

Article

Towards Learning Discrete Representations via Self-Supervision for Wearables-Based Human Activity Recognition

Harish Haresamudram ^{1,*}, Irfan Essa ² and Thomas Plötz ² ¹ School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA² School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA; irfan@gatech.edu (I.E.); thomas.ploetz@gatech.edu (T.P.)

* Correspondence: hharesamudram3@gatech.edu

Abstract: Human activity recognition (HAR) in wearable and ubiquitous computing typically involves translating sensor readings into feature representations, either derived through dedicated pre-processing procedures or integrated into end-to-end learning approaches. Independent of their origin, for the vast majority of contemporary HAR methods and applications, those feature representations are typically continuous in nature. That has not always been the case. In the early days of HAR, discretization approaches had been explored—primarily motivated by the desire to minimize computational requirements on HAR, but also with a view on applications beyond mere activity classification, such as, for example, activity discovery, fingerprinting, or large-scale search. Those traditional discretization approaches, however, suffer from substantial loss in precision and resolution in the resulting data representations with detrimental effects on downstream analysis tasks. Times have changed, and in this paper, we propose a return to discretized representations. We adopt and apply recent advancements in vector quantization (VQ) to wearables applications, which enables us to directly learn a mapping between short spans of sensor data and a codebook of vectors, where the index comprises the discrete representation, resulting in recognition performance that is at least on par with their contemporary, continuous counterparts—often surpassing them. Therefore, this work presents a proof of concept for demonstrating how effective discrete representations can be derived, enabling applications beyond mere activity classification but also opening up the field to advanced tools for the analysis of symbolic sequences, as they are known, for example, from domains such as natural language processing. Based on an extensive experimental evaluation of a suite of wearable-based benchmark HAR tasks, we demonstrate the potential of our learned discretization scheme and discuss how discretized sensor data analysis can lead to substantial changes in HAR.

Keywords: human activity recognition; wearables; self-supervised learning; discrete representations

Citation: Haresamudram, H.; Essa, I.; Plötz, T. Towards Learning Discrete Representations via Self-Supervision for Wearables-Based Human Activity Recognition. *Sensors* **2024**, *24*, 1238. <https://doi.org/10.3390/s24041238>

Academic Editors: Athina Tsanousa, Georgios Meditskos, Stefanos Vrochidis, Periklis Chatzimisios and Ioannis Yiannis Kompatsiaris

Received: 20 December 2023

Revised: 25 January 2024

Accepted: 6 February 2024

Published: 15 February 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The widespread availability of commodity wearables such as smartphones and smartwatches, has resulted in increased interest in their utilization for applications such as sports and fitness tracking [1–5]. These devices benefit from onboard sensors, including Inertial Measurement Units (IMUs), which can track and measure human movements that are subsequently analyzed for understanding activities. The ubiquitous nature of the devices, coupled with their form factor, enables the collection of large-scale movement data without substantial impact on user experience, albeit without annotations.

Human activity recognition (HAR) is one such application of wearable sensing, wherein features are extracted for segmented windows of sensor data, for classification into specific activities (or the null class). The de facto approach for obtaining features is to compute them via statistical metrics [6,7] and the empirical cumulative distribution function [8], or to learn them directly from the data itself (e.g., via end-to-end training [9–13] or unsupervised learning [14–16]). Either approach results in continuous-valued (or dense) features summarizing the movement present in the windows.

Alternatively, activity recognition has occasionally also been performed on discrete sensor data representations (e.g., [17]). In those cases, short windows of sensor data are converted into *discrete* symbols, where each symbol typically covers ranges of sensor values and a span of time. The motivation for such discretization efforts was to convert complex movements into a smaller, finite alphabet of discrete (symbolic) representations, thereby simplifying tasks such as spotting gestures [17], and recognizing activities [18,19] via the use of efficient algorithms from string matching and bioinformatics, or even simple nearest neighbors in conjunction with Dynamic Time Warping (DTW). Some approaches for deriving the small collection of symbols include symbolic aggregate approximation (SAX) [20] and the Sliding Window and BottomUp (SWAB) algorithm [21,22]. SAX in particular, is especially effective at discretizing even a long-duration time series efficiently [17,20].

Existing discretization methods are rather limited with regard to their expressive power—resulting in substantial loss of resolution, as movements and activities can only be expressed by a small alphabet of symbols, which often negatively impacts downstream recognition performance. It is also observed especially acutely in tasks where minute differences in movements are important for discriminating between activities, e.g., in fine-grained gesture recognition. Moreover, the discretization methods are more difficult to apply to multi-channel sensor data, requiring specialized handling and containing exploding alphabet sizes [23]. The low recognition accuracy coupled with difficulty in handling multi-sensor setups especially has resulted in discretization methods falling behind their continuous representation counterparts and, thus, have been somewhat abandoned.

Yet, discrete representations—that are on par with contemporary continuous representations—can be crucial for tasks such as activity/routine discovery via characteristic actions [24], discovering multi-variate motifs from sensor data [25], dimensionality reduction [20], and performing interpretable time-series classification [26], since these techniques require time-series simplification as they would be produced by discretization. In recent years, the study and application of vector quantization (VQ) techniques to, for example, automatic speech recognition has resulted in the ability to learn mappings between the—continuous—audio and—discrete—codebooks of vectors, i.e., to map short durations of raw audio to discrete symbols [27,28]. In this paper, we propose to adopt and adapt such recent advancements of discrete representation learning for the HAR community, so that the symbolic representations of movement data can be derived in an unsupervised, data-driven manner, and be used for effective sensor-based human activity recognition tasks.

To this end, we apply *learned* vector quantization (VQ) [27,29] to wearables applications, which enables us to directly *learn* the mapping between short spans of sensor data and a codebook of vectors, where the index of the closest codebook vector comprises the discrete representation. In addition, we utilize self-supervised learning as a base (via the Enhanced CPC framework [30]), thereby deriving the representations without the need for annotations. Sensor data are first encoded using convolutional blocks, which can handle multiple data channels (e.g., x - y - z axes of triaxial accelerometry) in a straightforward manner. This is followed by the vector quantization module which replaces the encoding with the nearest codebook vector. They are subsequently summarized using a causal convolutional encoder, which utilizes vectors from the previous timesteps in order to predict multiple future timesteps of data in a contrastive learning setup.

We present our method as a *proof of concept* for discrete representation learning, and, as such, as a proposal for a return to discretized representations in HAR. We focus on recognizing human activities in order to demonstrate the efficacy of the representations using a standard classification backend, yet we also outline the potential the proposed return to discretized processing has for the field of sensor-based HAR. The representations are derived using wrist-based accelerometer data from Capture-24 [31–33]—a large-scale dataset that was collected in-the-wild and, as such, is representative for real-world Ubicomp applications of HAR. The performance of our discrete representation learning is contrasted against other representations, including end-to-end training (where the features are learned for accurate prediction) and self-supervised learning (where unlabeled data are first used

for representation learning), the latter recently having seen a substantial boost in the field. The evaluation is performed on six diverse benchmarks, containing a variety of activities including locomotion, daily living, and gym exercises, and comprising different numbers of participants and three sensor locations (as detailed in [16]).

The conversion of continuous-valued sensor data to discrete representations often results in comparable activity recognition accuracy, which we show in our extensive experimental evaluation. In fact, in some cases, the change in representation actually leads to improved recognition accuracy. In addition to standard activity recognition, the return to now much-improved discretization of sensor data also bears great potential for a range of additional applications such as activity discovery, activity summarization and fingerprinting, which could be used for large-scale behavior assessments both longitudinally or population-wide (or both). Effective discretization also opens up the field to the potential of entirely different categories of subsequent processing techniques, for example, NLP-based pre-training such as RoBERTa [34]—an optimized version of BERT [35]—so as to further learn effective embeddings, improving the recognition accuracy.

The contributions of our work can be summarized as follows:

- We combine learned vector quantization (VQ)—based on state-of-the-art self-supervised learning methods—with wearable-based human activity recognition in order to learn discrete representations of human movements.
- We establish the utility of learned discrete representations towards recognizing activities, where they perform comparably or better than state-of-the-art learned representations on three datasets, across sensor locations.
- We also demonstrate the applicability of highly effective NLP-based pre-training (based on RoBERTa [34]) on the discrete representations, which results in further performance improvements for all target scenarios.

2. Background

As our goal is to evaluate the effectiveness of *learning* discrete representations of sensor data, we first discuss previous methods for deriving the symbols, followed by techniques from other domains that learn discrete representations. Finally, we summarize self-supervised methods for wearables, which can be used in conjunction with vector quantization for learning discrete representations.

2.1. Discretizing Sensor Data and Deriving Primitives

Discretization of time-series data has traditionally been performed using computational methods such as symbolic aggregate approximation (SAX) [20,36]. In this technique, Piecewise Aggregate Approximation (PAA) is first utilized to obtain representations covering spans of time, which are subsequently symbolized using the alphabet size (which is a parameter to be tuned). This process is simple, yet highly effective and fast, even for long-duration time-series data. While originally proposed for a single time series, it has been extended to multi-channel data as well, by applying SAX separately to each channel and combining the tuples or first applying Principal Component Analysis (PCA) to reduce to a single channel [37]. Other variations include applying Tf-idf weighting of the SAX features, as explored in SAX-VSM [38]. Such computational discretization methods have also been applied for HAR applications, using SAX [18] and its variants [19].

Other approaches for discretization include: (i) Multivariate Bag-Of-SFA Symbols (MBOSS), [23] which learns symbolic representations but via Symbolic Fourier Approximation (SFA) [39,40]; (ii) Word extraction for time series classification (WEASEL) [41], which is an extension for SFA, and applies the ANOVA f-test to determine the most informative Fourier coefficients and subsequently applies information gain binning for determining the boundaries; (iii) the Sliding Window and BottomUp (SWAB) algorithm, [21,22] which has been used for detecting leisure activities using dense motif discovery. Suffix trees are used to extract motifs from Piecewise Linear Approximation (PLA), which first produces linear

segments from sensor data, following which the slope between consecutive segments is binned to obtain the discrete representations.

Discretization is particularly useful for applications such as activity discovery [24], involving the identification of activities from sensor streams via motif discovery. This is because discovering motifs for the time series is challenging as they can be sparsely distributed, vary in duration, and exhibit some level of time warping. Online gesture spotting has also been performed with discretized movements using string matching algorithms [17]. Motion is represented by a string of symbols, and efficient string matching methods (incl. approximate matches) are employed to recognize gestures. Similarly, primitives of motion have been derived via shapelets [42] for time series classification [43], wherein each shapelet is a local pattern highly indicative of a class. Another technique for discovering primitives includes utilizing the matrix profile (and its extensions) [44,45], which also facilitates motif discovery. As mentioned previously, the computational methods detailed above have lower performance relative to deep learning in general and do not handle multi-channel data well. Due to these issues, they have not been studied extensively in recent years.

2.2. Discrete Representations Learning in Other Domains

Learning discrete representations with deep networks were first introduced in an autoencoder setting (so-called Vector Quantized Variational AutoEncoder (VQ-VAE)) [29], where the encoder outputs discrete codes instead of continuous latents. This was achieved with the use of an online K-means loss, which allowed for a differentiable mapping of data to a codebook of vectors. It was shown to be capable of modeling long-term dependencies through the compressed discrete latent space, and performance was demonstrated for generating images, audio modeling, and sampling conditional video sequences. The generation of high-fidelity images was shown in [46], which proposed improvements to the autoencoder setup from [29].

More recently, discrete representations have shown great promise in speech recognition by enabling a differentiable mapping of spans of audio waveforms to a codebook. Unsupervised speech representations were learned using Wavenet autoencoders in [47], which also demonstrated the correspondence between phonemes and the learned symbols. VQ-Wav2vec [27] pairs a future timestep prediction task with vector quantization, and studies the effectiveness of both the K-means approach from [29] as well as the gumbel softmax operation [48,49]. Subsequently, the discrete symbols are used for RoBERTa [34] pre-training, and the resulting embeddings are utilized by an acoustic model for improved speech recognition. Using these discrete representation models now represents the state-of-the-art, with the introduction of extensions to VQ-Wav2vec, including Wav2vec2.0 [28], unsupervised speech recognition [50], and VQ-APC [51]. Other works using vector quantization include w2v-BERT [52], which sets up a masked language modeling task, and HuBERT, which also carries out masked prediction of hidden units [53]. However, these methods typically require large amounts of unlabeled data for pre-training (e.g., 960 h of audio for a base model and 60k hours for a large model). Further, a language model is utilized with beam search to decode the outputs of the acoustic model. Interestingly, the discrete representations enable the unsupervised discovery of acoustic units where phonemes are automatically mapped to a small set of discrete representations, enabling phoneme discovery and segmentation [54–57]. This resulting property of automatic discovery of ground truth phonemes is of particular interest, as we hypothesize that it allows us to derive the atomic units of human movements from wearable sensor data by learning a mapping of discrete representations to spans of sensor data. We hypothesize that these movement units enable a more accurate classification of activities, even with the loss of resolution due to discretization.

