Tiles: LEARNING MULTIPLE TIME-FREQUENCY LATENTS FOR DIVERSE ACCELEROMETER TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Wearable accelerometers support many applications, but the signal characteristics relevant to each task differ greatly. Existing self-supervised learning methods often encourage invariance to selected signal characteristics or aggregate each window into a single embedding, potentially discarding information required by some tasks. We present *Tiles*, a self-supervised method for learning transferable representations of accelerometer signals. *Tiles* decomposes raw signals into frequency bands and uses iterative competitive cross-attention with learned queries to organize temporal and spectral features into multiple time-frequency latents. Two auxiliary reconstruction losses further encourage the latents to preserve local structure and high-frequency variations. To apply the learned representation to a target task, we keep the encoder fixed and train a lightweight prediction head that combines the latents. We evaluate *Tiles* on 11 classification and regression tasks that broaden downstream accelerometer SSL evaluation beyond standard human activity recognition. *Tiles* achieves the best result among the compared self-supervised methods on 8 tasks and performs comparably on the rest. Analysis of task-head weights and performance under individual-latent masking further shows that tasks rely on different combinations of the learned latents.

1 INTRODUCTION

Self-supervised learning (SSL) enables representations to be learned from large collections of unlabeled wearable sensor data, reducing reliance on task-specific annotations. This is particularly useful for accelerometers, which are widely used in applications including activity recognition, gesture recognition, gait analysis, and sports monitoring. However, most existing accelerometer SSL studies focus primarily on common Human Activity Recognition (HAR) tasks, such as exercise and locomotion classification (Logacjov, 2024). Tasks beyond this setting can depend on substantially different signal characteristics. For example, gesture and gait tasks may rely on orientation or periodic structure, while sports and physical-quantity estimation tasks may depend on signal magnitude or high-frequency variations. A representation intended for transfer across these tasks must therefore preserve multiple types of motion information.

Many SSL objectives are designed around assumptions about which signal characteristics should be preserved or ignored. Augmentation-based methods such as SimCLR (Tang et al., 2020) encourage representations to be invariant or sensitive to selected transformations. For example, rotation augmentation can improve robustness to device placement, benefiting activity recognition (Xu et al., 2025). The same invariance may remove orientation information that is important for tasks such as gait analysis or gesture recognition. The appropriate invariances can therefore depend on the downstream task, which is unknown during pre-training. Reconstruction-based approaches reduce the need to specify these invariances by encouraging the representation to preserve information from the input signal. However, standard autoencoders compress each window into a single global embedding, which can entangle the different temporal and spectral patterns required across tasks. Thus, pre-training must preserve a broad range of temporal and spectral patterns, while the representation must allow downstream tasks to access and combine them differently.

We propose *Tiles*, a self-supervised approach that represents each accelerometer signal using multiple time-frequency latents. To expose motion patterns across frequency ranges while retaining temporal structure, *Tiles* first decomposes the raw signal into frequency bands and processes each with

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

a band-specific encoder. A set of learned queries shared across inputs then iteratively aggregates the encoded features through competitive cross-attention, producing latents that draw information from different bands and temporal positions. During pre-training, the latents are jointly decoded to reconstruct the frequency-decomposed input. Reconstructing these bands with pointwise mean squared error can favor smoother outputs that do not fully preserve rapid high-frequency patterns. We therefore introduce two auxiliary reconstruction losses based on the Structural Similarity Index Measure (SSIM) and the Multi-Scale Short-Time Fourier Transform (MS-STFT) to emphasize local signal structure and spectral patterns across multiple scales. For each downstream task, the representation encoder remains fixed and a small task-specific head learns how to combine the latents.

We evaluate *Tiles* on 11 downstream tasks spanning classification and regression across daily activities, sports, gestures, transportation, and physical-quantity estimation. These tasks extend beyond the activity-classification benchmarks typically considered in accelerometer SSL and involve signal patterns that vary in temporal scale, frequency content, magnitude, and dependence on sensor orientation. *Tiles* achieves the best result among the compared SSL methods on 8 tasks and performs comparably on the remaining tasks. Task-head weights and individual-latent masking results further show that tasks rely on different combinations of the learned latents, supporting the use of multi-latent representations for accelerometer tasks with different signal requirements.

Our key contributions are as follows:
1. We propose *Tiles*, a self-supervised approach that learns a set of time-frequency latents from accelerometer signals, allowing task-specific heads to draw on different temporal and spectral features within a shared pre-trained representation.
2. We curate 11 classification and regression tasks from publicly available datasets to broaden downstream accelerometer SSL evaluation beyond standard activity classification.
3. We demonstrate that *Tiles* performs consistently across tasks with different signal requirements and that different tasks rely on different combinations of the learned latents.
4. We release the implementation of *Tiles* and the downstream evaluation pipeline to support reproducibility and future research.

2 Related Work

Self-supervised learning for accelerometer signals: Self-supervised learning methods for accelerometer signals commonly use transformation prediction, contrastive learning, or reconstruction. Transformation-prediction methods train an encoder to identify augmentations applied to the input (Saeed et al., 2019; Yuan et al., 2024), while contrastive methods construct positive and negative pairs, often using augmented views of the same signal as positives (Tang et al., 2020). REBAR learns signal similarity through cross-signal reconstruction (Xu et al., 2024), and RelCon models relative differences using soft positive and negative pairs (Xu et al., 2025). These objectives depend on assumptions about which signal characteristics should be preserved across related inputs. Reconstruction-based methods include autoencoders (Haresamudram et al., 2019) and masked autoencoders (Haresamudram et al., 2020), which recover selected portions of the input. This objective is widely used across different settings, including Transformer-based encoding of raw signals in LIMU-BERT (Xu et al., 2021), accelerometer spectrograms in SelfPAB (Logacjov et al., 2024), and minute-level multimodal wearable features in LSM (Narayanswamy et al., 2025). Reconstruction reduces the need to define relationships between different input signals by encouraging the representation to preserve information from each input. When these methods use a single global embedding for downstream prediction, each task-specific head must extract the temporal and spectral information relevant to its task. *Tiles* decomposes the input into frequency bands and aggregates the encoded features into multiple time-frequency latents, allowing task-specific heads to combine different parts of the representation.

Multiple representations and latent aggregation: Several methods learn multiple embeddings from an input to represent different components or characteristics separately. Taghanaki et al. train two contrastive pipelines for accelerometer-based HAR, one using time-frequency scalograms and the other raw accelerometer signals (Taghanaki et al., 2023). HiMAE uses a hierarchical masked autoencoder whose intermediate layers produce embeddings at multiple temporal resolutions from wearable physiological time series (Lee et al., 2026). TSPulse uses masked reconstruction across signal domains and abstraction levels to learn temporal, spectral, and semantic embedding views for

general time-series data (Ekambaram et al., 2026). In the image domain, Slot Attention decomposes visual scenes into multiple object-centric latent representations by allowing a set of slot vectors to attend to and compete over the input features, learning to associate objects with different slots without predefining what each represents (Locatello et al., 2020). BO-QSA extends this by replacing stochastic slot initialization with learned queries and introducing bi-level optimization, improving object segmentation and reconstruction on complex scenes (Jia et al., 2023). *Tiles* uses learned-query competitive cross-attention to aggregate frequency-decomposed accelerometer features into multiple time-frequency latents. These latents are learned jointly through reconstruction and combined by task-specific heads, without assigning them predefined temporal, spectral, or semantic roles.

Downstream task diversity: Accelerometer SSL methods have commonly been evaluated on human activity recognition tasks, such as exercises or daily routines (Koskimäki et al., 2017; Blunck et al., 2015). Some studies consider specialized tasks like shorter actions or freezing of gait (Scholl et al., 2015; Bachlin et al., 2009), while RelCon additionally evaluates regression tasks for gait metrics such as double support time (Xu et al., 2025). Yet in practice, accelerometers and IMUs have been used for a far broader range of tasks, including pose estimation (Mollyn et al., 2023), sports action classification (Park et al., 2024), hand gesture recognition (Zhang et al., 2021), jump height estimation (Villa et al., 2024), and user identification through gait (Liberman et al., 2024). Recent wearable foundation models have considered broader downstream settings, including activity and health tasks using multimodal sensor-language representations in SensorLM (Zhang et al., 2025), 35 health, lifestyle, and demographic tasks in SensorFM (Narayanswamy et al., 2026), 57 health-related tasks using higher-level wearable behavioral data (Erturk et al., 2025), and 20 classification and regression tasks using PPG (Pillai et al., 2025). Broad evaluations remain uncommon for accelerometer SSL, where downstream evaluation typically focuses on activity recognition tasks. We instead evaluate *Tiles* on 11 classification and regression tasks spanning gestures, sports, transportation, and physical-quantity estimation, directly testing whether a single pretrained accelerometer representation can generalize across tasks with substantially different signal requirements.

