TS-Rep: Self-supervised time series representation learning from robot sensor data

Pratik Somaiya
Lincoln Centre for Autonomous Systems
University of Lincoln, UK
psomaiya@lincoln.ac.uk

Harit Pandya
Toshiba Research Europe
Cambridge, UK
harit.pandya@crl.toshiba.co.uk

Riccardo Polvara
Lincoln Centre for Autonomous Systems
University of Lincoln, UK
rpolvara@lincoln.ac.uk

Marc Hanheide
Lincoln Centre for Autonomous Systems
University of Lincoln, UK
mhanheide@lincoln.ac.uk

Grzegorz Cielniak
Lincoln Centre for Autonomous Systems
University of Lincoln, UK
gcielniak@lincoln.ac.uk

Abstract

In this paper, we propose TS-Rep, a self-supervised method that learns representations from multi-modal varying-length time series sensor data from real robots. TS-Rep is based on a simple yet effective technique for triplet learning, where we randomly split the time series into two segments to form anchor and positive while selecting random subseries from the other time series in the mini-batch to construct negatives. We additionally use the nearest neighbour in the representation space to increase the diversity in the positives. For evaluation, we perform a clusterability analysis on representations of three heterogeneous robotics datasets. Then learned representations are applied for anomaly detection, and our method consistently performs well. A classifier trained on TS-Rep learned representations outperforms unsupervised methods and performs close to the fully-supervised methods for terrain classification. Furthermore, we show that TS-Rep is, on average, the fastest method to train among the baselines. Our code is available at https://github.com/imprs/TS-Rep.

1 Introduction

Time series are often sparsely labelled, making it challenging to use them in a supervised learning paradigm. Therefore, representation learning methods are employed to learn general-purpose representations that are not limited to any single supervised task. Learning a good representation of time series data is often difficult due to its complexity and high dimensionality. Furthermore, representation learning is well studied in domains like vision and language; however, it is a relatively underexplored theme in the time series domain [28], especially in the robotics context. To learn representation for anomaly detection [4, 29] and terrain classification [31] in robotics, autoencoder (AE) and variational autoencoder-based architectures have been the most common in the literature.

Recent works [14, 28, 13, 30] employ self-supervised contrastive learning to learn time series representations. When these existing methods were evaluated on multi-modal time series datasets of

3rd Workshop on Self-Supervised Learning - Theory and Practice (NeurIPS 2022).

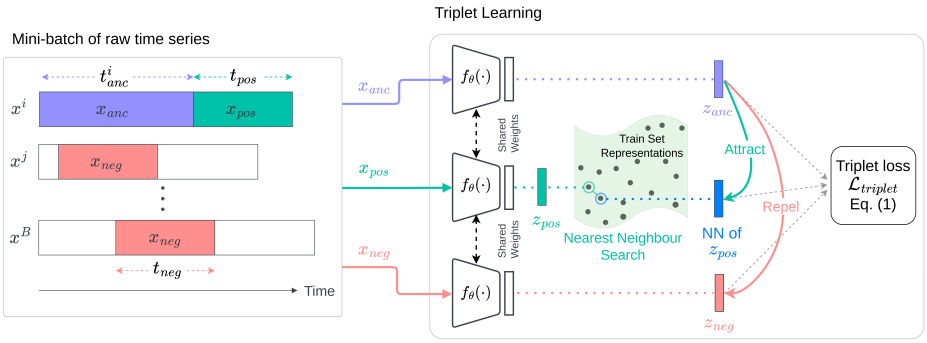

Figure 1: TS-Rep overview: (left) Anchor x_{anc}, Positive x_{pos} and Negatives x_{neg} sampling from a mini-batch of time series. (right) Triplet loss brings anchor representation z_{anc} closer to the nearest neighbour of positive representation z_{pos} while pushing apart negative representations z_{neg}.

varied sizes collected from a variety of real robots and scenarios, they struggled to learn effective representations, as we show later in experiments. Our proposed approach, TS-Rep, not only learns effective representations through a simple technique to sample positives and negatives in its triplet loss objective but also reduces the training time significantly. To support our claims, we conduct extensive experiments through clusterability analysis on three different datasets, including a mobile robot and manipulator and compare TS-Rep with recent self-supervised and AE-based approaches. Furthermore, we showcase the effectiveness of the learned representation by applying them to downstream tasks of anomalous instance detection and terrain classification, where our approach outperforms unsupervised methods and is at par with even supervised approaches.