2.3. Self-Supervised Representation Learning for Human Activity Recognition

As mentioned previously, human activity recognition (HAR) involves automatically recognizing activities from windows of sensor data. In recent years, the use of supervised deep learning [10,11] has resulted in great improvements in performance, relative to more traditional methods involving heuristics [14]. These methods can sometimes require regularization in order to prevent overfitting and improve generalization [58–60]. It has also been explored for wearable-based HAR, particularly from continual learning [61] and domain generalization [62,63] perspectives. Going beyond Restricted Boltzmann Machines (RBMs) and Autoencoders [64,65], recent years have seen the rise of ‘self-supervised learning’, that also utilizes unlabeled data for representation learning. These methods form the ‘pretrain-then-finetune’ training paradigm and have resulted in significant performance improvements over end-to-end training, especially when large-scale annotations are not available [15,16].

Multi-task self-supervision introduced self-supervised learning to wearable-based activity recognition by performing transformation discrimination in a multi-task setting [15]. Subsequently, SelfHAR combined self-training with transformation discrimination by applying knowledge distillation to train a teacher network with labeled data. The teacher is then used to pseudo-label the unlabeled data, following which the confident samples are combined with the labeled dataset for transformation discrimination. Transformers were explored for self-supervision in [66], by training to reconstruct only randomly masked timesteps of windows of sensor data from mobile phones. Extending this setup, spatio-temporal masking was explored in [67]. Contrastive Predictive Coding (CPC) was adopted and applied to wearable sensor data in [68], where future timestep prediction was performed under contrastive learning settings.

Siamese contrastive learning using the SimCLR framework [69] was explored in [70]. The input windows are randomly augmented in two different ways and comprise the positive pairs, whereas the remaining pairs are the negative pairs. After pre-training with SimCLR, ref. [71] improves clustering performance by leveraging nearest neighbors. SimSiam [72] and BYOL [73] also have a Siamese setup, albeit are not trained with contrastive learning. Ref. [16] studies the aforementioned methods and performs an assessment of the state-of-the-field of self-supervised human activity recognition by evaluating them on a collection of tasks, in order to understand their strengths and shortcomings. Similarly, ref. [74] explores these contrastive learning tasks and studies suitable augmentations and architectures for effective performance. The contrastive setup has also been extended to multiple sensors through approaches such as Learning from the best [75], ColloSSL [76], and COCOA [77]. Analyzing self-supervised methods, ref. [78] examines the pre-training data efficiency, i.e., the minimal quantities of pre-training data required for effective wearable-based self-supervised learning. Enhancements to wearable-based CPC were investigated in [30], by considering three components: the encoder architecture, the autoregressive network, and the future timestep prediction task. The resulting ‘Enhanced CPC’ demonstrates substantial improvements over the original framework [68] as well as outperforms state-of-the-art self-supervision on four of six target datasets. This superior performance, coupled with the fully convolutional architecture (which improves the parallelizability), motivates the use of Enhanced CPC as the base for discretization.

For all methods detailed above, self-supervision results in dense (continuous-valued), high-dimensional representations of data. In contrast, we propose to perform *discrete* representation learning, as it derives a collection of symbolic representations, aiding in the lower-level analysis of human movements while also performing comparably to state-of-the-art self-supervision.

3. Methodology

In this paper, we introduce the discrete representation learning framework for wearable sensor data, with the ultimate goal of improved activity recognition performance and better analysis of human movements. This paper represents the first step towards this

goal, which effectively demonstrates the proof of concept for the effectiveness of learned discretization, which warrants the aforementioned “return to discretized representations”. Based on our framework, we explore the potential and next steps for discretized human activity recognition. An overview of discretization is shown in Figure 1, which involves mapping windows of time-series accelerometer data to a collection of discrete ‘symbols’ (which are represented by strings of numbers).

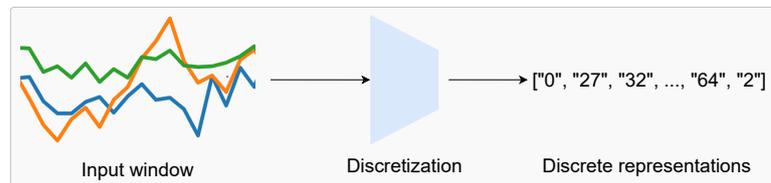


Figure 1. Discrete representation learning: given a window of accelerometer data as input, the output is a collection of enumerated symbols. We map short spans of continuous-valued sensor data to a list of ‘symbols’ (i.e., the numbers in the figure). Therefore, we obtain the “strings of motion”, as the symbols are discrete.

Following the self-supervised learning paradigm, our approach contains two stages: (i) pre-training, where the network learns to map unlabeled data to a codebook of vectors, resulting in the discrete representations; and (ii) fine-tuning/classification, which utilizes the discrete representations as input for recognizing activities. In order to enable the mapping, we apply vector quantization (VQ) to the Enhanced CPC framework [30]. Therefore, the base of the discretization process is self-supervision, where the loss from the pretext task is added to the loss from the VQ module in order to update the network parameters as well as the codebook vectors.

To this end, we first detail the self-supervised pretext task, Enhanced CPC, and describe how the VQ module can be added to it. With the aim of quantitatively measuring the utility of the representations, we perform activity recognition using discrete representations derived from target-labeled datasets. Therefore, we also discuss the classifier network used for such evaluation, and clarify how the setup is different from state-of-the-art self-supervision for wearables.

3.1. Discrete Representation Learning Setup

The setup for learning discrete representations of human movements contains two parts: (i) the self-supervised pretext task; and (ii) the vector quantization (VQ) module. We utilize the Enhanced Contrastive Predictive Coding (CPC) framework [30] as the self-supervised base, which comprises the prediction of multiple future timesteps in a contrastive learning setup. By predicting farther into the future, the network can capture the slowly varying features, or the long-term signal present in the sensor data while ignoring local noises, which is beneficial for representation learning [79]. We borrow notation from [27] for the method description below.

The aim of the Enhanced CPC [30] framework is to investigate three modifications to the original wearable-based CPC framework: (i) the convolutional encoder network; (ii) the Aggregator (or autoregressive network); and (iii) the future timestep prediction task. First, the encoder from [68] is replaced with a network with higher striding (details below), resulting in a reduction in the temporal resolution. In addition, a causal convolutional network is used to summarize previous latent representations into a context vector instead of the GRU-based autoregressive network. Finally, the future timestep prediction is performed at every context vector instead of utilizing a random timestep to make the prediction. These changes, put together, substantially improve the performance of the learned Enhanced CPC representations, compared to state-of-the-art methods. In what follows, we provide the architectural details and a detailed description of the technique.

As shown in Figure 2, we utilize a convolutional encoder to map windows of sensor data to latent representations $f : \mathcal{X} \mapsto \mathcal{Z}$ (called z-vectors). It comprises four blocks, each

containing a 1D convolutional network followed by the ReLU activation and dropout with $p = 0.2$. The layers consist of (32, 64, 128, 256) channels, respectively, with a kernel size of (4, 1, 1, 1) and a stride of (2, 1, 1, 1). The encoder output frequency is 24.5 Hz, as we obtain 49 z -vectors for each window of 100 timesteps (i.e., two seconds of data at 50 Hz). Therefore, we obtain one z_t for approx. every *two* timesteps of data. By adjusting the convolutional encoder architecture appropriately, the frequency can be adjusted to increase or reduce relative to the base setup detailed above (see Section 5.4). In addition, the convolutional encoder can also be modified for training on data recorded at higher sampling rates (i.e., >50 Hz) in order to maintain an output frequency of z -vectors at 24.5 Hz.

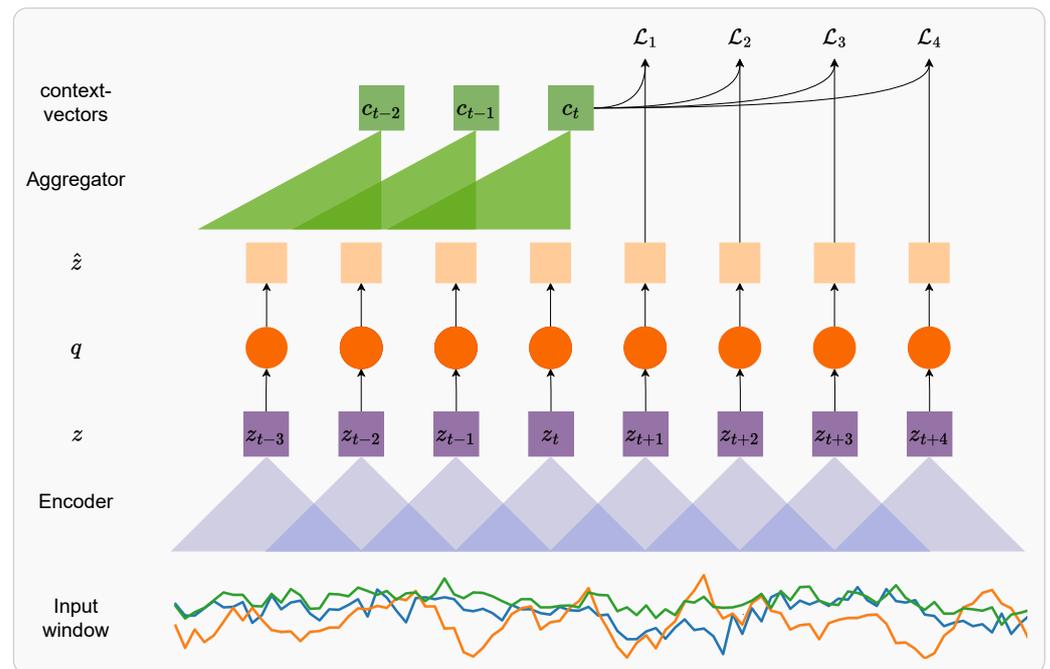


Figure 2. Overview of discrete representation learning for HAR using wearables: we combine a contrastive future timestep prediction problem with vector quantization to map spans of sensor data to a codebook of vectors. The index of the codebook vector closest to each z_t functions as the discrete representation.

The quantization module ($q : \mathcal{Z} \mapsto \hat{\mathcal{Z}}$) replaces each z_t with $\hat{z} = e_i$, which is the index of the closest codebook vector (also called the codeword), from a fixed size codebook $e \in \mathbb{R}^{V \times d}$, containing V representations of size d (details in Section 3.1.1). We utilize the online K-means-based quantization from [28], which is similar to the vector quantized autoencoder [29] detailed originally in [29].

Following Enhanced CPC [30], a causal convolutional network called the ‘Aggregator’ is used for summarizing previous timesteps of encoded representations $\hat{z}_{\leq t}$ ($g : \hat{\mathcal{Z}} \mapsto \mathcal{C}$) into the context vectors c_t , which are used to predict multiple future timesteps. This enables improved parallelization due to the convolutions and results in faster training times. Each block in the Aggregator has 256 filters with dropout $p = 0.2$, layer normalization, and residual connections between layers, as utilized in [28]. For each causal convolution layer in successive blocks, the stride is set to 1, whereas the kernel sizes are consecutively increased from 2. The network is once again trained to identify the ground truth z_{t+k} , which is k steps in the future from a collection of negatives sampled randomly from the batch for every c_t in the window. Such a setup was first introduced in VQ-Wav2vec [28], where two quantization approaches—Gumbel softmax [48] and K-means [28,29]—were studied for their effectiveness towards better speech recognition. In our work, however, preliminary explorations revealed the higher effectiveness of the online K-means-based quantization, described below.

3.1.1. K-Means Quantization

As detailed previously, the codebook has a size of $V \times d$, where V is the number of variables in the codebook, and d is their dimensionality. The vector quantization procedure allows for a differentiable process to select codebook indices. As shown in Figure 3, the nearest neighbor codebook vector to any z -vector in terms of the Euclidean distance is chosen, yielding $i = \operatorname{argmin}_j \|z - e_j\|_2^2$. The z -vector is replaced $\hat{z} = e_i$, which is the codebook vector at the index i . As mentioned in [29], this process can be considered a non-linearity that maps latent representations to one of the codebook vectors.

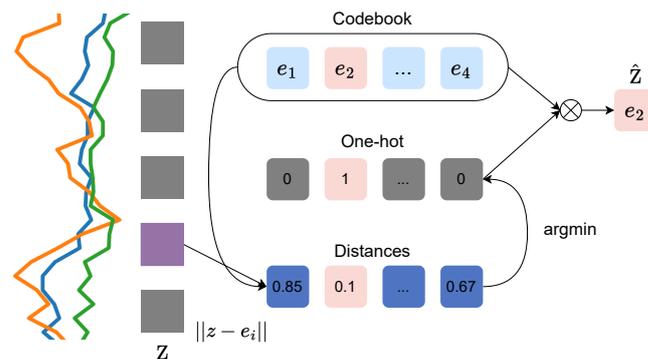


Figure 3. Visualizing K-means-based quantization: for each z -vector, the l_2 distance is computed to the codebook of vectors e_i . The index of the nearest codebook vector comprises the discrete representation, whereas the nearest vector itself is passed as the output (i.e., \hat{z} -vector). This figure has been adopted and adapted from [27].