3 Methodology

3.1 Approach Overview

Tiles represents an accelerometer window using multiple time-frequency latents. As illustrated in Figure 1, the input signal is first decomposed into three frequency bands, each of which is processed by a band-specific encoder to produce temporal features. These features are aggregated through iterative competitive cross-attention initialized by a set of learned queries shared across inputs. We

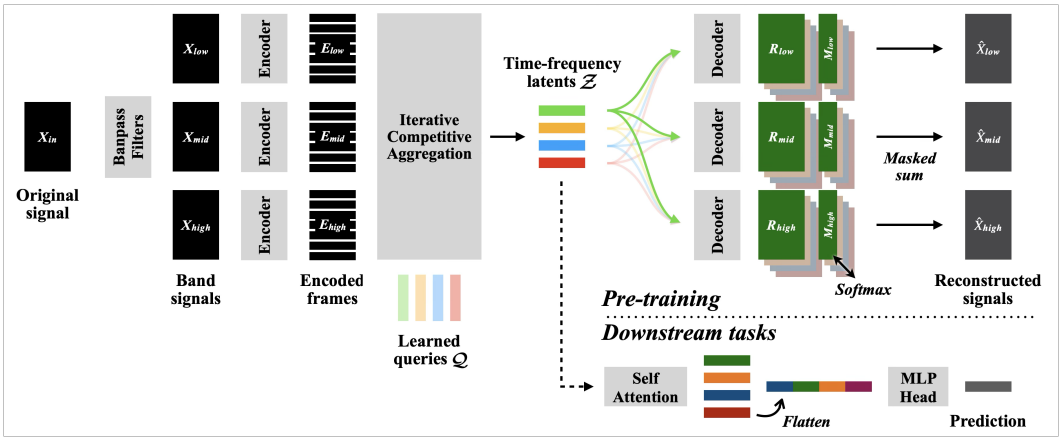

Figure 1: Architecture of *Tiles*. The input is decomposed into three frequency bands and processed by band-specific encoders. Iterative competitive cross-attention aggregates the encoded features into multiple time-frequency latents. During pre-training, the latents are jointly decoded to reconstruct each frequency band. For downstream tasks, a lightweight task-specific head combines the latents.

refer to the resulting outputs as time-frequency latents because they are formed by aggregating features indexed by both temporal position and frequency band. During pre-training, the latents are jointly decoded to reconstruct the frequency-decomposed input using a band-specific combination of MSE, SSIM, and MS-STFT losses. For downstream evaluation, the reconstruction decoders are removed, the pretrained frequency-band encoders and aggregation module are kept fixed, and a lightweight task-specific head is trained to combine the latents for prediction.

3.2 Frequency-Decomposed Signal Encoding

Motion signals contain information across different frequency ranges, and frequency-filtered signals or spectrograms are commonly used to separate these components (Yang & Hsu, 2010). We apply fourth-order Butterworth filters to decompose the input signal X_{in} into three bands: 0–1 Hz, 1–4 Hz, and above 4 Hz, following prior work on human movement frequencies (Fridolfsson et al., 2019). These bands provide a coarse separation of signal characteristics associated with different forms of movement. For example, low-frequency components can reflect gradual changes in posture and device orientation, mid-frequency components can capture periodic movements such as walking or running, and high-frequency components can represent rapid impacts such as foot strikes and other short-duration variations. While a finer decomposition could provide greater frequency resolution, our experiments show that three bands provide a suitable balance between signal detail and computational cost. The effect of the number of bands is examined in Appendix A.5.2.

Each frequency band, denoted by X_{low}, X_{mid}, and X_{high}, is processed by a separate ResNet-style encoder with unshared weights that preserves the temporal length of the signal. We add soft positional embeddings obtained through a learned linear projection of normalized temporal and frequency-band coordinates, allowing the subsequent attention module to distinguish the temporal position and frequency range of each encoded feature. The resulting features are concatenated to form

$$E = [E_{\text{low}}; E_{\text{mid}}; E_{\text{high}}] \in \mathbb{R}^{N \times D_{\text{enc}}},$$

where $N = 3T$, T is the temporal length, and D_{enc} is the encoded feature dimension.

3.3 Iterative Competitive Aggregation

The encoded features are aggregated into K time-frequency latents using iterative competitive cross-attention adapted from Slot Attention (Locatello et al., 2020). Slot Attention samples its initial slot vectors from a learned normal distribution each forward pass. Because these vectors act as queries, resampling them can change how information is assigned across the output slots. This variability is undesirable if the outputs are to be used at fixed positions by downstream prediction heads.

We instead define a set of K learned queries

$$\mathcal{Q} = \{q_k\}_{k=1}^{K}, \qquad q_k \in \mathbb{R}^{D_{\text{latent}}},$$

which are optimized during pre-training and shared across inputs, removing the variability introduced by stochastic initialization. Each query initializes the same latent position across forward passes, encouraging a more stable assignment of encoded features across the resulting latents.

At the first iteration, the learned queries provide the initial states of the K latents. At each iteration, the current latents serve as queries, while the encoded features E serve as keys and values. Cross-attention produces a score matrix of size $K \times N$. A softmax is applied across the K latents for each encoded feature, causing the latents to compete for its contribution. The attention weights are used to calculate an update for each latent from the encoded features. Following Slot Attention, each update is processed by a GRU and an MLP. This procedure is repeated for U iterations, producing

$$\mathcal{Z} = \{z_k\}_{k=1}^{K}, \qquad z_k \in \mathbb{R}^{D_{\text{latent}}}.$$

Because the encoded features span every temporal position and frequency band, the latents can aggregate information across both dimensions. The full pseudocode is provided in Appendix A.3.1.

3.4 Reconstruction-Based Pre-training

During pre-training, each time-frequency latent is decoded into a signal reconstruction and temporal mask for each frequency band using band-specific ResNet decoders. The masks are normalized

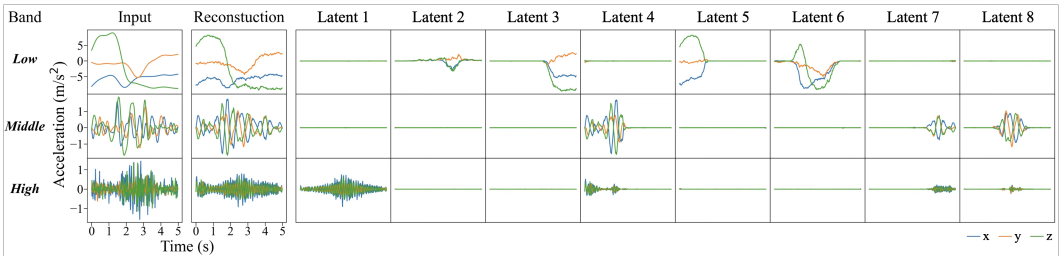

Figure 2: Per-latent contributions to reconstruction of frequency-decomposed inputs.

across latents at each time step, and the final reconstruction is obtained as a weighted sum of the per-latent reconstructions. As shown in Figure 2, different latents contribute to different temporal regions and frequency bands. This joint objective encourages the latents to distribute information while collectively preserving the frequency-decomposed input.

Mean squared error (MSE) is commonly used for reconstruction-based SSL with IMU signals. However, reconstructing the decomposed bands with MSE can favor smoother outputs that preserve low-frequency trends while losing finer motion patterns, because small temporal shifts in high-frequency peaks produce large errors (Cuturi & Blondel, 2017; Mathieu et al., 2016). We therefore add two auxiliary reconstruction losses to emphasize local signal structure and high-frequency variations.

The first auxiliary loss uses the Structural Similarity Index Measure (SSIM), originally developed to measure similarity in local image structure through statistics of neighboring pixels (Zhao et al., 2016). Accelerometer signals similarly contain local waveform patterns defined by relationships among neighboring samples. We therefore adapt SSIM to raw signals by computing local means, variances, and covariance within sliding windows. The loss is defined as

$$\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(x, y),$$
$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)},$$

where x and y denote the original and reconstructed signals, μ_x and μ_y their local means, σ_x^2 and σ_y^2 their local variances, σ_{xy} their local covariance, and C_1 and C_2 are stabilizing constants.