2 Methodology

Given the time series data obtained from the robot's sensors, our approach aims at learning a representation in a self-supervised fashion. We represent the dataset of N given time series as $X = \{x^i\}_{i=1}^{N}$. Each time series is F-dimensional and is written as $x^i \in \mathbb{R}^{t^i \times F}$, where t^i is the length of the i^{th} time series. The time series are univariate when $F = 1$ and multivariate when $F > 1$. We sample a mini-batch of time series $\mathcal{B}_x$ from dataset X. For each sample x^i in $\mathcal{B}_x$, we obtain the anchor (x_{anc}) and the positive (x_{pos}) subseries through randomised cropping of the original time series, and we sample random segments from all the remaining time series in the mini-batch as negatives (x_{neg}) (refer to section 2.2). These subseries (i.e., x_{anc}, x_{pos} and x_{neg}) are propagated through the network (refer to section 2.1) for obtaining the hidden representations (z_{anc}, z_{pos} and z_{neg}), and the mini-batch of these representations is denoted by $\mathcal{B}_z$. Triplet loss (refer to section 2.3) brings the representation of anchor (z_{anc}) and positive (z_{pos}) closer while pushing apart representations of negatives (z_{neg}). We further use the nearest neighbour in the representation space for positive sampling (refer to section 2.2). An overview of the proposed approach is presented in Fig. 1.

2.1 Network Architecture

Our network is identical to T-Loss [14], and further details about the network are given in Appendix D. The network outputs a mini-batch of representations $\mathcal{B}_z \in \mathbb{R}^{B \times L}$ for anchor, positive and negative subseries. Here, B is the mini-batch size and $z^i \in \mathcal{B}_z$ is an L dimensional vector.

2.2 Positive and Negative Selection for Triplet Learning

For any time series x^i, T-Loss [14] selects positive subseries x^i_{pos} such that the positive subseries itself is part of the anchor subseries, i.e., $x^i_{pos} \in x^i_{anc}$. Instead, in TS-Rep, we randomly crop the time series into two non-overlapping and temporally adjacent subseries to form the anchor and positive. If the length of the i^{th} time series in the mini-batch is t^i and the length of the shortest time series is t^{min}, then we first randomly sample the length of the positives (t_{pos}; $t_{pos} < t^{min}$). We keep t_{pos} the same for all time series in the mini-batch. The length of the anchor t^i_{anc} is the remaining length of

the time series, i.e., $t^i_{anc} = t^i - t_{pos}$. This sampling technique allows TS-Rep to learn temporally invariant representations and it is illustrated in Fig. 1.

In T-Loss, K negatives are sampled from the train set for each time series in the mini-batch of size B, and thus, the total number of negative representations required to be computed is KB. This computation makes T-Loss comparatively slower to train than more recent methods such as [30, 13]. In TS-Rep, we tackle this by using the remaining time series from the mini-batch for negatives which only requires computing B representations while improving the training time considerably, as shown empirically in our results.

In addition to using a temporally adjacent subseries as a positive, we further leverage the diversity in the training data and sample positive from the nearest neighbour of z_{pos}, as illustrated in Fig. 1. This idea of using the nearest neighbour has been studied before in visual and neural activity representation learning [12, 3], and our approach is in a similar direction. To further randomise the selection process of a positive, we sample from a Bernoulli distribution. If the outcome is 1, we use the nearest neighbour of z_{pos} as a positive, else we use the z_{pos} (representation of the subseries) itself. To obtain the nearest neighbour, we use the train set representations S (see Fig. 1) from the previous epoch.

2.3 Triplet loss

Our triplet loss formulation is based on word2vec [19] and T-Loss [14]. Given representations of anchor (z_{anc}), positive (z_{pos}) and negatives (z_{neg}), our training objective for a mini-batch is:

$$\mathcal{L}_{triplet} = \frac{1}{B}\sum_{i=1}^{B}\left[-\log\Big(\sigma\Big((z^i_{anc})^T\ NN(z^i_{pos}, S)\Big)\Big) - \sum_{j=1;j\neq i}^{B} log\Big(\sigma\Big(-(z^i_{anc})^T(z^j_{neg})\Big)\Big)\right] \quad (1)$$

where σ is the sigmoid function, B is the mini-batch size, and $NN()$ returns the nearest neighbour of z^i_{pos} from train set representations S.

3 Experiments

In this section, we present the experiments to evaluate the quality of the learned representations through clusterability analysis, anomaly detection and terrain classification. We refer our readers to Appendix C and Appendix E for details about the datasets and evaluation metrics, respectively.