As choosing the codebook indices does not have a gradient associated with it, the straight-through estimator [80] is employed to simply copy gradients from the Aggregator input $q(z)$ to the encoder output $f(x)$. Therefore, the forward pass comprises the selection of the closest codebook vector, whereas during the backward pass, the gradient becomes copied as-is to the encoder. The parameters are updated using the future timestep prediction loss as well as two additional terms:

$$\mathcal{L} = \sum_{k=1}^K \mathcal{L}_k^{CPC} + (\|sg(z) - \hat{z}\|^2 + \gamma \|z - sg(\hat{z})\|^2) \quad (1)$$

where $sg(x) \equiv x$, $\frac{d}{dx} sg(x) \equiv 0$ is the stop gradient operator, k is the future timestep, and γ is a hyperparameter. Due to the straight-through estimation, the codebook does not obtain any gradients from \mathcal{L}^{CPC} . However, the second term $\|sg(z) - \hat{z}\|^2$ moves the codebook vectors closer to the z -vectors, whereas the third term $\|z - sg(\hat{z})\|^2$ ensures that z -vectors are close to a codeword. Therefore, the Aggregator network is updated via the first loss term, whereas the convolutional encoder is optimized by the first and third loss terms. The codebook vectors are initialized randomly and updated using the second loss term. This is visualized in Figure A3 in Appendix A. The weighting term γ is set to 0.25 as utilized in [27,29], as we obtained good performance.

3.1.2. Preventing Mode Collapse

As discussed in [28], replacing z by a single entry e_i from the codebook is prone to mode collapse, where very few (or only one) codebook vectors are actually used. This leads to very poor outcomes due to a lack of diversity in the discrete representations. To mitigate this issue, ref. [28] suggests independent quantization of partitions, such that $z \in \mathbb{R}^d$ is organized into multiple groups G using the form $z \in \mathbb{R}^{G \times (\frac{d}{G})}$. Each row is represented by an integer index, and the discrete representation is given by indices $i \in [V]^G$, where V is the number of codebook variables for the particular group and each element i_j is the index of a codebook vector. For each of the groups, the vector quantization is applied and the

codebook weights are not shared between them. During pre-training, we utilize $G = 2$ (as per [28]), and $V = 100$, resulting in a possible V^G possible codewords. In practice, the number of unique discrete representations is generally significantly smaller than 100^2 .

3.2. Classifier Network

As the obtained representations (or symbols) are discrete in nature, applying a classifier directly is not possible. Therefore, we utilize an established setup from the natural language processing domain (which also deals with discrete sequences) to perform activity recognition, shown in Figure 4.

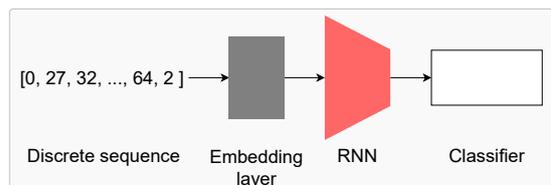


Figure 4. Performing classification using the discrete representations: the sequences of symbolic representations (indexed) are first passed through a learnable embedding layer. Subsequently, an RNN network (GRU or LSTM) is utilized along with an MLP network for classifying the sequences into the activities of interest.

First, the discrete representations are indexed, i.e., assigned a number based on the total number of such symbols present in the data. For each window of symbols, we append the $\langle START \rangle$ and $\langle END \rangle$ tokens to the beginning and end of the window. The dictionary also contains the pad $\langle PAD \rangle$ and unknown $\langle UNK \rangle$ tokens, which represent padding (sequences of differing lengths can be padded to a common length) and unknown (symbols present during validation/test but not during training, for example). The indexed sequences are used as input to a learnable embedding layer (shown in grey in Figure 4), followed by an LSTM or GRU network of 128 nodes and two layers with dropout ($p = 0.2$). Subsequently, a MLP network identical to the classifier network from [16] is applied. It contains three linear layers of 256, 128, and $num_classes$ units with batch normalization, ReLU activation, and dropout in between.

4. Setup

In Section 3, we introduced an overview of our framework for deriving discrete representations from sensor data and for performing a quantitative evaluation using activity recognition. Here, we describe the setup utilized to learn such representations, including the datasets utilized for pre-training and evaluation (Section 4.1), the data pre-processing (Section 4.2), and the implementation details (Section 4.3). Put together, these provide an overview of the practical details vital for learning and utilizing discrete representations for sensor-based HAR.

4.1. Datasets

Both pre-training and classification are performed using data from a single accelerometer, as we can reasonably expect a single wearable to be feasible in most scenarios. Pre-training is performed using the Capture-24 dataset, which contains a single wrist-worn accelerometer. We chose Capture-24 primarily due to its large scale and recording setup: it contains around 2500 h of data from 151 participants in daily living conditions (i.e., in the wild), thereby not limiting the types of movements and activities recorded. In addition, prior works such as [16,30] have utilized it as the base for self-supervised pre-training, allowing us to compare our results against those works. Based on the assessment framework in [16], the performance of the discrete representations is evaluated on target datasets collected at the wrist, waist, and leg, albeit we utilize two datasets per location (unlike [16], which uses three). The source (Capture-24) and target datasets are described in detail in

the Appendix (Appendix A.1) and summarized in Table 1. As in [16], we downsample all datasets to 50 Hz.

Table 1. Summary of the datasets used in our study. Capture-24 is the source dataset (wrist) whereas the others comprise the target, spread across three sensor locations—the wrist, waist, and leg/ankle (adopted/adapted with permission from [16]).

Dataset	Location	# Users	# Act.	Activities
Capture-24 [31–33]	Wrist	151	N/A	Free living
HHAR [81]	Wrist	9	6	Biking, sitting, going up and down the stairs, standing, and walking
Myogym [1]	Wrist	10	31	Seated cable rows, one-arm dumbbell row, wide-grip pulldown behind the neck, bent over barbell row, reverse grip bent-over row, wide-grip front pulldown, bench press, incline dumbbell flyes, incline dumbbell press and flyes, pushups, leverage chest press, close-grip barbell bench press, bar skullcrusher, triceps pushdown, bench dip, overhead triceps extension, tricep dumbbell kickback, spider curl, dumbbell alternate bicep curl, incline hammer curl, concentration curl, cable curl, hammer curl, upright barbell row, side lateral raise, front dumbbell raise, seated dumbbell shoulder press, car drivers, lying rear delt raise, null
Mobiact [82]	Waist/Trousers	61	11	Standing, walking, jogging, jumping, stairs up, stairs down, stand to sit, sitting on a chair, sit to stand, car step-in, and car step-out
Motionsense [83]	Waist/Trousers	24	6	Walking, jogging, going up and down the stairs, sitting and standing
MHEALTH [84,85]	Leg/Ankle	10	13	Standing, sitting, lying down, walking, climbing up the stairs, waist bend forward, frontal elevation of arms, knees bending, cycling, jogging, running, jump front and back
PAMAP2 [86]	Leg/Ankle	9	12	Lying, sitting, standing, walking, running, cycling, nordic walking, ascending and descending stairs, vacuum cleaning, ironing, rope jumping

4.2. Data Pre-Processing

For pre-training, the sampling rate of Capture-24 is reduced to 50 Hz by sub-sampling so as to reduce the computational load and training times (identical to [16]). We also downsample all target datasets to 50 Hz via sub-sampling (if they were higher originally), as it was shown in [16] that matching the sampling rates between pre-training and fine-tuning is important for optimal performance. Following [16], the window size is set to 2 s with an overlap of 0 (for Capture-24) and 50% (for target datasets), in order to ensure that both long- and short-term activities are sufficiently captured in any randomly picked window. For Capture-24, the dataset is split randomly by participants at a 90:10 ratio for training and validation. The train split was normalized to have zero mean and unit variance, and the resulting means and variances were applied to the validation split as well. Further, we only pre-train on randomly sampled 10% of the windows from the train split, as it was shown to have comparable performance to using the entire split in [16], thereby reducing the time taken for pre-training.

For evaluation, the target datasets are separated into five folds: the first fold is split at an 80:20 ratio by the participant into the train-val and test sets. The train-val set is once again partitioned randomly by participant IDs at an 80:20 ratio into the training and validation splits. For the remaining folds, 20% of the participants are chosen randomly to be the test set so that no participant appears in more than one test set across five folds. The train and validation splits are constructed from the remaining participants (i.e., participants that are not a part of the test set), once again at an 80:20 ratio. The means and variances from the Capture-24 train split are also applied to all sets from the target datasets for improved performance (as per [16]).

4.3. Implementation Details

All models were implemented using the Pytorch framework [87]. For the self-supervised baselines, including multi-task self-supervision, Autoencoder, SimCLR, CPC, and Enhanced CPC, we report the performance detailed in the original Enhanced CPC paper [30].

For the discrete version of CPC, we set the learning rate and L2 regularization during pre-training to 10^{-4} and tune over the number of convolutional aggregator layers $\in \{2, 4, 6\}$.

The loss weighting parameter, γ , is set to 0.25 as recommended in [27,29]. In addition, we find that a prediction horizon of $k = 10$ with number of negatives = 10, is sufficient for effective training. For most experiments (save Section 5.4), each symbol spans approx. two timesteps, as we found it to have a good balance between the resulting pre-training times and the temporal resolution.

The pre-training is performed for a maximum of 50 epochs, but early stopping with a patience of 5 epochs is also employed to terminate training if the validation loss does not improve. A cosine learning rate schedule is employed—first, the learning rate is warmed up linearly to 10^{-4} (as mentioned above) for a duration of 8% of the total number of updates. Subsequently, the learning rate is decayed to zero using a cosine function. The early stopping only begins after 20 epochs in order to ensure the completion of warmup and sufficient training before the termination of pre-training. The Adam [88] optimizer is utilized with a batch size of 128.

The evaluation with the RNN classifier is also performed for 50 epochs, where the learning rate and L2 regularization are tuned over $\{1 \times 10^{-3}, 1 \times 10^{-4}, 5 \times 10^{-4}\}$ and $\{0, 1 \times 10^{-4}, 1 \times 10^{-5}\}$, respectively. Once again, the Adam optimizer is utilized, with a batch size of 256. The learning rate is decayed by a factor of 0.8 every 10 epochs. We average the validation F1-scores across the folds in order to identify the best performing hyperparameter combination. The corresponding average test set F1-score across the folds (for the best hyperparameters) is reported in Table 2, where five randomized runs are also performed. The best performing hyperparameters for GRU-based evaluation are listed in Table A1 in Appendix A for reference.

Table 2. Activity recognition performance: we report the mean and standard deviation of the five fold test F1-score across five randomized runs. We observe that discrete representations show comparable if not better performance to self-supervision on three of the benchmark datasets, indicating the capabilities of the learned symbols. The best performing technique overall for each dataset is denoted in **green**, whereas the best unsupervised method is in **bold**. Therefore, methods with **green** are the best method overall and the best unsupervised method as well. The performance for the methods with * was obtained from [30].

Method	Wrist		Waist		Leg	
	HHAR	Myogym	Mobiact	Motionsense	MHEALTH	PAMAP2
Supervised baselines						
Conv. classifier *	55.63 ± 2.05	38.21 ± 0.62	78.99 ± 0.38	89.01 ± 0.89	48.71 ± 2.11	59.43 ± 1.56
DeepConvLSTM *	52.37 ± 2.69	39.36 ± 1.56	82.36 ± 0.42	84.44 ± 0.44	44.43 ± 0.95	48.53 ± 0.98
GRU classifier *	45.23 ± 1.52	36.38 ± 0.60	75.74 ± 0.60	87.42 ± 0.52	44.78 ± 0.47	54.35 ± 1.64
Self-supervision + MLP classifier						
Multi-task self. sup *	57.55 ± 0.75	42.73 ± 0.49	72.17 ± 0.38	86.15 ± 0.42	50.39 ± 0.72	60.25 ± 0.72
Autoencoder *	53.64 ± 1.04	46.91 ± 1.07	72.19 ± 0.35	83.10 ± 0.60	40.33 ± 0.37	59.69 ± 0.72
SimCLR *	56.34 ± 1.28	47.82 ± 1.03	75.78 ± 0.37	87.93 ± 0.61	42.11 ± 0.28	58.38 ± 0.44
CPC *	55.59 ± 1.40	41.03 ± 0.52	73.44 ± 0.36	84.08 ± 0.59	41.03 ± 0.52	55.22 ± 0.92
Enhanced CPC *	59.25 ± 1.31	40.87 ± 0.50	78.07 ± 0.27	89.35 ± 0.32	53.79 ± 0.83	58.19 ± 1.22
Discrete representations + RNN classifier						
VQ CPC + LSTM class.	60.76 ± 1.09	29.62 ± 0.52	76.34 ± 0.30	89.06 ± 0.24	48.86 ± 0.34	55.28 ± 0.34
VQ CPC + GRU class.	60.26 ± 0.83	31.65 ± 0.29	77.78 ± 0.17	89.23 ± 0.23	49.01 ± 0.30	56.92 ± 0.26

5. Results

Through the work presented in this paper, we aim to demonstrate the potential of learning discrete representations of human movements. For this, we first evaluate their effectiveness for HAR via a simple recurrent classifier. The performance is contrasted against established supervised baselines (such as DeepConvLSTM), as well as the state-of-the-art for representation learning, which is self-supervision. Subsequently, we contrast the impact of *learning* the discrete representations rather than computing via prior methods involving SAX. This is followed by an exploration into the discrete representation learning framework, where we study the impact of controlling the resulting alphabet size (i.e.,

the fidelity of the representations) and the effect of the duration of sensor data on each symbol's representation. Finally, we apply self-supervised pre-training techniques designed for discrete sequences in order to study whether such tasks can further help improve recognition performance. Overall, these experiments are designed to not only study whether discrete representations can be useful but also to derive a deeper understanding of their working.