The second auxiliary loss is the Multi-Scale Short-Time Fourier Transform (MS-STFT) loss, used to preserve spectral structure in neural audio reconstruction (Défossez et al., 2023). Accelerometer signals similarly contain motion patterns spanning different frequencies and durations. We therefore compare the magnitude spectra of the original and reconstructed signals at multiple window sizes. At each scale, we sum the MAE and MSE between the spectra and average across scales:

$$\mathcal{L}_{\text{MS-STFT}}(x, y) = \frac{1}{|\mathcal{F}|} \sum_{n_{\text{fft}} \in \mathcal{F}} \Big[\text{MAE}\big(|\,\text{STFT}_{n_{\text{fft}}}(x)|, |\,\text{STFT}_{n_{\text{fft}}}(y)|\big) + \text{MSE}\big(|\,\text{STFT}_{n_{\text{fft}}}(x)|, |\,\text{STFT}_{n_{\text{fft}}}(y)|\big) \Big],$$
$$\mathcal{F} = \{16, 32, 64, 128\}.$$

Because the frequency bands contain different signal patterns, we use a different combination of losses for each band. The low-frequency band uses $\mathcal{L}_{\text{MSE}}$ to preserve absolute signal levels and slow variations. The mid-frequency band uses all three losses, while the high-frequency band uses $\mathcal{L}_{\text{SSIM}}$ and $\mathcal{L}_{\text{MS-STFT}}$ to emphasize local and spectral structure. For each band b,

$$\mathcal{L}_b = \alpha_b \mathcal{L}_{\text{MSE}}(X_b, \hat{X}_b) + \beta_b \mathcal{L}_{\text{SSIM}}(X_b, \hat{X}_b) + \gamma_b \mathcal{L}_{\text{MS-STFT}}(X_b, \hat{X}_b),$$

where α_b, β_b, and γ_b are band-specific weights. The complete pre-training objective is

$$\mathcal{L}_{\text{pretrain}} = \sum_{b \in \{\text{low,mid,high}\}} \mathcal{L}_b.$$

3.5 Downstream Adaptation

For downstream tasks, the reconstruction decoders are removed and the pretrained frequency-band encoders and aggregation module are kept fixed. The K time-frequency latents are processed by a single multi-head self-attention layer to model relationships among them. The resulting latents are flattened and passed to an MLP prediction head. Only this task-specific head is trained for each downstream dataset. This design allows each task to combine information from the shared latents differently without modifying the pretrained representation.

4 Experiments

4.1 Pre-training

We use CAPTURE-24 (Chan et al., 2024), the largest publicly available wrist-worn accelerometer dataset, for self-supervised pre-training. The pre-training set includes approximately 2,500 hours of data from 151 participants, with 120 used for training and the remaining 31 for validation. Following Yuan et al. (2024), we apply weighted sampling during training, where windows are sampled in proportion to their standard deviation. This increases the representation of high-movement windows and prevents long periods of low activity from dominating the sampled batches.

The model takes 5-second windows of raw three-axis accelerometer data sampled at 50 Hz as input. We train for up to 500 epochs using four V100 GPUs, with early stopping when the validation loss does not improve for 40 consecutive epochs. Each band-specific encoder and decoder follows a ResNet-style architecture consisting of four residual blocks constructed using one-dimensional convolutions with a kernel size of 3. We use eight time-frequency latents with dimension 32, producing a 256-dimensional representation when concatenated. The pretrained representation model contains 4.8 million parameters. Further implementation details are provided in Appendix A.3.

4.2 Downstream Evaluation

We evaluate *Tiles* on 11 classification and regression tasks that broaden accelerometer SSL evaluation beyond the activity-classification benchmarks typically considered in prior work. Table 1 provides an overview, with dataset processing and task construction detailed in Appendix A.2. These tasks are curated from publicly available datasets and span daily activities, sports, gestures, transportation, and physical-quantity estimation. They differ in temporal scale, frequency content, signal magnitude, and dependence on sensor orientation.

We compare *Tiles* with five competitive self-supervised learning methods for accelerometer signals. These include an Autoencoder (Haresamudram et al., 2019), a Masked Autoencoder (Haresamudram et al., 2020), SimCLR (Tang et al., 2020), Augmentation Prediction (Yuan et al., 2024), and Relative Contrastive Learning (RelCon) (Xu et al., 2025). The first three provide widely used reconstruction

Table 1: Downstream tasks and datasets used in our evaluation. For classification tasks, the number of classes is indicated. Details on dataset processing and task construction are in Appendix A.2.

	Task name	Task description	# Subjects	# Samples	Dataset
Classification	Basketball	Basketball actions (e.g., layup, shoot). [C=5]	24	9781	Hoelzemann et al. (2023)
	Cooking	Cooking actions (e.g., turn stove on, wipe pan). [C=14]	17	237	Arakawa et al. (2023)
	Locomotion	Exercise and locomotion (e.g., walking, jogging). [C=8]	15	11527	Sztyler & Stuckenschmidt (2016)
	TennisShot	Tennis shot measured from the non-dominant arm. [C=6]	20	6000	Park et al. (2024)
	Transportation	Transportation mode that participant is in. [C=6]	3	40059	Gjoreski et al. (2018)
	Workouts	Outdoor workouts (e.g., burpees, lunges). [C=18]	24	8008	Bock et al. (2024)
	Writing	Letters/characters written on a device. [C=39]	10	11685	Roggen (2022)
Regression	JumpPower	Power of vertical jump	73	1128	White et al. (2022)
	NumSteps	Number of steps taken during the given window	25	492	Santos et al. (2022)
	ThrowSpeed	Speed of handball throw	4	105	Gençoğlu & Gümüş (2020)
	WalkDistance	Distance walked during the given window	25	492	Santos et al. (2022)

and contrastive learning baselines. Augmentation Prediction has remained competitive in recent accelerometer SSL evaluations, while RelCon is a recent relative contrastive approach that reports improvements over prior methods. We also include a supervised baseline for which the full encoder is trained from scratch on each downstream task.

All SSL methods are pre-trained on CAPTURE-24. We reimplement each baseline using a 12-block ResNet-style encoder matched to *Tiles* in parameter count and representation dimension, with method-specific objectives and augmentations following the corresponding paper. After pre-training, each representation model is kept fixed and only a task-specific prediction head is trained. For each baseline, we use an MLP prediction head whose parameter count is matched to that of the *Tiles* task-specific head. The supervised baseline is instead trained end-to-end. Implementation details, hyperparameters, and parameter counts are provided in Appendix A.4.

All tasks use five-fold cross-validation with three training folds, one validation fold, and one test fold, using identical assignments across methods. Folds are participant-disjoint when possible, while ThrowSpeed and Transportation use non-overlapping trial- and segment-level folds, respectively, as detailed in Appendix A.2. We use cross-entropy loss for classification, with class weighting when needed, and mean squared error for regression. Models are trained for up to 200 epochs with an early-stopping patience of 10 epochs, and we report the mean and standard deviation across folds.

5 Results

5.1 Downstream Task Performance

Table 2 presents the performance of the compared methods across the 11 downstream tasks. *Tiles* achieves the best or tied-best SSL result on 8 tasks. On the remaining tasks, it remains close to the best SSL result, with differences of 0.7 points on Transportation, 0.05 RMSE on NumSteps, and 0.02 RMSE on WalkDistance. *Tiles* also outperforms the supervised baseline on four classification tasks, showing the benefit of self-supervised pre-training when downstream annotations are limited.

Across all 11 tasks, *Tiles* performs consistently without the large task-dependent drops observed for several baselines. Individual baselines can perform well on particular tasks but degrade substantially on others. For example, Augmentation Prediction performs best on Transportation but substantially worse on Cooking. This difference may reflect its use of rotation invariance, which can provide robustness to device orientation for transportation recognition but suppress orientation changes relevant to wrist-based cooking actions. Similarly, SimCLR achieves best performance on NumSteps but performs substantially worse on JumpPower. These results suggest that individual SSL objectives can favor particular signal characteristics, limiting their performance when downstream tasks have different requirements. By preserving information across frequency ranges and temporal po-

Table 2: Downstream results for *Tiles*, a supervised baseline, and five SSL baselines. Classification performance is reported as macro-F1 in percent, and regression performance as RMSE. Values are mean ± standard deviation across five folds. The best SSL result for each task is bolded.