3.1 Clusterability Analysis of Learned Representations

We first assess the quality of the learned representations through clusterability analysis, as in [28]. Real-world time series data in robotics often have a multi-categorical underlying structure, which leads to clustering properties in low-dimensional representation. Learning such clustering properties is an indicator of a well-learned representation [28, 8].

We compare TS-Rep with the self-supervised methods [14, 13, 30], and a VAE-based method Incr-VAE [4]. We use Normalized Mutual Information (NMI) as in [20], and Silhouette score and Davies-Bouldin index (DBI) following TNC [28].

Table 1: Comparison of clustering results on train set representations.

	Training	Manipulation			Water Monitoring Robot			QCAT-6		
	time (hrs)	NMI ↑	Silhouette ↑	DBI ↓	NMI ↑	Silhouette ↑	DBI ↓	NMI ↑	Silhouette ↑	DBI ↓
Incr-VAE [4]	0.86	**0.70±0.05**	0.26±0.04	1.39±0.15	0.19±0.17	0.22±0.05	1.44±0.13	0.65±0.05	0.16±0.01	2.14±0.12
T-Loss [14]	4.83	0.45±0.11	0.11±0.02	2.34±0.15	0.84±0.03	0.25±0.03	1.72±0.14	0.72±0.06	0.11±0.01	2.75±0.15
TS-TCC [13]	3.09	0.20±0.05	0.17±0.06	1.70±0.15	0.97±0.04	0.67±0.04	0.83±0.12	0.65±0.06	0.15±0.01	2.01±0.06
TS2Vec [30]	0.72	0.34±0.09	0.11±0.02	2.14±0.11	0.80±0.06	0.11±0.01	2.27±0.10	0.48±0.07	0.07±0.01	2.75±0.09
TS-Rep	**0.64**	0.65±0.06	**0.31±0.05**	**1.26±0.22**	**1.00±0.00**	**0.70±0.04**	**0.50±0.07**	**0.80±0.02**	**0.41±0.02**	**1.10±0.06**

We present our results in Table 1, which shows that our method TS-Rep outperforms all the baselines in all three metrics, except in Manipulation on NMI, where Incr-VAE achieves better results. Incr-VAE

also performs well on QCAT-6 dataset but achieves the lowest performance on the Water Monitoring Robot (WMR) dataset. After TS-Rep, T-Loss performs consistently on all datasets. The results of TS-TCC are close to TS-Rep on WMR, but TS-TCC still falls behind on the other datasets. TS2Vec achieves the lowest scores on QCAT-6 and falls behind TS-Rep and T-Loss on Manipulation and WMR. These results indicate that TS-Rep is able to learn the underlying patterns in the data and group instances with similar characteristics. The baselines do not achieve as competitive performance as TS-Rep and do not perform consistently well across different datasets. Qualitative results and more details on training time are provided in Appendix F.2 and Appendix F.1, respectively.

3.2 Anomaly Detection

Anomaly detection refers to the identification of anomalous instances in the data, which is an important application in robotics. We train a one-class SVM (OCSVM) classifier on the nominal class representations for instance-wise anomaly detection. OCSVM has been previously used for anomaly detection tasks on learned features [29, 25] and on raw features [26].

We assess anomaly detection performance with F1-score, area under a receiver operating characteristic (AUROC) and False-positive rate (FPR) at 95% True-positive rate (TPR) following [29, 22, 10, 15].

Table 2: Comparison of anomaly detection results on Test set.

	Manipulation			Water Monitoring Robot			QCAT-6		
	F1 ↑	AUROC ↑	FPR-95%-TPR ↓	F1 ↑	AUROC ↑	FPR-95%-TPR ↓	F1 ↑	AUROC ↑	FPR-95%-TPR ↓
Incr-VAE [4]	**0.83±0.25**	**0.99±0.02**	**0.02±0.06**	0.33±0.41	0.80±0.13	0.56±0.24	0.99±0.01	1.00±0.00	0.01±0.02
T-Loss [14]	0.77±0.12	0.90±0.06	0.37±0.21	1.00±0.00	1.00±0.00	0.00±0.00	**1.00±0.00**	1.00±0.00	**0.00±0.00**
TS-TCC [13]	0.39±0.05	0.58±0.05	0.69±0.15	1.00±0.00	1.00±0.00	0.00±0.00	0.92±0.09	0.97±0.04	0.10±0.08
TS2Vec [30]	0.76±0.03	0.75±0.04	0.50±0.13	0.99±0.01	1.00±0.00	0.00±0.00	0.96±0.01	0.98±0.01	0.09±0.06
TS-Rep	0.81±0.05	0.85±0.05	0.33±0.05	1.00±0.00	1.00±0.00	0.00±0.00	0.98±0.02	1.00±0.00	0.01±0.03