5.1. Activity Recognition with Discrete Representations

First, we evaluate the performance of the discrete representations for recognizing activities from windows of discrete sequential data. After the pre-training is complete, we perform inference to obtain the discrete representations and utilize the setup detailed in Section 3.2 for classification. The performance is compared against diverse self-supervised learning techniques, which represent the state of the art of representation learning in HAR (as shown in [16]), including: (i) multi-task self-supervision, which utilizes transformation discrimination; (ii) Autoencoder, reconstructing the original input through an additional decoder network; (iii) SimCLR, contrasting two augmented versions of the same input window against negative pairs from the batch; (iv) CPC, which uses multiple future timestep prediction for pre-training; and (v) Enhanced CPC, as before but with improvements to the CPC framework on the encoder, aggregator, and future prediction tasks [30]. Here, the encoder weights are frozen and only the MLP classifier is updated with annotations.

DeepConvLSTM, a Conv. classifier with the same architecture as the encoder for Multi-task, Autoencoder, and SimCLR, along with a GRU classifier function as the end-to-end training baselines. We perform five fold cross validation and report the performance for five randomized runs in Table 2. The comparison is performed on six datasets across sensor locations (Capture-24 is collected at the wrist, whereas the target datasets are spread across the wrist, waist, and leg) and activities (which include locomotion, daily living, health exercises, and fine-grained gym exercises).

For the waist-based Mobiact, which covers locomotion-style activities along with transitional classes such as stepping in and out of a car, the discrete representation learning performs comparably or better than all methods, obtaining a mean of 77.8%. However, for Motionsense, the performance is similar to the best performing model overall, which is Enhanced CPC, once again outperforming other self-supervised and supervised baselines. Considering the leg-based PAMAP2 dataset, VQ-CPC obtains lower performance and is similar to the GRU classifier. For MHEALTH as well, the performance drops significantly compared to Enhanced CPC, showing a reduction of around 4.8%, yet outperforming the Autoencoder, SimCLR, Multi-task, and CPC.

Finally, we consider wrist-based datasets such as HHAR and Myogym. HHAR comprises locomotion-style activities and the discrete representations improve the performance over Enhanced CPC by around 1.5%, thereby constituting the best option for wrist-based recognition of locomotion activities. Interestingly, the discretization results in poor features for classifying fine-grained gym activities, with the performance dropping significantly compared to other self-supervised methods. Enhanced CPC also sees substantially lower performance than SimCLR, likely due to the increased striding in the encoder, which results in a latent representation for approx. every second timestep, thereby negatively impacting the recognition of activities such as fine-grained curls and pulls. In addition, the discretization results in a smaller, finite codebook, which is a loss in temporal resolution compared to continuous-valued high-dimensional features. This is detrimental to Myogym, resulting in poor performance.

Therefore, the discrete representations can result in effective recognition of locomotion-style and daily living activities, and overall perform the best (or similar to the best) on three benchmark datasets, at the wrist and waist. The loss in resolution due to mapping the continuous-valued sensor data to a finite collection of codebook vectors (and their indices) does not have a significant negative impact on locomotion-style activities but is detrimental for recognizing fine-grained movements (as present in Myogym for example).

In addition, the effective performance across sensor locations indicates the capability of the discrete representation learning process and shows its promise for sensor-based HAR. This result presents practitioners with a new option for activity recognition, with comparable performance and potentially lowered data upload costs, as the discretized representations result in more compressed data than continuous-valued sensor readings.

5.2. Comparison to Established Discretization Methods

In this experiment, we compare the performance of SAX, which is an established method for discretizing uni-variate time-series data, and SAX-REPEAT [37], which utilizes SAX for discretizing multi-channel time-series data. For appropriate comparison, SAX also results in one symbol for every second timestep of sensor data, with an alphabet size of 512. SAX-REPEAT separately applies SAX to each channel of accelerometer data, resulting in tuples of indices for every second timestep. As utilizing the tuples as-is results in a possible dictionary size of 512^3 , SAX-REPEAT performs K-Means clustering (with $k = 512$) on the tuples in order to maintain an alphabet size of 512, where the cluster indices function as the discrete representation. The same classifier setup (Section 3.2) is utilized for activity recognition (including the parameter tuning for classification) and five random runs of the five fold validation F1-score is detailed in Table 3. The comparison is drawn against the learned discrete representation method, which is VQ-CPC.

Table 3. Comparing the performance of the proposed discrete representation learning technique to SAX and SAX-REPEAT (multi-variate version of SAX): SAX and SAX-REPEAT perform poorly relative to VQ-CPC, demonstrating that learning the discrete representations results in better recognition. The best performing models are shown in green.

Method	Wrist		Waist		Leg	
	HHAR	Myogym	Mobiact	Motionsense	MHEALTH	PAMAP2
SAX + LSTM class.	43.78 ± 0.96	14.22 ± 0.40	66.45 ± 0.18	70.14 ± 0.36	41.04 ± 0.58	47.61 ± 1.04
SAX-REPEAT + LSTM class.	39.17 ± 0.69	28.65 ± 0.69	71.73 ± 0.74	71.31 ± 0.74	38.89 ± 0.48	43.30 ± 1.83
VQ-CPC + LSTM class.	60.76 ± 1.09	29.62 ± 0.52	76.34 ± 0.30	89.06 ± 0.24	48.86 ± 0.34	55.28 ± 0.34
VQ-CPC + GRU class.	60.26 ± 0.83	31.65 ± 0.29	77.78 ± 0.17	89.23 ± 0.23	49.01 ± 0.30	56.92 ± 0.26

For all datasets, the SAX baseline performs poorly compared to the learned discrete representations, showing a reduction of over 10% for HHAR, Myogym, and Motionsense, and a smaller reduction for Mobiact, MHEALTH, and PAMAP2. This can be expected as SAX utilizes the magnitude of the accelerometer data as the input, thereby reducing three channels to one and losing information about the direction of movement. Considering SAX-REPEAT next, we see that it shows worsened performance to SAX on HHAR, MHEALTH, and PAMAP2. For Mobiact, the performance is only 6% lower than VQ-CPC + GRU classifier, whereas for the other datasets, the difference is greater. Only on Myogym, the performance is better than VQ-CPC, albeit substantially lower than the state-of-the-art self-supervised as well as end-to-end training methods. The lower performance for SAX and SAX-REPEAT for Myogym also indicates that discretization is not a good option for fine-grained activities. Our experiments clearly show that SAX and SAX-REPEAT are worse at recognizing activities compared to VQ-CPC. Further, the reduction in performance of SAX-REPEAT relative to SAX on HHAR, MHEALTH, and PAMAP2, indicates that modifying SAX to apply to multi-variate data is challenging. Overall, Table 3 shows that the traditional methods are not effective for discretizing accelerometer data, and that learning a codebook in an unsupervised, data-driven way results in a better mapping of sensor data to discrete representations.

5.3. Effect of the Learned Alphabet Size

One of the advantages of discrete representation learning via vector quantization is the control over the size of the learned dictionary. It can be set depending on the required fidelity of the learned representations and the capacity of the computation power available

for classification. For applications where the separation of activities requires a small dictionary (e.g., 8 or 16 symbols), we can accordingly set the dictionary size and thereby save computation power during classification. For our base setup (Table 2), we utilize independent quantization of partitions of the vectors, resulting in a possible 100^2 dictionary size. Here, we explicitly control the dictionary size by setting the number of groups to 1 and varying the number of variables (i.e., the number of codebook vectors) between (32, 64, 128, 256, 512). We also note that the final dictionary size can be lower than the codebook size and depends on the underlying movements and sensor data. We perform activity recognition on the resulting discrete representations of windows of sensor data using the best performing models from Table 2, albeit with increasing alphabet sizes. The results from this experiment are tabulated in Table 4. A similar analysis was also performed in VQ-APC [51].

Table 4. Studying the impact of the maximum dictionary size on activity recognition: we explicitly limit to {32, 64, 128, 256, 512} codebook vectors and study how performance is affected by the applied constraint. For HHAR, we observe a substantial increase of over 7% by limiting the size to 64. For MHEALTH and PAMAP2, the improvements are more modest. This indicates that a more deliberate choice of dictionary size can result in further performance increases. The mean dictionary size across the five folds for the base setup of the VQ CPC + GRU classifier is shown in brackets in the last row. The best performing models are shown in green.

Max. Dict. Size	Wrist		Waist		Leg	
	HHAR	Myogym	Mobiact	Motionsense	MHEALTH	PAMAP2
32	11.58 ± 0.12	2.80 ± 0.00	29.89 ± 0.24	37.46 ± 0.26	14.38 ± 0.28	21.26 ± 0.37
64	67.62 ± 0.21	20.78 ± 0.28	74.19 ± 0.19	89.70 ± 0.18	48.77 ± 0.36	58.06 ± 0.51
128	57.71 ± 0.87	27.81 ± 0.28	75.73 ± 0.41	79.49 ± 0.24	49.62 ± 0.51	56.56 ± 0.66
256	60.21 ± 0.66	18.13 ± 0.33	75.53 ± 0.24	90.28 ± 0.28	47.71 ± 0.49	57.12 ± 0.34
512	60.41 ± 0.55	12.92 ± 0.44	64.25 ± 0.51	71.73 ± 0.23	46.81 ± 0.80	53.40 ± 0.60
VQ CPC + GRU classifier	60.26 ± 0.83 (127.8)	31.65 ± 0.29 (155)	77.78 ± 0.17 (148)	89.23 ± 0.23 (140)	49.01 ± 0.30 (130.2)	56.92 ± 0.26 (140.4)

First, we notice that having a max. alphabet size of 32 results in poor performance. Such a small dictionary size provides limited descriptive power for the representations and therefore leads to significant drops in performance relative to the base setup of utilizing multiple groups during quantization (see Section 3.1.2). Along the same lines, having too large a dictionary size is also slightly detrimental (max dict size = 512), as it can lead to long-tailed distributions of the symbolic representations and the network starting to pay attention to noises instead.

We obtain the highest performance when the max dictionary sizes are 64, 128, or 256. For HHAR, the constraint on the dictionary size results in an increase of over 7% relative to the base setup (VQ CPC + GRU classifier). For Myogym and Mobiact, however, not constraining the resulting dictionary sizes is the best option, with clear increases over the constrained models. For Motionsense and PAMAP2, controlling the learned alphabet size results in modest performance improvements of 1%, whereas for MHEALTH, it is around 0.6%. Clearly, with reducing codebook sizes, the model is forced to choose what information to discard and what to encode [51]. This process can result in higher performance as the network can more efficiently learn to ignore irrelevant information (such as noise) and picks up more discriminatory information.

Next, we consider the mean dictionary size across all folds obtained by utilizing groups = 2 (as in Table 2; see Section 3.1.2 for reference). For all target datasets, the size <160 symbols, emphasizing that effective recognition can be obtained using just around 130–160 symbols. This is encouraging, as downstream tasks such as gesture or activity spotting, can be performed more easily with a smaller dictionary size. The importance of creating groups during discretization is also visible, as it results in the highest performance

for two target datasets, along with comparable performance for three datasets, without having to further tune the dictionary size as a hyperparameter.

5.4. Impact of the Encoder’s Output Frequency

In Section 3, we detailed the architecture for learning discrete representations of human movements. The convolutional encoder results in approximately one latent representation per two timesteps of sensor data. With appropriate architectural modifications, we can increase or reduce the output frequency of the encoder. Intuitively, a lower output frequency can be problematic as too much motion (and variations of motion) can be mapped to each symbol. When this occurs, nuances in movements are not captured well by the symbolic representations. In this experiment, we vary the output frequency and study the impact on performance. The convolutional encoder is modified accordingly: (i) for an output frequency of 50 Hz (i.e., no downsampling relative to the input), we change the stride of the first block to 1 and for the second block, set the kernel size and stride = 1; and (ii) for an output frequency of 11.5 Hz (i.e., further downsampling by two relative to the base setup), the second block also has a kernel size of 4 with stride = 2. We perform activity recognition on the six target datasets, and report the five fold cross validation performance across five randomized classification runs in Table 5.

Table 5. Investigating the impact of the encoder’s output frequency: by adjusting the encoder architecture, we study whether an output frequency of 50 Hz (same as input), or 24.5 Hz (base setup), or 11.5 Hz (approx. halved relative to the base setup) is more suited for the representations. We observe that the base setup of 24.5 Hz results in better performance while also reducing computational costs. The best performing models are shown in green .

Encoder Output Freq.	Wrist		Waist		Leg	
	HHAR	Myogym	Mobiact	Motionsense	MHEALTH	PAMAP2
50 Hz	58.02 ± 0.46	18.30 ± 0.45	77.65 ± 0.43	85.25 ± 0.39	48.43 ± 0.66	51.36 ± 0.91
11.5 Hz	48.95 ± 1.26	16.50 ± 0.31	53.51 ± 0.26	63.68 ± 0.37	32.78 ± 0.66	41.62 ± 0.61
24.5 Hz	60.26 ± 0.83	31.65 ± 0.29	77.78 ± 0.17	89.23 ± 0.23	49.01 ± 0.30	56.92 ± 0.26

As expected, an encoder output frequency of 11.5 Hz (i.e., # timesteps/symbol ≈ 4) results in substantial reductions in performance relative to the base setup (where the output frequency is 24.5 Hz). For HHAR, the drop in performance is around 10%. However, for Myogym, MHEALTH, and PAMAP2, it is over 15%. The waist-based datasets see the highest impact on performance, experiencing a reduction of over 20% with the longer duration mapping. We can reasonably expect that a lower encoder output frequency, will result in further reduction in performance.