Tasks	Supervised	Autoencoder	Masked Autoencoder	Augmentation Prediction	SimCLR	RelCon	*Tiles*
Classification – Macro-F1 (↑)							
Basketball	62.2 ± 3.4	59.6 ± 3.3	57.1 ± 2.8	49.7 ± 3.4	54.9 ± 4.1	57.1 ± 2.8	$\mathbf{67.1 \pm 4.3}$
Cooking	60.6 ± 4.4	53.2 ± 4.7	50.9 ± 6.7	30.2 ± 9.2	53.3 ± 7.2	51.6 ± 6.1	$\mathbf{55.5 \pm 3.6}$
Locomotion	72.4 ± 2.1	66.1 ± 3.2	71.9 ± 4.4	71.4 ± 2.3	69.5 ± 1.8	69.7 ± 1.8	$\mathbf{73.6 \pm 3.1}$
TennisShot	79.3 ± 3.2	78.6 ± 2.2	72.7 ± 3.7	62.5 ± 2.8	67.1 ± 3.8	66.2 ± 5.8	$\mathbf{82.3 \pm 4.0}$
Transportation	56.4 ± 6.9	55.9 ± 3.7	55.4 ± 5.6	$\mathbf{63.5 \pm 7.0}$	56.7 ± 5.3	55.7 ± 3.3	62.8 ± 4.2
Workouts	73.6 ± 5.1	67.3 ± 5.7	69.5 ± 4.8	69.4 ± 4.2	66.7 ± 2.8	67.2 ± 4.6	$\mathbf{71.0 \pm 4.1}$
Writing	55.6 ± 9.5	45.5 ± 6.2	35.8 ± 3.8	13.3 ± 0.9	26.4 ± 3.7	31.8 ± 3.7	$\mathbf{49.0 \pm 4.6}$
Regression – RMSE (↓)							
JumpPower	3.88 ± 0.28	10.23 ± 1.71	8.08 ± 1.32	9.77 ± 0.58	9.83 ± 1.08	8.53 ± 2.10	$\mathbf{6.46 \pm 0.79}$
NumSteps	0.65 ± 0.16	0.71 ± 0.08	$\mathbf{0.64 \pm 0.13}$	0.74 ± 0.13	$\mathbf{0.64 \pm 0.16}$	0.70 ± 0.15	0.69 ± 0.13
ThrowSpeed	2.73 ± 0.21	3.73 ± 0.43	3.75 ± 0.48	4.33 ± 0.66	$\mathbf{3.03 \pm 0.31}$	3.07 ± 0.38	$\mathbf{3.03 \pm 0.42}$
WalkDistance	0.25 ± 0.07	0.33 ± 0.04	0.29 ± 0.06	0.30 ± 0.09	0.29 ± 0.06	$\mathbf{0.27 \pm 0.08}$	0.29 ± 0.05

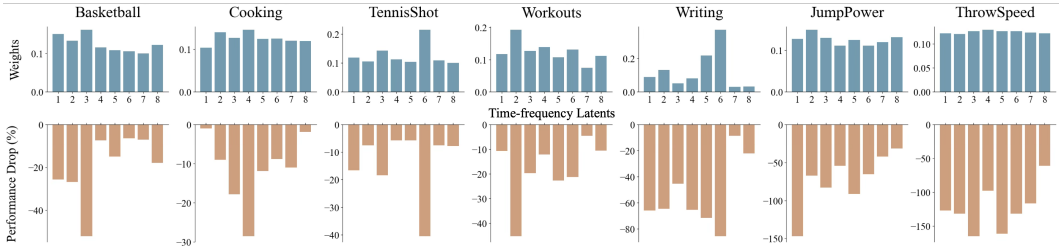

Figure 3: Softmax-normalized task-head weights associated with each latent in the first linear layer (top) and performance changes when each latent is set to zero before the task-specific head (bottom). Different tasks show different weight and masking patterns across the learned latents.

sitions in multiple latents, *Tiles* allows each task head to combine the parts of the representation relevant to its task, contributing to more consistent performance across the evaluation.

5.2 Task-Dependent Latent Use

To examine how downstream tasks use different parts of the learned representation, we first analyze the task-head weights associated with each latent in the first linear layer. We compute the mean absolute weight associated with each latent and apply softmax normalization across the resulting latent-level values. As shown in the top row of Figure 3, the softmax-normalized weight distributions differ across tasks. Some tasks assign larger weights to a small subset of latents, including Writing and TennisShot, while others distribute their weights more evenly. These weight patterns provide an initial indication that the task heads combine the latents differently.

We further examine this behavior through individual-latent masking. During inference, we set one time-frequency latent at a time to zero before applying the task-specific head and measure the resulting change in performance. As shown in the bottom row of Figure 3, the effect of masking each latent differs across tasks. The latent whose masking causes the largest performance degradation also varies by task. Together, the weight and masking results indicate that downstream tasks are sensitive to different combinations of the learned latents. Complementary t-SNE visualizations of the most and least weighted latents are provided in Appendix A.5.4.

5.3 Component Ablations

Table 3 reports the relative performance change when individual components of *Tiles* are removed or replaced. For macro-F1, relative change is calculated as $100(m_{\text{ablated}} - m_{\text{full}})/m_{\text{full}}$. For RMSE, it is calculated as $100(m_{\text{full}} - m_{\text{ablated}})/m_{\text{full}}$. Negative values therefore indicate worse performance than the complete model. The change is calculated for each task and then summarized across the classification and regression tasks.

Removing frequency decomposition and encoding the raw signal directly reduces classification performance by 5.9% on average, while the average change across regression tasks is smaller. Without explicit frequency decomposition, the aggregation module receives a single sequence of features organized only along the temporal dimension. The resulting latents can therefore partition information across temporal regions, but not across separately represented frequency ranges.

Table 3: Relative performance change after removing or replacing components of *Tiles*, reported as mean across tasks. Negative values indicate worse performance than the complete model.

	Classification tasks	**Regression tasks**
No frequency decomposition	$-5.9 \pm 9.1\%$	$-0.5 \pm 6.2\%$
No auxiliary reconstruction losses	$-12.9 \pm 9.4\%$	$-33.7 \pm 24.4\%$
No competitive aggregation	$-8.2 \pm 8.8\%$	$-6.1 \pm 18.9\%$
Stochastic query initialization	$-12.8 \pm 14.9\%$	$-29.7 \pm 20.9\%$
No self-attention head	$-6.7 \pm 6.3\%$	$-8.3 \pm 17.3\%$

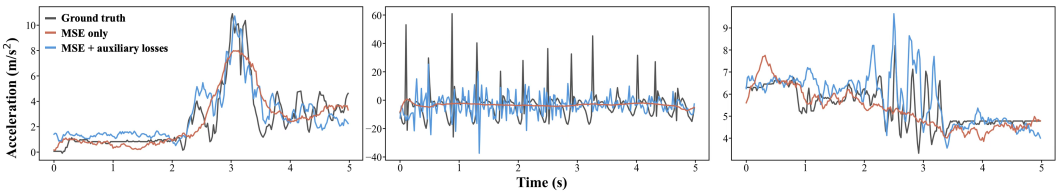

Figure 4: Example signal reconstructions obtained using MSE alone and the complete band-specific reconstruction objective with auxiliary SSIM and MS-STFT losses. The auxiliary losses encourage the reconstruction to preserve local variations that are smoothed when only pointwise MSE is used.

Removing the auxiliary SSIM and MS-STFT losses and using MSE alone produces a larger decrease, particularly on the regression tasks. Classification performance decreases by 12.9% on average, while regression performance decreases by 33.7%, although the effect varies across tasks. These results indicate that explicitly constraining local structure and multi-scale spectral patterns contributes to downstream performance. Figure 4 compares reconstructions produced using MSE alone with those produced using the complete objective containing the auxiliary $\mathcal{L}_{\text{SSIM}}$ and $\mathcal{L}_{\text{MS-STFT}}$ losses. With MSE alone, the model captures the overall signal trend but smooths higher-frequency variations and misses several sharp peaks, particularly in the second example. Adding the auxiliary losses preserves more of the local fluctuations and sharp transition patterns, although the reconstructed peak magnitudes remain lower than those of the original signal.

Replacing iterative competitive aggregation with a standard autoencoder while retaining frequency decomposition and the auxiliary losses reduces performance on both classification and regression tasks. This result supports organizing the encoded features into multiple time-frequency latents rather than compressing the signal into a single global representation.

Replacing the learned queries with vectors sampled from a learned normal distribution produces average decreases of 12.8% on classification and 29.7% on regression. Stochastic initialization can change the assignment of information across latent positions between forward passes, making it more difficult for task-specific heads to use particular latents consistently.

Finally, removing the self-attention layer from the task-specific head reduces classification and regression performance by 6.7% and 8.3%, respectively. Although this layer accounts for less than 10% of the head's parameters, it allows the head to model relationships among the time-frequency latents before prediction. Increasing the sizes of the baseline MLP heads by an equivalent number of parameters did not produce comparable improvements in our experiments, suggesting that the gain comes from combining information across latents rather than from parameter count alone.

Additional ablations on the numbers of frequency bands and latents, as well as t-SNE visualizations under selected component ablations, are provided in Appendix A.5.2, A.5.3, and A.5.1, respectively.

