000 and a learning rate of 0.01. The GRU hidden state dimension is searched within $\{4, 8, 16, 32\}$.

C.9 USGAN

We employ the BRITS model as the generator and a bidirectional GRU model as the discriminator. The generator configurations align with those outlined in Section C.7. As our data does not have labels for each time series, we omit the classifier mentioned in the original paper. In contrast to the original implementation, which updates the discriminator five times after each generator update, we observe that updating the discriminator only once results in more stable training and improved performance in our case.

We train the USGAN model for 30 epochs with a batch size of 10,000 and a learning rate of 0.01. The RNN hidden state dimensions for both the generator and discriminator are explored within $\{4, 8, 16, 32, 64\}$. Additionally, we search for the weight of the discriminator loss during training, which balances it and BRITS loss, within $\{0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$.

C.10 Computing Infrastructure

We use Faiss-cpu [2] to search for k-nearest neighbors. Iterative imputation method is implemented using scikit-learn v1.0.2. Regression Imputation, CNN-DAE, Self-Attention, BRITS, MRNN and USGAN are implemented using PyTorch v1.13.1 and trained with NVIDIA Tesla T4 GPUs provided in the *All of Us* workspace.

D Formal Definitions of Evaluation Metrics

Let $m_n \in \mathcal{M}^{(n)}$ indicates the index of a single hourly block to be imputed from participant n in either train, valid or test set. Then, Micro and Macro MAE and RMSE are defined as below:

$$\textit{Micro MAE} = \frac{1}{\sum_n \sum_{m_n} 1} \sum_{n=1}^{N} \sum_{m_n=1}^{|\mathcal{M}^{(n)}|} AE_{n,m_n}$$

$$\textit{Macro MAE} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{|\mathcal{M}^{(n)}|} \sum_{m_n=1}^{|\mathcal{M}^{(n)}|} AE_{n,m_n}$$

$$\textit{Micro RMSE} = \sqrt{\frac{1}{\sum_n \sum_{m_n} 1} \sum_{n=1}^{N} \sum_{m_n=1}^{|\mathcal{M}^{(n)}|} SE_{n,m_n}}$$

$$\textit{Macro RMSE} = \frac{1}{N} \sum_{n=1}^{N} \sqrt{\frac{1}{|\mathcal{M}^{(n)}|} \sum_{m_n=1}^{|\mathcal{M}^{(n)}|} SE_{n,m_n}}$$

where N is the number of participants in the dataset and $|\mathcal{M}^{(n)}|$ is the number of imputed hourly blocks from participant n.

[2] https://github.com/facebookresearch/faiss

E STEP COUNT DISTRIBUTION OF THE TRAINING DATA SET

Figure 8 shows the distribution of our imputation target variable (i.e. step count) between 6:00am to 10:00pm for the training cohort with 100 participants. It is clear that the distribution is significantly skewed towards low step count values.

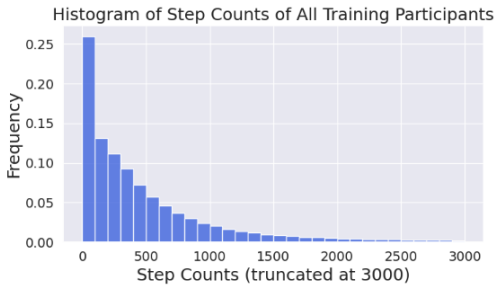


Figure 8: Histogram of observed hourly step counts between 6:00am and 10:00pm for the 100 training participants. Step count values are truncated at $3,000$ for a better illustration.