We also note that maintaining the same output frequency as the input also causes a drop, albeit smaller, in the test set performance. While this configuration can be utilized for obtaining discrete representations, the training times are considerably higher, while also not resulting in performance improvements. Therefore, an output frequency of 24.5 Hz (relative to an input of 50 Hz) is better, allowing for quicker training while also covering more of the underlying motion.

5.5. NLP-Based Pre-Training with RoBERTa

One of the advantages of converting the sensor data into discrete sequences is that it allows us to apply powerful NLP-based pre-training techniques such as BERT [35], RoBERTa [34], GPT [89], etc., as learned embeddings for the RNN classifier. In addition, the release of new techniques for text-based self-supervision can be accompanied by corresponding updates to the classification of the discrete representations learned from movement data. Therefore, in this experiment, we investigate whether Robustly Optimized BERT Pretraining Approach (RoBERTa) [34] based pre-training on the symbolic representations is useful for improving activity recognition performance. While RoBERTa can increase

the computational footprint of the recognition system, it can be potentially replaced with recent advancements in distilling and pruning BERT models such as SNIP [90], ALBERT [91], and DistillBERT [92] while maintaining similar performance.

First, we extract the symbolic representations on the large-scale Capture-24 dataset (utilizing 100% of the train split), and use it to pre-train two RoBERTa models, called ‘small’ and ‘medium’. The ‘small’ model contains an embedding size of 128 units, a feedforward size of 512 units, and two Transformer [93] encoder layers with eight heads each. On the other hand, the ‘medium’ sized model comprises embeddings of size 256, with a feedforward dimension of 1024, and four Transformer encoder layers with eight heads each. The aim of training models with two different sizes is to investigate whether increased depth results in corresponding performance improvements or not. Following the protocol from Table 2, the performance across five random runs is reported for five fold cross validation is reported in Table 6. As shown in Figure A1 (in Appendix A), the randomly initialized learnable embedding layer is replaced with the learned RoBERTa models, which are frozen. Only the GRU classifier is updated with label information during the classifier training.

Table 6. Evaluating the impact of utilizing pre-trained RoBERTa embeddings (obtained from the discretized Capture-24 dataset) for recognizing activities: we observe that the addition of RoBERTa embeddings results in improvements for **all** target datasets. Further, we achieve the state-of-the-art for self-supervision for three datasets, which cover locomotion and daily living activities. The best performing technique overall for each dataset is denoted in **green**, whereas the best unsupervised method is in **bold**. Therefore, methods with **green** are the best method overall and the best unsupervised method as well. The performance for the methods with * was obtained from [30].

Method	Wrist		Waist		Leg	
	HHAR	Myogym	Mobiact	Motionsense	MHEALTH	PAMAP2
Supervised baselines						
Conv. classifier *	55.63 ± 2.05	38.21 ± 0.62	78.99 ± 0.38	89.01 ± 0.89	48.71 ± 2.11	59.43 ± 1.56
DeepConvLSTM *	52.37 ± 2.69	39.36 ± 1.56	82.36 ± 0.42	84.44 ± 0.44	44.43 ± 0.95	48.53 ± 0.98
GRU classifier *	45.23 ± 1.52	36.38 ± 0.60	75.74 ± 0.60	87.42 ± 0.52	44.78 ± 0.47	54.35 ± 1.64
Self-supervision + MLP classifier						
M-task self. sup *	57.55 ± 0.75	42.73 ± 0.49	72.17 ± 0.38	86.15 ± 0.42	50.39 ± 0.72	60.25 ± 0.72
SimCLR *	56.34 ± 1.28	47.82 ± 1.03	75.78 ± 0.37	87.93 ± 0.61	42.11 ± 0.28	58.38 ± 0.44
Enhanced CPC *	59.25 ± 1.31	40.87 ± 0.50	78.07 ± 0.27	89.35 ± 0.32	53.79 ± 0.83	58.19 ± 1.22
Discrete representations + GRU classifier						
VQ-CPC	60.26 ± 0.83	31.65 ± 0.29	77.78 ± 0.17	89.23 ± 0.23	49.01 ± 0.30	56.92 ± 0.26
VQ-CPC + RoBERTa sml.	62.30 ± 0.68	34.41 ± 0.45	79.42 ± 0.38	91.76 ± 0.22	51.77 ± 0.28	59.34 ± 0.48
VQ-CPC + RoBERTa med.	63.31 ± 0.38	35.16 ± 0.29	78.99 ± 0.33	91.45 ± 0.26	51.59 ± 0.42	59.53 ± 0.38

First, we observe that utilizing the learned RoBERTa embeddings (VQ CPC + RoBERTa small/medium in Table 6) instead of the random learnable embeddings (VQ CPC in Table 6) results in performance improvements for **all** target datasets. This indicates the positive impact of pre-training with RoBERTa. For the small version, the HHAR and Myogym see increases of 2% and 2.8%, respectively. A similar trend is observed for the waist-based Mobiact and Motionsense as well, improving by 1.6% and 2.5%. Finally, the leg-based datasets also see improvements of around 2.5% each. Interestingly, the medium-sized model of RoBERTa shows a similar performance to the small version, except for the wrist-based HHAR and Myogym, where the increase over random embeddings is 3% and 3.5%, respectively. The similar performance demonstrated by the medium version indicates that the increase in model size did not result in corresponding performance improvements, likely because Capture-24 is not large enough to leverage the bigger architecture. Potentially, an even larger dataset (e.g., Biobank [94]) can be utilized for the medium version (or even larger variants).

The advantage of performing an additional round of pre-training via RoBERTa is clearly observed in Table 6, as VQ CPC + RoBERTa outperforms the state-of-the-art self-

supervision on three datasets (HHAR, Mobiact, and Motionsense) by clear margins. For the leg-based datasets, the performance with the addition of RoBERTa is closer to the most effective methods through improved learning of embeddings. This result is promising for wearables applications, as it proves that the rapid advancements from NLP can be applied for improved activity recognition as well.

6. Discussion

In this paper, we propose a return to discrete representations as descriptors of human movements for wearable-based applications. Going beyond prior works such as SAX, we instead learn the mapping between short spans of sensor data and symbolic representations. In what follows, we will first visualize the distributions of the discrete representations for activities across the target datasets and examine the similarities and differences. The latter half of this section contains an introspection of the method itself, along with the lessons learned during our explorations.

6.1. Visualizing the Distributions of the Discrete Representations

We demonstrated that training GRU classifiers with randomly initialized embeddings (Table 2) results in effective activity recognition on five of six benchmark datasets. In addition, deriving pre-training embeddings from the discrete representations via RoBERTa further pushes the performance, exceeding state-of-the-art self-supervision on three datasets. Given their recognition capabilities, we plot the distributions of the discrete representations for each activity, in order to visualize how the underlying movements may be different. This serves as a first check to visually examine whether similar activities such as walking and walking up/downstairs—which may have similar underlying movements—are actually represented in similar discrete representations.

In Figures 5 and A2, we present the histograms of the discrete representations per activity. The y -axis comprises the fraction of the total sum of representations held by each discrete symbol. First, we note that the discrete representations exhibit long-tailed distributions, with a significant portion of representations being used very sparsely (more clearly visible in Figure A2). The impact of such a distribution is challenging to predict, on one hand, the rarely occurring symbols can increase the complexity of the classifiers (and embeddings) due to their numerosity, while on the other, it is likely they capture more niche movements as they may be performed by participants. Such niche movements can potentially help with the classification of less frequent activities. In addition, we also note that the distributions for sitting and standing contain limited variability, as the underlying movements themselves exhibit less motion. This somewhat verifies that the learned discrete representations correspond to the movements themselves, given that a lack of movement is captured in the distribution of representations per activity. Interestingly, the histograms for going up and down the stairs look very similar, while walking also retains similarities to them. Running looks slightly different, with more spreading out across the symbolic representations, therefore indicating that higher variability of underlying movements involving running, which makes sense intuitively. Therefore, the distributions of the discrete representations provide practitioners with an additional tool for understanding human activities as well as the underlying movements. This is a point in favor of discrete representations, as the analysis is possible in conjunction with comparable if not better performance for activity recognition.

6.2. Analyzing the VQ-CPC Discretization Framework

Here, we examine specific components of the framework, in order to understand their impact on both discrete representation learning, as well as on downstream activity recognition. To this end, we consider the following components: (i) *the Encoder network*, where the architecture determines the duration of time covered by each symbol; and (ii) *the self-supervised pretext task*, which acts as the base for the discrete representation learning. In what follows, we study the activity recognition performance for the same

target datasets, albeit replacing the aforementioned components of the framework with suitable alternatives, and examine the performance.

6.2.1. Impact of the Encoder Architecture

Our Encoder architecture is based on the Enhanced CPC framework [30]. As detailed in Section 3.1, it contains four convolutional blocks, with a kernel size of (4,1,1,1), and stride of (2,1,1,1), respectively. Therefore, the resulting z-vectors are obtained approximately once every second timestep (we obtain 49 z-vectors from an input window of 100 timesteps due to the striding). From Table 5, we see that decreasing the encoder output frequency to 11.5 Hz is detrimental to performance. We now conduct a deeper analysis of the design of suitable encoders by considering the following configurations: (i) increasing the kernel size of the first layer to (8, 16), while keeping the architecture otherwise identical to the base setup; and (ii) utilizing an encoder identical to the convolutional encoders of multi-task self-supervision [15] and SimCLR [70]. The learning rate and L2 regularization are identical to the base setup, and we also tune the number of aggregator layers across (2,4,6) layers, as described in Section 4.

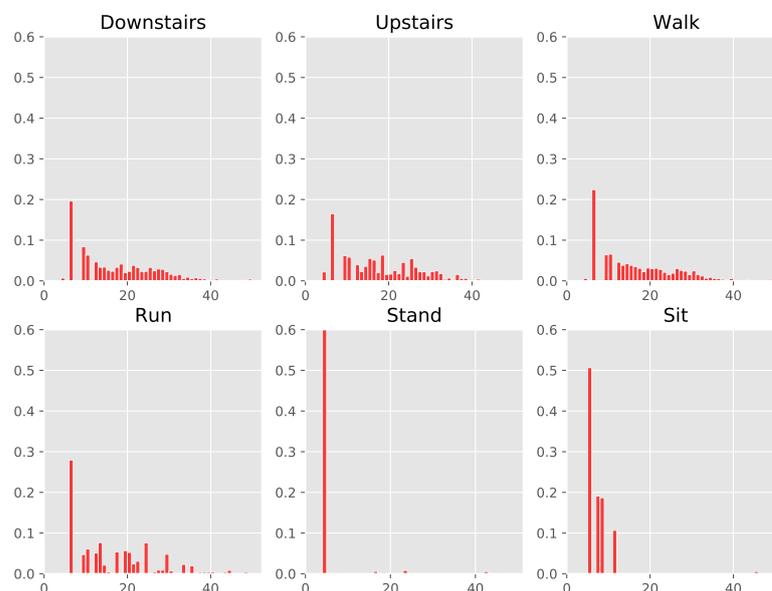


Figure 5. Plotting the histograms per class of the derived discrete representations for the train set from the first fold of Motionsense: The y -axis corresponds to the fraction of the number of occurrences for the symbols computed against the available data. The x -axis comprises the discrete representations, which are numbered. We observe that standing and sitting are covered by only a few symbols, which can be reasoned by the lack of movements in these classes. Walking up and down the stairs is also similar, yet hard to distinguish via visual examination from both walking and running. For clarity, we truncate the y -axis to 0.6 and x -axis to 50 symbols in order to show the plot in more detail. The full figure is available in the Appendix (refer to Figure A2).

In the base setup, movement across four timesteps (0.08 s at 50 Hz) contributes to each symbol. This is increased to (8, 16) timesteps depending on the filter size of the first layer. From Table 7, we see that this is detrimental to HAR. Clearly, it becomes difficult to learn the mapping between eight timesteps (and greater, i.e., ≥ 0.16 s) to symbols, as the underlying movements cover much longer durations and thus become too coarse for symbolic representations. We extend this analysis by applying the encoder from multi-task self-supervision. Ref. [15] instead of the base encoder, for pre-training. As the encoder contains three blocks with filter sizes of (24, 16, 8), a total of 46 timesteps (i.e., 0.92 s of movements) contribute to each symbol. We observe a significant drop in performance as a result, with around 15% reduction for HHAR and approx. 30% decrease for Mobiact. While it is preferable to learn symbols that represent short spans of time, clearly, accurately

mapping longer durations to symbols is a difficult proposition. From our exploration, a filter size of 4 (for the first layer) seems ideal, covering sufficient motion as well as resulting in accurate HAR. This also motivates the architecture of our encoder, where all layers apart from the first one have a filter size of 1. Having multiple layers (after the first) with filter size >1 would result in z-vectors corresponding to longer durations, thereby resulting in reduced performance.