6 Conclusion

We introduced *Tiles*, a self-supervised approach that learns multiple time-frequency latents from accelerometer signals through frequency-decomposed encoding and iterative competitive aggregation with learned queries. Across 11 classification and regression tasks extending beyond standard activity classification, *Tiles* achieves the best or tied-best SSL result on 8 tasks and comparable results on the rest. Other SSL methods sometimes perform best on one task but degrade substantially on another, suggesting that their pre-training objectives favor particular signal characteristics. *Tiles* performs consistently across tasks with different signal requirements. Task-head weights and individual-latent masking further show that tasks rely on different combinations of the learned latents. This indicates that downstream tasks draw on different signal characteristics, making it important for a transferable representation to preserve multiple types of signal information. We also introduced two auxiliary reconstruction losses that preserve local and high-frequency details otherwise smoothed by MSE-only reconstruction. Removing these losses substantially reduces downstream performance, showing that preserving fine temporal and spectral structure during pre-training is important for retaining information useful to downstream tasks. Overall, these findings support multi-latent representations for accelerometer tasks.

Reproducibility Statement

Source code for *Tiles* and the downstream evaluation pipeline will be released upon acceptance. The code includes scripts for downloading and preprocessing the publicly available downstream datasets. Model pseudocode and hyperparameters are provided in Appendix A.3.

AI Use Statement

Generative AI tools were used to draft and polish parts of the manuscript. All AI-assisted text was reviewed and revised by the authors, who take responsibility for the final content.

References

Riku Arakawa, Hiromu Yakura, Vimal Mollyn, Suzanne Nie, Emma Russell, Dustin P DeMeo, Haarika A Reddy, Alexander K Maytin, Bryan T Carroll, Jill Fain Lehman, et al. Prism-tracker: A framework for multimodal procedure tracking using wearable sensors and state transition information with user-driven handling of errors and uncertainty. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 6(4):1–27, 2023.

Marc Bachlin, Meir Plotnik, Daniel Roggen, Inbal Maidan, Jeffrey M Hausdorff, Nir Giladi, and Gerhard Troster. Wearable assistant for parkinson's disease patients with the freezing of gait symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):436–446, 2009.

Henrik Blunck, Sourav Bhattacharya, Thor Prentow, Mikkel Kjrgaard, and Anind Dey. Heterogeneity activity recognition. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5689X.

Marius Bock, Hilde Kuehne, Kristof Van Laerhoven, and Michael Moeller. Wear: An outdoor sports dataset for wearable and egocentric activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 8(4):1–21, 2024.

Shing Chan, Yuan Hang, Catherine Tong, Aidan Acquah, Abram Schonfeldt, Jonathan Gershuny, and Aiden Doherty. CAPTURE-24: A large dataset of wrist-worn activity tracker data collected in the wild for human activity recognition. *Scientific Data*, 11(1):1135, 2024.

Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *International conference on machine learning*, pp. 894–903. PMLR, 2017.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=ivCd8z8zR2. Featured Certification, Reproducibility Certification.

Vijay Ekambaram, Subodh Kumar, Arindam Jati, Sumanta Mukherjee, Tomoya Sakai, Pankaj Dayama, Wesley Gifford, and Jayant Kalagnanam. TSPulse: Tiny pre-trained models with disentangled representations for rapid time-series analysis. In *International Conference on Learning Representations*, volume 2026, pp. 62090–62133, 2026.

Eray Erturk, Fahad Kamran, Salar Abbaspourazad, Sean Jewell, Harsh Sharma, Yujie Li, Sinead Williamson, Nicholas J Foti, and Joseph Futoma. Beyond sensor data: Foundation models of behavioral data from wearables improve health predictions. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=DtVVltU1ak.

Jonatan Fridolfsson, Mats Börjesson, Christoph Buck, Örjan Ekblom, Elin Ekblom-Bak, Monica Hunsberger, Lauren Lissner, and Daniel Arvidsson. Effects of frequency filtering on intensity and noise in accelerometer-based physical activity measurements. *Sensors*, 19(9):2186, 2019.

Celal Gençoğlu and Hikmet Gümüş. Standing handball throwing velocity estimation with a single wrist-mounted inertial sensor. *Annals of Applied Sport Science*, 8(3), 2020. doi: 10.29252/aassjournal.893. URL http://aassjournal.com/article-1-893-en.html.

Hristijan Gjoreski, Mathias Ciliberto, Lin Wang, Francisco Javier Ordonez Morales, Sami Mekki, Stefan Valentin, and Daniel Roggen. The university of sussex-huawei locomotion and transportation dataset for multimodal analytics with mobile devices. *IEEE Access*, 6:42592–42604, 2018.

Harish Haresamudram, David V Anderson, and Thomas Plötz. On the role of features in human activity recognition. In *Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pp. 78–88, 2019.

Harish Haresamudram, Apoorva Beedu, Varun Agrawal, Patrick L Grady, Irfan Essa, Judy Hoffman, and Thomas Plötz. Masked reconstruction based self-supervision for human activity recognition. In *Proceedings of the 2020 ACM International Symposium on Wearable Computers*, pp. 45–49, 2020.

Alexander Hoelzemann, Julia L. Romero, Marius Bock, Kristof Van Laerhoven, and Qin Lv. Hang-time HAR: A benchmark dataset for basketball activity recognition using wrist-worn inertial sensors. *Sensors*, 23(13), 2023. URL https://doi.org/10.3390/s23135879.

Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving object-centric learning with query optimization. In *The Eleventh International Conference on Learning Representations*, 2023.

Heli Koskimäki, Pekka Siirtola, and Juha Röning. Myogym: introducing an open gym data set for activity recognition collected using myo armband. In *Proceedings of the 2017 ACM international joint conference on pervasive and ubiquitous computing and proceedings of the 2017 ACM international symposium on wearable computers*, pp. 537–546, 2017.

Simon Lee, Cyrus Tanade, Hao Zhou, Juhyeon Lee, Megha Thukral, Md Sazzad Hissain Khan, Keum San Chun, Baiying Lu, Migyeong Gwak, Mehrab Bin Morshed, et al. HiMAE: Hierarchical masked autoencoders discover resolution-specific structure in wearable time series. In *International Conference on Learning Representations*, volume 2026, pp. 64859–64895, 2026.

Asaf Liberman, Oron Levy, Soroush Shahi, Cori Tymoszek Park, Mike Ralph, Richard Kang, Abdelkareem Bedri, and Gierad Laput. Moonwalk: Advancing gait-based user recognition on wearable devices with metric learning. *arXiv preprint arXiv:2402.08451*, 2024.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Aleksej Logacjov. Self-supervised learning for accelerometer-based human activity recognition: A survey. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 8(4):1–42, 2024.

Aleksej Logacjov, Sverre Herland, Astrid Ustad, and Kerstin Bach. SelfPAB: large-scale pretraining on accelerometer data for human activity recognition: A. logacjov et al. *Applied Intelligence*, 54(6):4545–4563, 2024.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.05440.

Vimal Mollyn, Riku Arakawa, Mayank Goel, Chris Harrison, and Karan Ahuja. Imuposer: Full-body pose estimation using imus in phones, watches, and earbuds. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2023.

Girish Narayanswamy, Xin Liu, Kumar Ayush, Yuzhe Yang, Xuhai Xu, Shun Liao, Jake Garrison, Shyam A. Tailor, Jacob Sunshine, Yun Liu, Tim Althoff, Shrikanth Narayanan, Pushmeet Kohli, Jiening Zhan, Mark Malhotra, Shwetak Patel, Samy Abdel-Ghaffar, and Daniel McDuff. Scaling wearable foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yb4QE6b22f.

Girish Narayanswamy, Maxwell A Xu, A Ali Heydari, Samy Abdel-Ghaffar, Marius Guerard, Kara Vaillancourt, Zhihan Zhang, Jake Garrison, Levi Albuquerque, Dimitris Spathis, et al. Towards a general intelligence and interface for wearable health data. *arXiv preprint arXiv:2605.22759*, 2026.

Junyong Park, Saelyne Yang, and Sungho Jo. Silent impact: Tracking tennis shots from the passive arm. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–15, 2024.

Arvind Pillai, Dimitris Spathis, Fahim Kawsar, and Mohammad Malekzadeh. Papagei: Open foundation models for optical physiological signals. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kYwTmlq6Vn.

Daniel Roggen. HCI Tabletop Gestures dataset. http://har-dataset.org/doku.php?id=wiki:dataset, 2022. Release: 2022-02-13, Accessed: 2025-08-15.

Aaqib Saeed, Tanir Ozcelebi, and Johan Lukkien. Multi-task self-supervised learning for human activity detection. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):1–30, 2019.

Geise Santos, Marcelo Wanderley, Tiago Tavares, and Anderson Rocha. A multi-sensor human gait dataset captured through an optical system and inertial measurement units. *Scientific Data*, 9(1): 545, 2022.

Philipp M Scholl, Matthias Wille, and Kristof Van Laerhoven. Wearables in the wet lab: a laboratory system for capturing and guiding experiments. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 589–599, 2015.