Table 7. Studying the impact of the encoder network architecture on recognition performance: we examine whether a smaller receptive field in the encoder is more preferable for discrete representations. We note that larger receptive fields (e.g., 8 or 16) result in performance reduction compared to the base setup (filt. size = 4). The best performing models are shown in green.

Encoder Arch.	Wrist		Waist		Leg	
	HHAR	Myogym	Mobiact	Motionsense	MHEALTH	PAMAP2
Base setup (filt. size = 4)	60.26 ± 0.83	31.65 ± 0.29	77.78 ± 0.17	89.23 ± 0.23	49.01 ± 0.30	56.92 ± 0.26
Base setup + filt. size = 8	55.15 ± 0.60	16.80 ± 0.44	70.15 ± 0.39	77.95 ± 0.28	47.73 ± 0.26	43.68 ± 0.52
Base setup + filt. size = 16	49.07 ± 0.45	16.59 ± 0.42	71.22 ± 0.12	81.38 ± 0.56	46.18 ± 0.37	44.29 ± 0.50
Multi-task enc. [15]	45.32 ± 1.43	22.31 ± 0.11	48.92 ± 0.23	76.37 ± 0.41	39.60 ± 0.76	47.93 ± 0.76

6.2.2. Effect of the Base Self-Supervised Method on Recognition Performance

Next, we evaluate the applicability and utility of various self-supervised methods to serve as the basis for the discrete representation learning setup. Such analysis enables us to determine which self-supervised method can be utilized, allowing us to provide suggestions for specific scenarios. For example, as discussed in [16], simpler methods such as Autoencoders may be preferable—even though the performance is slightly lower—as they are easier and quicker to train. Furthermore, K-means-based vector quantization (VQ) was also originally introduced in an Autoencoder setup [29]. Therefore, we not only study Autoencoders, but also other baselines such as multi-task self-supervision and SimCLR, for their effectiveness toward functioning as the base for deriving discrete representations. For this analysis we also perform brief hyperparameter tuning for the baseline methods, using the best parameters detailed in [16] and over the number of convolutional aggregator layers $\in \{2, 4, 6\}$ (similar to VQ-CPC). The results from this analysis are given in Table 8.

Table 8. Evaluating the impact of the base self-supervised method on activity recognition: we observe that Enhanced CPC (i.e., the base self-supervision for VQ-CPC) is clearly the most suitable self-supervised method, while simpler techniques such as Autoencoders can also be utilized, albeit with performance reductions. Further, we also investigate whether the VQ-CPC encoder can result in better performance for other methods, and find that to be true for both multi-task self-supervision and SimCLR. The best performing models are shown in green.

Base Self-Supervision	Wrist		Waist		Leg	
	HHAR	Myogym	Mobiact	Motionsense	MHEALTH	PAMAP2
VQ-CPC	60.26 ± 0.83	31.65 ± 0.29	77.78 ± 0.17	89.23 ± 0.23	49.01 ± 0.30	56.92 ± 0.26
VQ-Autoencoder	48.12 ± 0.72	30.61 ± 0.97	73.03 ± 0.59	79.59 ± 0.68	45.61 ± 0.81	48.99 ± 1.37
VQ-Autoencoder + VQ-CPC encoder	50.60 ± 1.05	33.15 ± 0.44	73.26 ± 0.47	75.89 ± 0.62	37.75 ± 0.59	51.23 ± 0.66
VQ-multi-task self-supervision	45.32 ± 1.43	22.31 ± 0.11	48.92 ± 0.23	76.37 ± 0.41	39.60 ± 0.76	47.93 ± 0.76
VQ-multi-task self-supervision + VQ-CPC encoder	58.29 ± 0.69	24.53 ± 0.62	72.87 ± 0.14	80.49 ± 0.60	44.85 ± 0.63	56.28 ± 0.40
VQ-SimCLR	25.34 ± 0.40	11.17 ± 0.17	14.24 ± 0.38	60.67 ± 0.12	9.04 ± 0.09	19.62 ± 0.66
VQ-SimCLR + VQ-CPC encoder	58.49 ± 0.29	2.85 ± 0.01	59.62 ± 0.35	59.54 ± 0.15	41.62 ± 0.32	55.80 ± 0.52

First, we compare the performance of VQ-CPC against adding the VQ module to the Autoencoder setup (“VQ-Autoencoder” in Table 8). For target datasets such as HHAR, Motionsense, and PAMAP2, the drop in performance while utilizing the Convolutional Autoencoder as the base is around 10%. In the remaining target scenarios, the reduction

is lower at around 4%, save for Myogym where the performance is comparable. The established multi-task self-sup. Ref. [15] framework is ill-suited for discrete representation learning, with significant reductions in performance throughout, consistently performing worse by approx. 10% for most datasets and peaking at around 29% for Mobiact. A similar analysis of SimCLR shows an even more substantial reduction in performance, dropping by over 35% consistently. As analyzed in Section 6.2.1, the low performance of SimCLR and multi-task self-sup. is likely due to the encoder architecture itself, which has a large receptive field (see below).

In order to study whether smaller filters are more suitable for other self-supervised methods as well, we replace their encoders with the encoder network from VQ-CPC, and study the impact on the recognition accuracy. For the Autoencoder, the effect is mixed, with the performance increasing slightly for datasets such as HHAR, Myogym, and PAMAP2, whilst reducing for Motionsense and MHEALTH. In the case of multi-task self-sup., we observe that matching the encoder network (to VQ-CPC) has a significant impact on performance, resulting in improvements of approx. 8% for PAMAP2, 13% for HHAR, 24% for Mobiact, and more modest 4–5% for Motionsense and MHEALTH. This clearly shows that large receptive fields such as the one resulting from the original Multi-task encoder are detrimental to discrete representation learning. Furthermore, utilizing a filter size of 4 is also a better option for other methods, including SimCLR. Overall, we observe that the Autoencoder or multi-task self-sup. with the replaced encoder can function as viable alternatives, albeit there is generally a reduction in performance relative to VQ-CPC. This can be useful in some situations as simpler methods may be preferable due to computational constraints.

6.3. Potential Impact beyond Standard Activity Recognition, and Next Steps

This work presents a *proof of concept* in favor of learning discrete representations for sensor-based human activity recognition. We present an alternative data processing and feature extraction pipeline to the HAR community, to be utilized for further application scenarios but also to (once again) jumpstart research into developing discrete learning methods. In what follows, we describe future potential application scenarios where discrete representations can be especially useful.

NLP-Based Pre-training: We observe in Table 6 that adding pre-trained RoBERTa embeddings results in clear improvements over utilizing randomly initialized learnable embeddings for all target datasets. For the locomotion-style and daily living datasets in particular, this results in state-of-the-art performance, which is highly encouraging, as it opens the possibility of adopting more powerful recent advancements from natural language processing for improved recognition of activities. Replacing RoBERTa, larger models such as GPT-2 [95] or GPT-3 [96] or modifications to existing methods (e.g., masking spans of data [27]) can be utilized on larger scale datasets such as UK Biobank [94], leading to potential classification performance improvements. This is promising as advancements in NLP can also result in tandem improvements in sensor-based HAR. In resource-constrained situations, however, works miniaturizing and pruning language models [90–92] can be employed to reduce size while maintaining similar performance.

Activity Summarization: The discrete representations also enable us to utilize established methods from NLP for unsupervised text summarization. They typically involve extracting key information/sentences from texts via methods like graphs [97,98] and clustering [99,100]—i.e., they are extractive rather than generative, as they do not have access to paired summaries for training. Therefore, we can utilize such summarization techniques in order to extract the most *informative* sensor data, allowing us to, e.g., reduce noise (by removing unnecessary data), or summarize the important movements during the hour/day, etc., for understanding routines.

Sensor Data Compression: The discretization results in symbolic representations, which are essentially the ‘strings of human movements’, effectively compressing the original data requiring substantially less memory for storing relative to multi-dimensional floating

point numbers. This can be helpful in situations where data needs to be transmitted from the wearable to a mobile phone or server, leading to a reduction in transfer costs. Furthermore, it also enables more efficient processing of extremely large-scale wearables datasets (such as the UK Biobank with 700k person days of data [94]), where the size is a crutch for analysis and model development.

Activity and Routine Discovery: As mentioned in [24], the process of discovering activities from unlabeled data is (in many ways) the opposite of building classifiers to recognize known activities using labeled data. An important application includes health monitoring where typical healthy behavior can be characterized by such discovery algorithms, whereas they may be difficult for humans (incl. experts) to fully specify [24]. One approach involves deriving ‘characteristic actions’ via motif discovery, as such sequences are statistically unlikely to occur across activities and therefore correspond to important actions within the activity [24]. Discovering motifs is easier in the discrete space (rather than raw sensor data space), especially for multi-channel data, as the simplification to a smaller alphabet aids with the identification of recurring patterns. Such a setup can be vital for understanding and analyzing human behaviors.

7. Conclusions

The primary aim of this work was to serve as a *proof of concept* to demonstrate how discrete representations can be learned from wearable sensor data, and that the performance of activity recognition systems based on such learned discretized representations is comparable to, if not better than, when using dense, i.e., continuous representations derived through state-of-the-art representation learning methods. In particular, we showed how automatically deriving the mapping between sensor data and a codebook of vectors in an unsupervised manner can solve some of the existing concerns with HAR applications based on discrete representations, including low activity recognition performance and difficulty with multi-channel data.

A deeper dive into the workings of discretization showed that explicitly controlling the maximum dictionary size can result in better representations. Further, the addition of powerful NLP-based pre-training techniques such as RoBERTa resulted in improved activity recognition for all target datasets. Therefore, this paper casts the multi-channel time-series classification problem as a discrete sequence analysis problem (similar to natural language processing), thereby facilitating the adoption of recent advancements in discrete representation learning for the field of sensor-based human activity recognition. In summary, our work offers an alternative feature extraction pipeline in sensor-based HAR, allowing for discretized abstractions of human movements and therefore enabling improved analysis of movements.

Author Contributions: Conceptualization, H.H. and T.P.; Methodology, H.H.; Software, H.H.; Validation, H.H.; Investigation, H.H. and T.P.; Resources, I.E. and T.P.; Writing—original draft, H.H.; Writing—review & editing, H.H., I.E. and T.P.; Supervision, I.E. and T.P.; Project administration, I.E. and T.P.; Funding acquisition, T.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by NSF grant IIS-2112633.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Appendix A.1. Datasets

Appendix A.1.1. Capture-24

It is a large-scale dataset recorded from a single wrist-worn accelerometer based on the Axivity AX3 platform [31–33]. It comprises free living conditions, with 151 users of data for approximately one day each, resulting in around 4000 h of data in total. Out of this,

2500 h are coarsely labeled into the six broad activities sleep, sit-stand, mixed, walking, vehicle, and bicycling. There are also 200 fine-grained activities, and the annotation was performed using a chest-mounted Vicon Autograph and Whitehall II sleep diaries. Going by the broad labels, around 75% of the data is either sleep or sit-stand, thereby rendering it imbalanced.

Appendix A.1.2. HHAR

The underlying goal for the collection of this dataset was to study the impact of heterogeneous recording devices (incl. sensors, devices, and workloads) on recognition of human activities [81]. It contains data from both mobile phones (LG Nexus 4, Samsung Galaxy S Plus, Samsung Galaxy S3, Samsung Galaxy S3 mini) and smartwatches (LG G and Samsung Galaxy Gear). We only utilize the data from the wrist watches, which were worn on each arm, as we are interested in studying the performance obtained at the wrist. Data was collected from nine users in total, who performed five minutes per activity, resulting in a balanced dataset.

Appendix A.1.3. Myogym

The Myogym [1] dataset was collected from 10 participants performing thirty different gym activities (or the NULL class), where each activity contains ten repetitions. The Myo armband on the right forearm was utilized for recording the data, and comprises an IMU containing an accelerometer, gyroscope, and magnetometer, along with eight electromyogram (EMG) sensors. Data were recorded at 50 Hz and the aim of the study was to facilitate activity and gesture recognition, as well as sensor fusion.

Appendix A.1.4. Mobiact

A Samsung Galaxy S3 smartphone placed freely in a trouser pocket was utilized to four types and twelve locomotion-style + transitional activities [82] at a rate of 200 Hz. Following [15], we removed the lying class, resulting in eleven activities from a total of 61 participants (out of 66). The sensors include accelerometer, gyroscope, and orientation, and we utilize v2 of this dataset (as in [15]).

Appendix A.1.5. Motionsense

It contains data recorded from an iPhone 6s, including accelerometer, gyroscope, and attitude information at a rate of 50 Hz [83]. A total of 24 subjects (14 men and 10 women) were recruited for data collection, with the aim of developing privacy preserving sensor data transmission systems. It mainly contains locomotion-style activities (see Table 1), collected from the front trouser pocket.

Appendix A.1.6. MHEALTH

The MHEALTH dataset [84,85] consists of data from 10 participants performing 12 activities. Shimmer 2 [101] wearable sensors were utilized for the recording, and placed on the chest, right wrist, and left ankle. The sampling rate is 50 Hz and the activities under study include locomotion along with some exercises. The collection was performed out of the laboratory, without any constraints on how they had to be executed; the subjects were asked to try their best while executing the activities.