Timo Sztyler and Heiner Stuckenschmidt. On-body localization of wearable devices: An investigation of position-aware activity recognition. In *2016 IEEE international conference on pervasive computing and communications (PerCom)*, pp. 1–9. IEEE, 2016.

Setareh Rahimi Taghanaki, Michael Rainbow, and Ali Etemad. Self-supervised human activity recognition with localized time-frequency contrastive representation learning. *IEEE Transactions on Human-Machine Systems*, 53(6):1027–1037, 2023.

Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, and Cecilia Mascolo. Exploring contrastive learning in human activity recognition for healthcare. *arXiv preprint arXiv:2011.11542*, 2020.

Giacomo Villa, Alessandro Bonfiglio, Manuela Galli, and Veronica Cimolin. Vertical jump height estimation using low-sampling IMU in countermovement jumps: A feasible alternative to motion capture and force platforms. *Sensors*, 24(24):7877, 2024.

Mark GE White, Neil E Bezodis, Jonathon Neville, Huw Summers, and Paul Rees. Determining jumping performance from a single body-worn accelerometer using machine learning. *Plos one*, 17(2):e0263846, 2022.

Huatao Xu, Pengfei Zhou, Rui Tan, Mo Li, and Guobin Shen. LIMU-BERT: Unleashing the potential of unlabeled data for imu sensing applications. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pp. 220–233, 2021.

Maxwell Xu, Alexander Moreno, Hui Wei, Benjamin Marlin, and James Matthew Rehg. REBAR: Retrieval-based reconstruction for time-series contrastive learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3zQo5oUvia.

Maxwell A Xu, Jaya Narain, Gregory Darnell, Haraldur T Hallgrimsson, Hyewon Jeong, Darren Forde, Richard Andres Fineman, Karthik Jayaraman Raghuram, James Matthew Rehg, and Shirley You Ren. Relcon: Relative contrastive learning for a motion foundation model for wearable data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=k2uUeLCrQq.

Che-Chang Yang and Yeh-Liang Hsu. A review of accelerometry-based wearable motion detectors for physical activity monitoring. *Sensors*, 10(8):7772–7788, 2010.

Hang Yuan, Shing Chan, Andrew P Creagh, Catherine Tong, Aidan Acquah, David A Clifton, and Aiden Doherty. Self-supervised learning for human activity recognition using 700,000 person-days of wearable data. *NPJ digital medicine*, 7(1):91, 2024.

Dian Zhang, Zexiong Liao, Wen Xie, Xiaofeng Wu, Haoran Xie, Jiang Xiao, and Landu Jiang. Fine-grained and real-time gesture recognition by using IMU sensors. *IEEE Transactions on Mobile Computing*, 22(4):2177–2189, 2021.

Yuwei Zhang, Kumar Ayush, Siyuan Qiao, A. Ali Heydari, Girish Narayanswamy, Maxwell A Xu, Ahmed Metwally, Jinhua Xu, Jake Garrison, Xuhai Xu, Tim Althoff, Yun Liu, Pushmeet Kohli, Jiening Zhan, Mark Malhotra, Shwetak Patel, Cecilia Mascolo, Xin Liu, Daniel McDuff, and Yuzhe Yang. SensorLM: Learning the language of wearable sensors. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=TrHeq0yFhv.

Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57, 2016.

A Appendix

A.1 Limitations and Future Work

Our study focuses on accelerometer signals and uses CAPTURE-24 as its only pre-training dataset. Several downstream datasets contain relatively few participants, and our latent-use analyses do not establish fixed semantic meanings for individual latents. Future work could examine transfer to other wearable sensing modalities and pre-training on more varied datasets. Alternative task heads could provide more flexible ways to combine the learned latents, while task-conditioned queries could adapt latent construction to individual downstream tasks.

A.2 Downstream Datasets and Task Construction

All downstream datasets are resampled to 50 Hz and divided into non-overlapping 5-second windows, producing inputs of 250 samples, unless indicated otherwise. Acceleration measurements are converted to m/s^2. We retain the preprocessing provided with each dataset and do not apply additional normalization. TennisShot and Writing retain their original sampling rates while using the same input length of 250 samples, resulting in shorter temporal durations.

To account for differences in dataset size and characteristics, we evaluate six downstream configurations consisting of three learning rates, 1×10^{-3}, 5×10^{-4}, and 1×10^{-4}, and task heads with either two or three MLP layers. For each method and fold, the configuration is selected using validation performance, and the corresponding test performance is reported.

Basketball is constructed from the Hang-Time HAR dataset (Hoelzemann et al., 2023). We use five basketball-related actions consisting of dribbling, shot, layup, pass, and rebound.

Cooking is constructed from the PrISM dataset (Arakawa et al., 2023). Because the dataset provides only the starting timestamp of each action, we extract a single 5-second window beginning at each annotated timestamp. We use all 14 cooking action classes.

Locomotion is constructed from the RealWorld dataset (Sztyler & Stuckenschmidt, 2016). We use all eight locomotion-related activities.

TennisShot is constructed from the Silent Impact dataset (Park et al., 2024). Each tennis shot is provided as a 1.5-second segment sampled at 120 Hz, giving 180 samples. We retain the original sampling rate and append 70 copies of the final value to produce the fixed input length of 250 samples. We use accelerometer data from the non-dominant arm and include all six shot classes.

Transportation is constructed from the Sussex-Huawei Locomotion and Transportation dataset (Gjoreski et al., 2018). We use accelerometer data collected from a phone placed in each participant's front trouser pocket. The six classes are still, bike, car, bus, train, and subway. Because the dataset contains only three participants, we partition continuous same-label segments across five folds while keeping each segment intact.

Workouts is constructed from the WEAR dataset (Bock et al., 2024). We use all 18 outdoor workout activities.

Writing is constructed from the HCI Tabletop Gestures dataset (Roggen, 2022). Characters are generally written in less than 2 seconds. We retain the original sampling rate of 100 Hz and construct 250-sample inputs covering at most 2.5 seconds, repeating the final value when padding is required. We use only the *Table* interaction mode because the *Mouse* and *Slate* modes contain substantially less wrist movement. All 39 character classes from the *Table* mode are included.

JumpPower is constructed from the dataset by White et al. (2022). We use all recorded jumps from 73 participants and use peak power as the regression target.

NumSteps is constructed from the dataset by Santos et al. (2022). We use the vertical motion-capture trajectories of both heel markers to count the number of steps taken within each window.

ThrowSpeed is constructed from the dataset by Gençoğlu & Gümüş (2020). The regression target is the measured speed of each throw. Because the released dataset contains only four participants and does not provide participant identifiers for individual trials, we divide the trials into five non-overlapping folds.

WalkDistance is constructed from the dataset introduced by Santos et al. (2022). We calculate the distance walked from the two-dimensional motion-capture trajectory of the clavicle marker.

A.3 *Tiles* Implementation Details

A.3.1 Pseudocode

Algorithms 1 and 2 provide pseudocode for *Tiles*. Algorithm 1 describes the representation encoder used during both pre-training and downstream evaluation. It decomposes an accelerometer window into frequency bands, encodes the bands into temporal features, and aggregates these features into K time-frequency latents. Algorithm 2 describes the reconstruction decoder used only during pre-training, which jointly reconstructs each frequency band from the latents.

Algorithm 1: Time-frequency latent encoder. An accelerometer window is decomposed into three frequency bands and processed by band-specific encoders. Iterative competitive attention with learned queries then aggregates the encoded features into K time-frequency latents.

Input : $X \in \mathbb{R}^{C \times T}$
Output : $\mathcal{Z} \in \mathbb{R}^{K \times D_{\text{latent}}}$
Params : BandFilters; $\{\text{Encoder}_b\}_{b \in \mathcal{B}}$; learned queries $\mathcal{Q} \in \mathbb{R}^{K \times D_{\text{latent}}}$; W_q, W_k, W_v; number of iterations U; GRU; MLP; PositionalEmbedding; LinearProjection; LayerNorm