Appendix A.1.7. PAMAP2

This dataset comprises three IMUs and a heart rate (HR) monitor, recorded to facilitate the development of physical activity monitoring systems [86]. One IMU is placed at the chest along with the HR monitor, whereas the remaining two are placed on the dominant wrist and the ankle. A total of 9 participants (8 males + 1 female) are present in the dataset and followed a protocol of 12 activities (listed in Table 1) along with (watch TV, computer work, drive car, fold laundry, clean house, and play soccer) optionally. In our study, we

only utilize the 12 that form a part of the protocol for collection. Further, we only utilize data from the ankle-worn accelerometer, as we evaluate the performance across sensor locations as well.

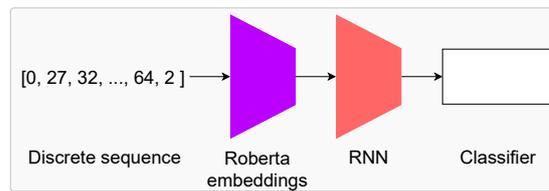


Figure A1. Using RoBERTa embeddings for classifying discrete representations: first, we pre-train a RoBERTa model on discrete representations (indexed) from Capture-24. Subsequently, we replace the learnable embeddings with frozen RoBERTa embeddings in order to study the impact of the NLP-based pre-training. A GRU network is used along with an MLP for activity recognition.

Table A1. Hyper-parameters utilized for pre-training and evaluation on the target datasets.

Dataset	Pre-Training						Evaluation	
	lr	l2 reg.	# Conv. agg.	k	# neg.	γ	lr	wd
HHAR	1×10^{-4}	1×10^{-4}	2	10	10	0.25	5×10^{-4}	1×10^{-4}
Myogym	1×10^{-4}	1×10^{-4}	2	10	10	0.25	5×10^{-4}	1×10^{-5}
Mobiact	1×10^{-4}	1×10^{-4}	2	10	10	0.25	1×10^{-3}	1×10^{-4}
Motionsense	1×10^{-4}	1×10^{-4}	2	10	10	0.25	5×10^{-4}	0.0
MHEALTH	1×10^{-4}	1×10^{-4}	2	10	10	0.25	5×10^{-4}	1×10^{-5}
PAMAP2	1×10^{-4}	1×10^{-4}	2	10	10	0.25	1×10^{-3}	1×10^{-4}

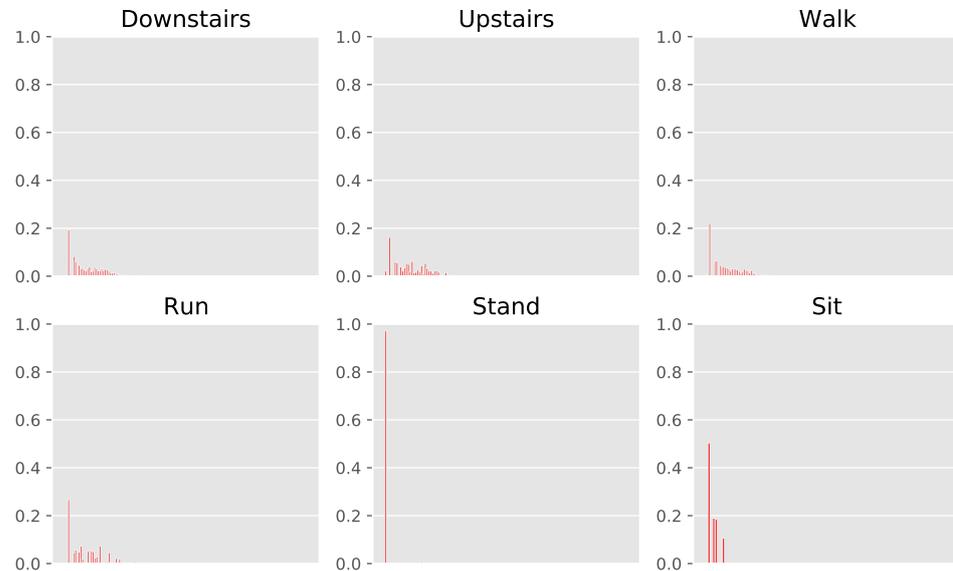


Figure A2. Visualizing the histograms of discrete representations per class for the first fold of Motionsense. This is the full figure from which Figure 5 is obtained by truncating the x -axis to 50 symbols.

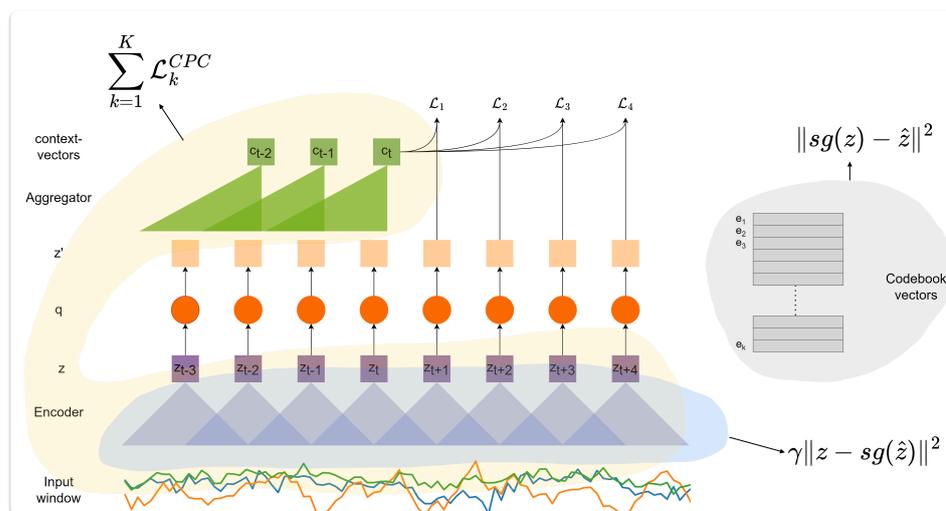


Figure A3. Visualizing how the weights of the VQ-CPC architecture are updated using different parts of the overall loss. As detailed in Section 3.1.1, the Aggregator network is updated using $\sum_{k=1}^K \mathcal{L}_k^{CPC}$ whereas the codebook vectors utilize $\|sg(z) - \hat{z}\|^2$. Finally, the encoder is updated using both $\sum_{k=1}^K \mathcal{L}_k^{CPC}$ and $\gamma \|z - sg(\hat{z})\|^2$.

References

1. Koskimäki, H.; Siirtola, P.; Röning, J. 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, Maui, HI, USA, 11–15 September 2017; pp. 537–546.
2. Bock, M.; Kuehne, H.; Van Laerhoven, K.; Moeller, M. WEAR: An Outdoor Sports for Wearable and Egocentric Activity Recognition. *arXiv* **2023**, arXiv:2304.05088.
3. Sbröllini, A.; Morettini, M.; Maranesi, E.; Marcantoni, I.; Nasim, A.; Bevilacqua, R.; Riccardi, G.R.; Burattini, L. Sport Database: Cardiorespiratory data acquired through wearable sensors while practicing sports. *Data Brief* **2019**, *27*, 104793. [[CrossRef](#)]
4. Seçkin, A.Ç.; Ateş, B.; Seçkin, M. Review on Wearable Technology in sports: Concepts, Challenges and opportunities. *Appl. Sci.* **2023**, *13*, 10399. [[CrossRef](#)]
5. Henriksen, A.; Haugen Mikalsen, M.; Woldaregay, A.Z.; Muzny, M.; Hartvigsen, G.; Hopstock, L.A.; Grimsgaard, S. Using fitness trackers and smartwatches to measure physical activity in research: Analysis of consumer wrist-worn wearables. *J. Med. Internet Res.* **2018**, *20*, e110. [[CrossRef](#)]
6. Huynh, T.; Schiele, B. Analyzing features for activity recognition. In Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies, Grenoble, France, 12–14 October 2005; pp. 159–163.
7. Bulling, A.; Blanke, U.; Schiele, B. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv. (CSUR)* **2014**, *46*, 1–33. [[CrossRef](#)]
8. Hammerla, N.Y.; Kirkham, R.; Andras, P.; Ploetz, T. On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution. In Proceedings of the 2013 International Symposium on Wearable Computers, Zurich, Switzerland, 8–12 September 2013; pp. 65–68.
9. Ordóñez, F.J.; Roggen, D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* **2016**, *16*, 115. [[CrossRef](#)] [[PubMed](#)]
10. Hammerla, N.Y.; Halloran, S.; Plötz, T. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv* **2016**, arXiv:1604.08880.
11. Guan, Y.; Plötz, T. Ensembles of deep lstm learners for activity recognition using wearables. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*; Association for Computing Machinery: New York, NY, USA, 2017; Volume 1, pp. 1–28.
12. Kim, Y.W.; Cho, W.H.; Kim, K.S.; Lee, S. Inertial-Measurement-Unit-Based Novel Human Activity Recognition Algorithm Using Conformers. *Sensors* **2022**, *22*, 3932. [[CrossRef](#)]
13. Yang, J.; Nguyen, M.N.; San, P.P.; Li, X.; Krishnaswamy, S. Deep convolutional neural networks on multichannel time series for human activity recognition. In Proceedings of the IJCAI, Buenos Aires, Argentina, 25–31 July 2015; Volume 15, pp. 3995–4001.
14. Plötz, T.; Hammerla, N.Y.; Olivier, P.L. Feature learning for activity recognition in ubiquitous computing. In Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, Catalonia, Spain, 16–22 July 2011.
15. Saeed, A.; Ozcebe, T.; Lukkien, J. Multi-task self-supervised learning for human activity detection. *Proc. Acm Interact. Mobile Wearable Ubiquitous Technol.* **2019**, *3*, 1–30. [[CrossRef](#)]

16. Haresamudram, H.; Essa, I.; Plötz, T. Assessing the State of Self-Supervised Human Activity Recognition Using Wearables. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2022**, *6*, 116. [[CrossRef](#)]
17. Stiefmeier, T.; Roggen, D.; Tröster, G. Gestures are strings: Efficient online gesture spotting and classification using string matching. In Proceedings of the 2nd International ICST Conference on Body Area Networks, Florence, Italy, 11–13 June 2007.
18. Junejo, I.N.; Al Aghbari, Z. Using SAX representation for human action recognition. *J. Vis. Commun. Image Represent.* **2012**, *23*, 853–861. [[CrossRef](#)]
19. Sousa Lima, W.; de Souza Bragança, H.L.; Montero Quispe, K.G.; Pereira Souto, E.J. Human activity recognition based on symbolic representation algorithms for inertial sensors. *Sensors* **2018**, *18*, 4045. [[CrossRef](#)] [[PubMed](#)]
20. Lin, J.; Keogh, E.; Wei, L.; Lonardi, S. Experiencing SAX: A novel symbolic representation of time series. *Data Min. Knowl. Discov.* **2007**, *15*, 107–144. [[CrossRef](#)]
21. Keogh, E.; Chu, S.; Hart, D.; Pazzani, M. An online algorithm for segmenting time series. In Proceedings of the 2001 IEEE International Conference on Data Mining, San Jose, CA, USA, 29 November–2 December 2001; IEEE: Piscataway, NJ, USA, 2001; pp. 289–296.
22. Berlin, E.; Van Laerhoven, K. Detecting leisure activities with dense motif discovery. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing, Pittsburgh, PA, USA, 5–8 September 2012; pp. 250–259.
23. Montero Quispe, K.G.; Sousa Lima, W.; Macêdo Batista, D.; Souto, E. MBOSS: A symbolic representation of human activity recognition using mobile sensors. *Sensors* **2018**, *18*, 4354. [[CrossRef](#)]
24. Minnen, D.; Starner, T.; Essa, I.; Isbell, C. Discovering characteristic actions from on-body sensor data. In Proceedings of the 2006 10th IEEE International Symposium on Wearable Computers, Montreux, Switzerland, 11–14 October 2006; IEEE: Piscataway, NJ, USA, 2006; pp. 11–18.
25. Minnen, D.; Isbell, C.L.; Essa, I.; Starner, T. Discovering multivariate motifs using subsequence density estimation and greedy mixture learning. In Proceedings of the 22nd National Conference on Artificial Intelligence, Vancouver, BC, Canada, 22–26 July 2007; AAAI Press: Menlo Park, CA, USA, 2007; MIT Press: Cambridge, MA, USA, 2007; Volume 22, p. 615.
26. Nguyen, T.L.; Gsponer, S.; Ilie, I.; Ifrim, G. Interpretable time series classification using all-subsequence learning and symbolic representations in time and frequency domains. *arXiv* **2018**, arXiv:1808.04022.
27. Baevski, A.; Schneider, S.; Auli, M. vq-wav2vec: Self-supervised learning of discrete speech representations. *arXiv* **2019**, arXiv:1910.05453.
28. Baevski, A.; Zhou, Y.; Mohamed, A.; Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 12449–12460.
29. Van Den Oord, A.; Vinyals, O. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; Volume 30.
30. Haresamudram, H.; Essa, I.; Plötz, T. Investigating enhancements to contrastive predictive coding for human activity recognition. In Proceedings of the 2023 IEEE International Conference on Pervasive Computing and Communications (PerCom), Atlanta, GA, USA, 13–17 March 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 232–241.
31. Chan Chang, S.; Doherty, A. *Capture-24: Activity Tracker Dataset for Human Activity Recognition*; University of Oxford: Oxford, UK, 2021.
32. Gershuny, J.; Harms, T.; Doherty, A.; Thomas, E.; Milton, K.; Kelly, P.; Foster, C. Testing self-report time-use diaries against objective instruments in real time. *Sociol. Methodol.* **2020**, *50*, 318–349. [[CrossRef](#)]
33. Willetts, M.; Hollowell, S.; Aslett, L.; Holmes, C.; Doherty, A. Statistical machine learning of sleep and physical activity phenotypes from sensor data in 96,220 UK Biobank participants. *Sci. Rep.* **2018**, *8*, 7961. [[CrossRef](#)]
34. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
35. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
36. Shieh, J.; Keogh, E. i SAX: Indexing and mining terabyte sized time series. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 623–631.
37. Mohamad, Y.; Nishida, T. Robust learning from demonstrations using multidimensional SAX. In Proceedings of the 2014 14th International Conference on Control, Automation and Systems (ICCAS 2014), Gyeonggi-do, Republic of Korea, 22–25 October 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 64–71.
38. Senin, P.; Malinchik, S. Sax-vsm: Interpretable time series classification using sax and vector space model. In Proceedings of the 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, 7–10 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1175–1180.
39. Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. *Data Min. Knowl. Discov.* **2015**, *29*, 1505–1530. [[CrossRef](#)]
40. Schäfer, P. Scalable time series classification. *Data Min. Knowl. Discov.* **2016**, *30*, 1273–1298. [[CrossRef](#)]
41. Schäfer, P.; Leser, U. Fast and accurate time series classification with weasel. In Proceedings of the 2017 ACM Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 637–646.
42. Ye, L.; Keogh, E. Time series shapelets: A new primitive for data mining. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 947–956.