$\mathcal{B} = \{\text{low}, \text{mid}, \text{high}\}$
$\{X_b\}_{b \in \mathcal{B}} = \text{BandFilters}(X)$
foreach $b \in \mathcal{B}$ **do**
  $E_b = \text{Encoder}_b(X_b)$ // $E_b \in \mathbb{R}^{T \times D_{\text{enc}}}$
$E = \text{concat}(E_{\text{low}}, E_{\text{mid}}, E_{\text{high}})$ // $E \in \mathbb{R}^{N \times D_{\text{enc}}}$, $N = 3T$
$E = \text{PositionalEmbedding}(E)$
$E = \text{LinearProjection}(E)$ // $E \in \mathbb{R}^{N \times D_{\text{latent}}}$
$E = \text{LayerNorm}(E)$
$\mathbf{K}_E = W_k(E), \qquad \mathbf{V}_E = W_v(E)$
$\mathcal{Z} = \mathcal{Q}$
for $u = 1 \dots U$ **do**
  $\mathcal{Z}_{\text{prev}} = \mathcal{Z}$
  $\mathbf{Q}_u = W_q(\text{LayerNorm}(\mathcal{Z}))$
  $\text{scores} = \mathbf{Q}_u \mathbf{K}_E^\top / \sqrt{D_{\text{latent}}}$ // $\mathbb{R}^{K \times N}$
  $\text{attn} = \text{softmax}(\text{scores},\ \text{axis=latents})$ // $\mathbb{R}^{K \times N}$
  $\text{updates} = \text{attn}\mathbf{V}_E$ // $\mathbb{R}^{K \times D_{\text{latent}}}$
  $\mathcal{Z} = \text{GRU}(\text{state} = \mathcal{Z}_{\text{prev}}, \text{inputs} = \text{updates})$
  $\mathcal{Z} = \mathcal{Z} + \text{MLP}(\text{LayerNorm}(\mathcal{Z}))$
return $\mathcal{Z}$

Algorithm 2: Pre-training reconstruction decoder. Each time-frequency latent is decoded into a signal reconstruction and temporal mask for each frequency band. The masks are normalized across latents and used to combine the per-latent reconstructions. This decoder is removed after pre-training.

Input : $\mathcal{Z} \in \mathbb{R}^{K \times D_{\text{latent}}}$
Output : $\{\hat{X}_b \in \mathbb{R}^{C \times T}\}_{b \in \mathcal{B}}$
Params : $\{\text{Decoder}_b\}_{b \in \mathcal{B}}$

$\mathcal{B} = \{\text{low}, \text{mid}, \text{high}\}$
foreach $b \in \mathcal{B}$ **do**
  for $k = 1 \dots K$ **do**
    $R_b[k], L_b[k] = \text{Decoder}_b(\mathcal{Z}[k])$ // $R_b[k] \in \mathbb{R}^{C \times T}$, $L_b[k] \in \mathbb{R}^{1 \times T}$
  $M_b = \text{softmax}(L_b,\ \text{axis=latents})$
  $\hat{X}_b = \sum_{k=1}^{K} M_b[k] \odot R_b[k]$
return $\{\hat{X}_b\}_{b \in \mathcal{B}}$

Table 4: Reconstruction-loss configuration for each frequency band. Window size refers to the sliding window used to calculate SSIM.

Band	Losses	α (MSE)	β (SSIM)	γ (MS-STFT)	Window Size
Low	MSE only	1	0	0	–
Mid	MSE, SSIM, MS-STFT	1	100	0.1	25
High	SSIM, MS-STFT	0	50	0.1	10

A.3.2 Training Hyper-parameters

We pretrain using the Adam optimizer with a learning rate of 1×10^{-4}. The total representation dimension is 256, and the number of iterative aggregation updates is fixed at three. At each batch, data from four participants are loaded, where for each participant, 128 windows are sampled with probabilities proportional to the standard deviation of their signals, resulting in a batch size of 512.

A.3.3 Positional Embeddings

Following Slot Attention (Locatello et al., 2020), we augment the encoded features with positional embeddings before passing them to the iterative competitive aggregation module. Cross-attention does not otherwise encode the frequency-band or temporal position of each feature. We construct a $B \times T \times 4$ coordinate tensor, where B is the number of frequency bands and T is the number of frames per band. Its four channels contain normalized frequency and temporal coordinates $(f, t, 1-f, 1-t)$. The coordinate tensor is linearly projected to the encoded feature dimension and added to the corresponding features.

A.3.4 Auxiliary Reconstruction Losses

We combine three loss functions, MSE, SSIM and MS-STFT, each weighted by coefficients α, β, and γ, respectively. Because the three frequency bands differ substantially in their signal characteristics, we assign distinct weights to each band. The SSIM coefficients are larger because the numerical scale of the SSIM loss is smaller than those of MSE and MS-STFT during training. For MS-STFT, we use a Hann window of length n_{fft} and a hop length of $n_{\text{fft}}/4$.

The coefficients were selected by first matching the scales of the losses during the initial epochs and then selecting among a few coarse candidate values that yielded stable training and consistent reconstruction quality. For the low-frequency band, which captures slow, large-magnitude movements, we apply only MSE to preserve absolute signal levels and slow variations. For the mid-frequency band, which contains more nuanced variations, we use all three losses with $\alpha = 1$, $\beta = 100$, and $\gamma = 0.1$, together with an SSIM window size of 25 frames. For the high-frequency band, which captures short-period fluctuations, we use SSIM and MS-STFT with $\beta = 50$ and $\gamma = 0.1$, together with a smaller SSIM window size of 10 frames. Table 4 summarizes the final configuration.

A.4 SSL Baseline Implementation Details

We provide implementation details for the five SSL baselines used in the downstream evaluation: Autoencoder, Masked Autoencoder, SimCLR, Augmentation Prediction, and RelCon. Each method was adapted from prior implementations but modified to ensure a fair comparison. The data, dataloading mechanism, batch size, and training epochs were kept identical to *Tiles*. The encoder architecture was also standardized, using a ResNet with 12 blocks that preserves the temporal dimension, followed by Global Average Pooling and a linear layer to produce a 256-dimensional embedding. This results in an encoder with 4.8M parameters, matching *Tiles*. Additional network components specific to each SSL approach were added consistent with the original work. Where necessary, design choices were explored to identify the best-performing configurations.

Autoencoder follows the convolutional autoencoder from Haresamudram et al. (2019). The ResNet encoder produces a global representation, which is decoded into the reconstructed signal using a mirrored ResNet decoder. The model is trained using the Adam optimizer and MSE between the original and reconstructed signals.

Table 5: Parameter sizes of *Tiles* and the baseline models. Pretext-specific components are included only during pre-training.

Method	Encoder Params	Pretext / Decoder Params	Total Params
Tiles	4.8M	4.8M (decoder)	9.6M
Autoencoder	4.8M	4.8M (decoder)	9.6M
Masked Autoencoder	4.8M	4.8M (decoder)	9.6M
Augmentation Prediction	4.8M	2K (pretext classifier)	4.8M
SimCLR	4.8M	80K (projection head)	4.9M
RelCon	4.8M	80K (projection head) + 59K (distance network)	4.9M

Masked Autoencoder extends the autoencoder by randomly masking parts of the raw input before encoding. After experimenting with different masking ratios (20%, 50%, 80%) and segment sizes (10, 25, 50 frames per segment), we finalized on masking 50% of the input by dividing each window into 10 segments and randomly selecting 5 to mask per epoch. The model is trained with the Adam optimizer and MSE loss over the full signal.

Augmentation Prediction follows the official implementation of Yuan et al. (2024), where the encoder is trained to classify which augmentation was applied to the input. The same ResNet backbone is used, with a separate linear classifier for each augmentation, making it a binary prediction task per augmentation. Following Yuan et al. (2024), random shuffle and rotation are applied for invariance, while the tasks used for prediction are timeflip, permute, and timewarp, each applied with 50% probability. The model is trained with the Adam optimizer and cross-entropy loss.

SimCLR follows the official contrastive learning implementation of Tang et al. (2020). Two augmented views of each input window are encoded by the ResNet backbone and projected through a two-layer MLP head into a 64-dimensional embedding. The model is trained with the NT-Xent loss with a temperature of 0.1, with positive and negative pairs drawn from within the batch. After experimenting with different augmentation combinations from the available eight, we finalized on random scaling and time warping, each applied with a probability of 0.3. The model is optimized with SGD (momentum 0.9, weight decay 1×10^{-5}) and a cosine annealing learning rate scheduler.

RelCon follows the official implementation of Xu et al. (2025). The Learnable Distance Measure uses the same architecture as in the original work and is trained with MSE loss between the input and reconstructed signal. In each batch, two of the eight available augmentations are chosen at random and applied to the input. The encoder of the main backbone is replaced with our ResNet-12, and the resulting embeddings are projected into a 64-dimensional space. The model is trained using the original Relative Contrastive Loss and the Adam optimizer.

Table 5 summarizes the number of trainable parameters for *Tiles* and all baselines. Note that all methods use the same encoder size, and only the decoder or other pretext-specific components used during pre-training differ in parameter count.

A.5 Further Ablation Studies

A.5.1 Qualitative Component Ablation

To provide a qualitative view of selected component ablations, Figure 5 shows t-SNE projections of the learned representations for the Workouts task. The complete *Tiles* configuration produces more distinct class clusters, while removing frequency decomposition, the auxiliary reconstruction losses, or iterative competitive aggregation results in greater overlap between classes. These patterns are consistent with the downstream performance decreases reported in Table 3.

A.5.2 Number of Bands

Figure 6 shows the effect of the number of bands used as input for the model. The solid black line shows the average across tasks, while the dotted colored lines show individual task performances. In calculating the average, the RMSE values for the JumpPower and ThrowSpeed tasks are divided by 10 to match the scale of the other regression tasks.

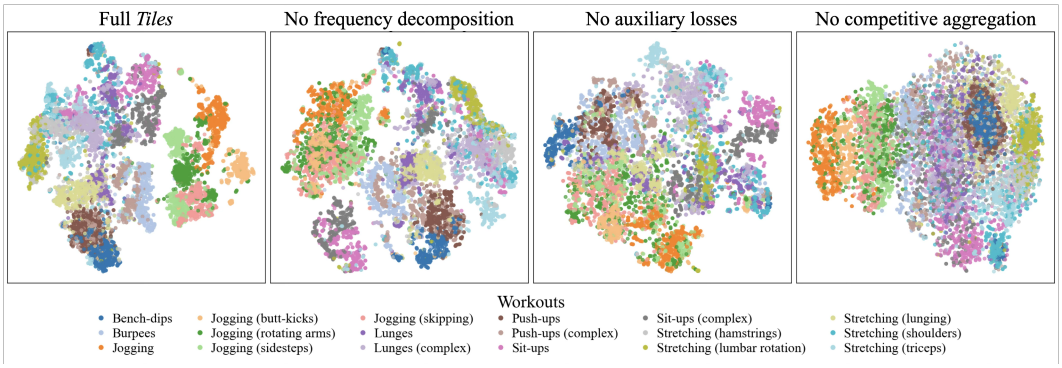

Figure 5: t-SNE projections of the learned representations for the Workouts task under the complete *Tiles* configuration and selected component ablations.

For each configuration, the input signal is processed through a bandpass filter to decompose it into a given number of bands. Table 6 shows the cutoff boundaries for each configuration. Since most of the motion in the signals lies in the lower frequencies, typically under 10 Hz, we allocated finer granularity to the lower frequency ranges. Following our model design, where each band has a separate encoder to account for differences in patterns and shapes across frequency ranges, we adjusted the number of encoder layers so that the total model size remained constant. This means that the single-band configuration uses the largest encoder with 12 ResBlocks, while the 12-band configuration uses only one ResBlock per band.

The results show that while decomposing the signal into multiple bands improves performance, more bands do not necessarily lead to better performance. For both classification and regression tasks, performance begins to degrade from 6 bands onward. This is likely because splitting into too many bands makes each band narrower and less informative, and the smaller encoders assigned to each band may not have enough capacity to capture meaningful features. Using 12 bands leads to a substantial drop in performance, since higher frequencies contain less information, and dedicating individual bands to these ranges does not contribute anything meaningful.

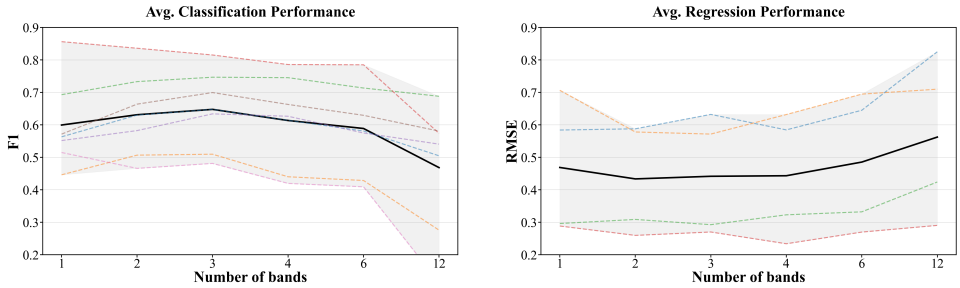

Figure 6: Average performance on classification and regression tasks when the input signal is bandpassed into different numbers of bands. The solid black line shows the overall average, and the dotted colored lines show the performance of each individual task.

A.5.3 Number of Time-Frequency Latents

Figure 7 shows the effect of the number of time-frequency latents on classification and regression performance. The dimension of each latent is adjusted so that the total representation dimension remains constant. For example, with 2 latents, each latent has a dimension of 128, while with 32 latents, each has a dimension of 8. Overall, performance is relatively stable across different numbers of latents. For classification, 8 latents give the best performance, while larger numbers lead to a drop, especially at 32. For regression, performance is mostly stable across the different configurations, but 32 latents show a clear degradation.

Table 6: Cutoff frequency boundaries for each band configuration.

# Bands	Cutoff Boundaries (Hz)
1	0-25
2	0-1, 1-25
3	0-1, 1-4, 4-25
4	0-1, 1-4, 4-10, 10-25
6	0-1, 1-2, 2-4, 4-6, 6-10, 10-25
12	0-1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-8, 8-10, 10-12, 12-14, 14-16, 16-25

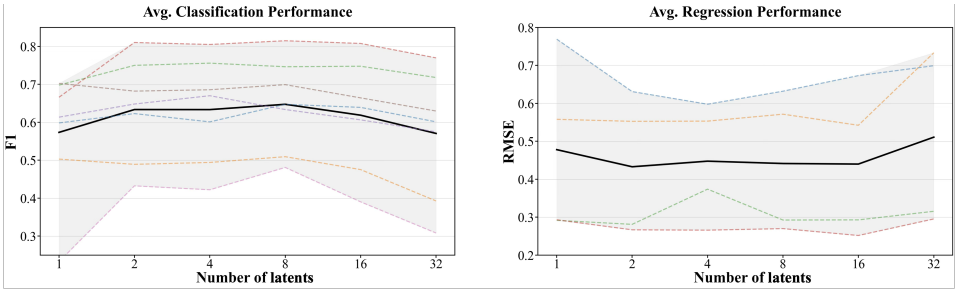

Figure 7: Average performance on classification and regression tasks with different numbers of time-frequency latents. The solid black line shows the overall average, and the dotted colored lines show the performance of each individual task.

A.5.4 t-SNE Analysis of Latent Use

Figure 8 shows t-SNE projections of the latents receiving the two highest and two lowest task-head weights for three tasks. The latents that exhibit the strongest separation differ by task: a

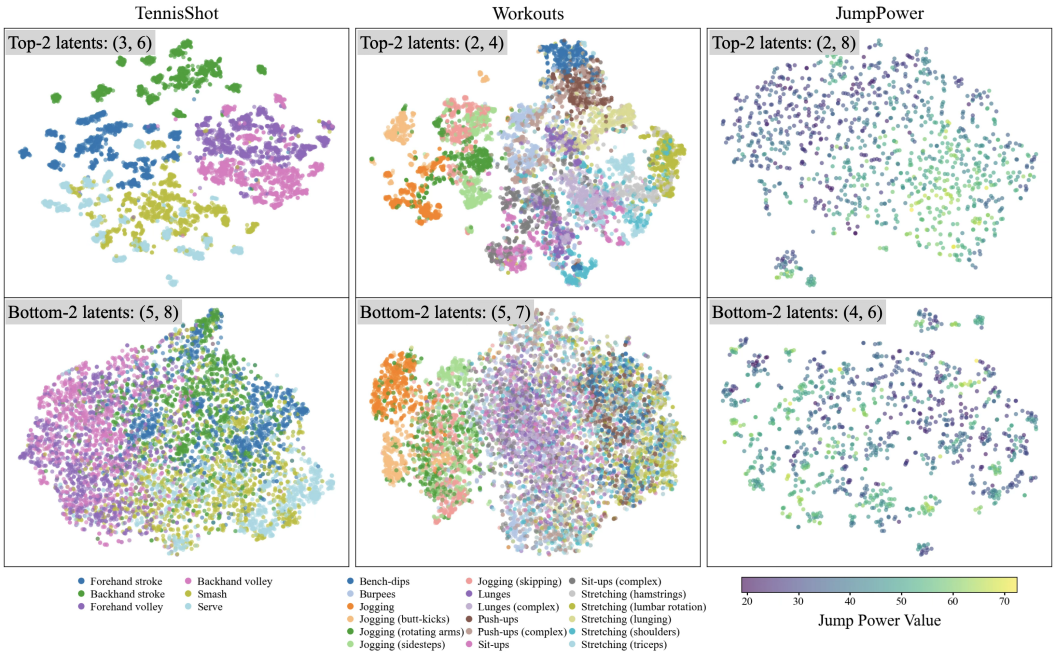

Figure 8: t-SNE visualizations of the time-frequency latents with the two highest task-head weights (top) and two lowest weights (bottom) for three tasks. The latents that show stronger label-related structure vary by task, indicating that different tasks rely on different parts of the representation.

latent that provides clear structure for one task may be among the least informative for another. The latents receiving higher weights generally produce more distinct clusters, while those receiving lower weights appear more mixed.