43. Mueen, A.; Keogh, E.; Young, N. Logical-shapelets: An expressive primitive for time series classification. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011; pp. 1154–1162.
44. Yeh, C.C.M.; Zhu, Y.; Ulanova, L.; Begum, N.; Ding, Y.; Dau, H.A.; Silva, D.F.; Mueen, A.; Keogh, E. Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1317–1322.
45. Zhu, Y.; Imamura, M.; Nikovski, D.; Keogh, E. Matrix profile VII: Time series chains: A new primitive for time series data mining (best student paper award). In Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM), New Orleans, LA, USA, 18–21 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 695–704.
46. Razavi, A.; Van den Oord, A.; Vinyals, O. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2019; Volume 32.
47. Chorowski, J.; Weiss, R.J.; Bengio, S.; Van Den Oord, A. Unsupervised speech representation learning using wavenet autoencoders. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2019**, *27*, 2041–2053. [[CrossRef](#)]
48. Gumbel, E.J. *Statistical Theory of Extreme Values and Some Practical Applications: A Series of Lectures*; US Government Printing Office: Washington, DC, USA, 1954; Volume 33.
49. Jang, E.; Gu, S.; Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv* **2016**, arXiv:1611.01144.
50. Baevski, A.; Hsu, W.N.; Conneau, A.; Auli, M. Unsupervised speech recognition. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 27826–27839.
51. Chung, Y.A.; Tang, H.; Glass, J. Vector-quantized autoregressive predictive coding. *arXiv* **2020**, arXiv:2005.08392.
52. Chung, Y.A.; Zhang, Y.; Han, W.; Chiu, C.C.; Qin, J.; Pang, R.; Wu, Y. W2v-bert: Combining contrastive learning and masked language modeling for self-supervised speech pre-training. In Proceedings of the 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Cartagena, Colombia, 13–17 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 244–250.
53. Hsu, W.N.; Bolte, B.; Tsai, Y.H.H.; Lakhota, K.; Salakhutdinov, R.; Mohamed, A. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2021**, *29*, 3451–3460. [[CrossRef](#)]
54. van der Merwe, W.; Kamper, H.; Preez, J.D. A Temporal Extension of Latent Dirichlet Allocation for Unsupervised Acoustic Unit Discovery. *arXiv* **2022**, arXiv:2206.11706.
55. Kamper, H. Word Segmentation on Discovered Phone Units with Dynamic Programming and Self-Supervised Scoring. *arXiv* **2022**, arXiv:2202.11929.
56. Cuervo, S.; Łańcucki, A.; Marxer, R.; Rychlikowski, P.; Chorowski, J. Variable-rate hierarchical CPC leads to acoustic unit discovery in speech. *arXiv* **2022**, arXiv:2206.02211.
57. Dieleman, S.; Nash, C.; Engel, J.; Simonyan, K. Variable-rate discrete representation learning. *arXiv* **2021**, arXiv:2103.06089.
58. Kukačka, J.; Golkov, V.; Cremers, D. Regularization for deep learning: A taxonomy. *arXiv* **2017**, arXiv:1710.10686.
59. Santos, C.F.G.D.; Papa, J.P. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. *ACM Comput. Surv.* **2022**, *54*, 213. [[CrossRef](#)]
60. Tian, Y.; Zhang, Y. A comprehensive survey on regularization strategies in machine learning. *Inf. Fusion* **2022**, *80*, 146–166. [[CrossRef](#)]
61. Jha, S.; Schiemer, M.; Ye, J. Continual learning in human activity recognition: An empirical analysis of regularization. *arXiv* **2020**, arXiv:2007.03032.
62. Bento, N.; Rebelo, J.; Carreiro, A.V.; Ravache, F.; Barandas, M. Exploring Regularization Methods for Domain Generalization in Accelerometer-Based Human Activity Recognition. *Sensors* **2023**, *23*, 6511. [[CrossRef](#)] [[PubMed](#)]
63. Suh, S.; Rey, V.F.; Lukowicz, P. TASKED: Transformer-based Adversarial learning for human activity recognition using wearable sensors via Self-Knowledge Distillation. *Knowl.-Based Syst.* **2023**, *260*, 110143. [[CrossRef](#)]
64. Haresamudram, H.; Anderson, D.V.; Plötz, T. On the role of features in human activity recognition. In Proceedings of the 23rd International Symposium on Wearable Computers, London, UK, 9–13 September 2019; pp. 78–88.
65. Varamin, A.A.; Abbasnejad, E.; Shi, Q.; Ranasinghe, D.C.; Rezafofighi, H. Deep auto-set: A deep auto-encoder-set network for activity recognition using wearables. In Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, New York, NY, USA, 5–7 November 2018; pp. 246–253.
66. Haresamudram, H.; Beedu, A.; Agrawal, V.; Grady, P.L.; Essa, I.; Hoffman, J.; Plötz, T. Masked reconstruction based self-supervision for human activity recognition. In Proceedings of the 2020 International Symposium on Wearable Computers, Virtual, 12–16 September 2020; pp. 45–49.
67. Miao, S.; Chen, L.; Hu, R. Spatial-Temporal Masked Autoencoder for Multi-Device Wearable Human Activity Recognition. *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.* **2024**, *7*, 1–25. [[CrossRef](#)]
68. Haresamudram, H.; Essa, I.; Plötz, T. Contrastive predictive coding for human activity recognition. *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.* **2021**, *5*, 1–26. [[CrossRef](#)]
69. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. In Proceedings of the International Conference on Machine Learning—PMLR, Virtual, 13–18 July 2020; pp. 1597–1607.
70. Tang, C.I.; Perez-Pozuelo, I.; Spathis, D.; Mascolo, C. Exploring Contrastive Learning in Human Activity Recognition for Healthcare. *arXiv* **2020**, arXiv:2011.11542.

71. Ahmed, A.; Haresamudram, H.; Ploetz, T. Clustering of human activities from wearables by adopting nearest neighbors. In Proceedings of the 2022 ACM International Symposium on Wearable Computers, Cambridge, UK, 11–15 September 2022; pp. 1–5.
72. Chen, X.; He, K. Exploring simple siamese representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 15750–15758.
73. Grill, J.B.; Strub, F.; Althé, F.; Tallec, C.; Richemond, P.; Buchatskaya, E.; Doersch, C.; Avila Pires, B.; Guo, Z.; Gheshlaghi Azar, M.; et al. Bootstrap your own latent—a new approach to self-supervised learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21271–21284.
74. Qian, H.; Tian, T.; Miao, C. What Makes Good Contrastive Learning on Small-Scale Wearable-based Tasks? *arXiv* **2022**, arXiv:2202.05998.
75. Fortes Rey, V.; Suh, S.; Lukowicz, P. Learning from the Best: Contrastive Representations Learning Across Sensor Locations for Wearable Activity Recognition. In Proceedings of the 2022 ACM International Symposium on Wearable Computers, Cambridge, UK, 11–15 September 2022; pp. 28–32.
76. Jain, Y.; Tang, C.I.; Min, C.; Kawsar, F.; Mathur, A. ColloSSL: Collaborative Self-Supervised Learning for Human Activity Recognition. *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.* **2022**, *6*, 1–28. [[CrossRef](#)]
77. Deldari, S.; Xue, H.; Saeed, A.; Smith, D.V.; Salim, F.D. COCOA: Cross Modality Contrastive Learning for Sensor Data. *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.* **2022**, *6*, 1–28. [[CrossRef](#)]
78. Dhekane, S.G.; Haresamudram, H.; Thukral, M.; Plötz, T. How Much Unlabeled Data are Really Needed for Effective Self-Supervised Human Activity Recognition? In Proceedings of the 2023 ACM International Symposium on Wearable Computers, Cancún, Mexico, 8–12 October 2023; pp. 66–70.
79. Oord, A.v.d.; Li, Y.; Vinyals, O. Representation learning with contrastive predictive coding. *arXiv* **2018**, arXiv:1807.03748.
80. Bengio, Y.; Léonard, N.; Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv* **2013**, arXiv:1308.3432.
81. Stisen, A.; Blunck, H.; Bhattacharya, S.; Prentow, T.S.; Kjærgaard, M.B.; Dey, A.; Sonne, T.; Jensen, M.M. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, Seoul, Republic of Korea, 1–4 November 2015; pp. 127–140.
82. Chatzaki, C.; Pediaditis, M.; Vavoulas, G.; Tsiknakis, M. Human daily activity and fall recognition using a smartphone’s acceleration sensor. In Proceedings of the International Conference on Information and Communication Technologies for Ageing Well and e-Health, Rome, Italy, 21–22 April 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 100–118.
83. Malekzadeh, M.; Clegg, R.G.; Cavallaro, A.; Haddadi, H. Protecting sensory data against sensitive inferences. In Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems, Porto, Portugal, 23–26 April 2018; pp. 1–6.
84. Banos, O.; Garcia, R.; Holgado-Terriza, J.A.; Damas, M.; Pomares, H.; Rojas, I.; Saez, A.; Villalonga, C. mHealthDroid: A novel framework for agile development of mobile health applications. In Proceedings of the International Workshop on Ambient Assisted Living, Belfast, UK, 2–5 December 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 91–98.
85. Banos, O.; Villalonga, C.; Garcia, R.; Saez, A.; Damas, M.; Holgado-Terriza, J.A.; Lee, S.; Pomares, H.; Rojas, I. Design, implementation and validation of a novel open framework for agile development of mobile health applications. *Biomed. Eng. Online* **2015**, *14*, 1–20. [[CrossRef](#)]
86. Reiss, A.; Stricker, D. Introducing a new benchmarked dataset for activity monitoring. In Proceedings of the 2012 16th International Symposium On wearable Computers, Newcastle, UK, 18–22 June 2012; IEEE: Piscataway, NJ, USA 2012; pp. 108–109.
87. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2019; Volume 32.
88. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
89. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (accessed on 1 December 2023).
90. Lin, Z.; Liu, J.Z.; Yang, Z.; Hua, N.; Roth, D. Pruning redundant mappings in transformer models via spectral-normalized identity prior. *arXiv* **2020**, arXiv:2010.01791.
91. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv* **2019**, arXiv:1909.11942.
92. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.
93. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; Volume 30.
94. Doherty, A.; Jackson, D.; Hammerla, N.; Plötz, T.; Olivier, P.; Granat, M.H.; White, T.; Van Hees, V.T.; Trenell, M.I.; Owen, C.G.; et al. Large scale population assessment of physical activity using wrist worn accelerometers: The UK biobank study. *PLoS ONE* **2017**, *12*, e0169649. [[CrossRef](#)]
95. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *Openai Blog* **2019**, *1*, 9.
96. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.

97. Mihalcea, R.; Tarau, P. TextRank: Bringing order into text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004; pp. 404–411.
98. Ramirez-Orta, J.; Milius, E. Unsupervised document summarization using pre-trained sentence embeddings and graph centrality. In Proceedings of the Second Workshop on Scholarly Document Processing, Online, 10 June 2021; pp. 110–115.
99. Fung, P.; Ngai, G.; Cheung, C.S. Combining optimal clustering and hidden Markov models for extractive summarization. In Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering, Sapporo, Japan, 11 July 2003; pp. 21–28.
100. Gokhan, T.; Smith, P.; Lee, M. Extractive financial narrative summarisation using sentenceBERT based clustering. In Proceedings of the 3rd Financial Narrative Processing Workshop, Lancaster, UK, 15–16 September 2021; pp. 94–98.
101. Burns, A.; Greene, B.R.; McGrath, M.J.; O’Shea, T.J.; Kuris, B.; Ayer, S.M.; Stroiescu, F.; Cionca, V. SHIMMER™—A wireless sensor platform for noninvasive biomedical research. *IEEE Sens. J.* **2010**, *10*, 1527–1534. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.