As we see in Table 2, there is no clear winner, but T-Loss and TS-Rep perform consistently well across all three datasets. Incr-VAE achieves the best results in all three metrics on the Manipulation dataset and achieves near-perfect scores on QCAT-6; however, it performs poorly on the Water Monitoring Robot (WMR) dataset. Incr-VAE struggles to separate all the classes in the WMR dataset, as shown by the clustering results in section 3.1, and that possibly explains Incr-VAE's poor performance in anomaly detection. TS-TCC and TS2Vec achieve perfect scores in the WMR dataset but fall behind in the other two datasets. These consistent results show that TS-Rep learned representations are useful for downstream tasks such as anomaly detection across different datasets.

3.3 Terrain Classification

Terrain classification is the task of identifying the type of terrain that the robot is traversing based on the sensor feedback. We utilise two terrain classification datasets, PUTany [7] and QCAT [1]. Following [14, 30], we fit an SVM classifier on the learned representations for classification.

On the QCAT dataset, existing baselines use 10-Fold Cross-Validation for evaluation and use classification accuracy for assessment. We evaluate against two fully-supervised methods, namely RNN-FCL [1] and HAPTR2 [6].

Table 3: Comparison of Terrain classification accuracy on the QCAT with 10-fold Cross-Validation.

		Force	Force + IMU
		Acc (%)	Acc (%)
Supervised	HAPTR2[6]	**97.33±1.21**	-
	RNN-FCL[1]	96.60±0.89	97.64±1.03
Self-supervised	TS-Rep	94.14±0.47	**98.31±0.33**

When evaluating with only force readings, TS-Rep achieves lower accuracy compared to supervised methods RNN-FCL and HAPTR2. When considering both force and Inertial Measurement Unit (IMU) sensors, TS-Rep outperforms the supervised method RNN-FCL, as shown in Table 3. TS-Rep also reports lower standard deviations and is a little more robust. We present the results of the PUTany dataset in Appendix F.3.

We provide ablation studies in Appendix G and implementation details in Appendix H.

4 Conclusion

We presented a self-supervised method for learning generalised representations from time series data. Our approach, TS-Rep, uses a triplet loss objective and crops time series into two temporally adjacent subseries to create an anchor-positive pair and further uses the nearest neighbour in the representation space to increase the variety in positives. TS-Rep employs negatives from the mini-batch itself, which substantially improves the training time compared to T-Loss. Our extensive evaluation through clusterability analysis, anomaly detection and terrain classification shows the effectiveness of TS-Rep learned representations for downstream applications. Although the focus of this paper was time series data from robotics, TS-Rep can also be used for time series from other domains, e.g., healthcare, energy, etc.

Acknowledgments

We would like to thank the reviewers for their constructive comments. This work has been supported by the European Commission as part of H2020 under grant number 871704 (BACCHUS) and UK Research and Innovation (InnovateUK) under the grant number 51367 (Robot Highways).

References

[1] Ahmadreza Ahmadi, Tønnes Nygaard, Navinda Kottege, David Howard, and Nicolas Hudson. Semi-supervised gated recurrent neural networks for robotic terrain classification. *IEEE Robotics and Automation Letters*, 6(2):1848–1855, 2021. 4, 9, 14

[2] Shahin Amiriparian, Michael J. Freitag, Nicholas Cummins, and Björn Schuller. Sequence to sequence autoencoders for unsupervised representation learning from audio. In *DCASE*, 2017. 8

[3] Mehdi Azabou, Mohammad Gheshlaghi Azar, Ran Liu, Chi-Heng Lin, Erik C Johnson, Kiran Bhaskaran-Nair, Max Dabagia, Bernardo Avila-Pires, Lindsey Kitchell, Keith B Hengen, et al. Mine your own view: Self-supervised learning through across-sample prediction. *arXiv preprint arXiv:2102.10106*, 2021. 3

[4] Davide Azzalini, Luca Bonali, and Francesco Amigoni. A minimally supervised approach based on variational autoencoders for anomaly detection in autonomous robots. *IEEE Robotics and Automation Letters*, 6(2):2985–2992, 2021. 1, 3, 4, 8, 9, 10, 13

[5] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018. 10, 14

[6] Michał Bednarek, Michał R Nowicki, and Krzysztof Walas. Haptr2: Improved haptic transformer for legged robots' terrain classification. *Robotics and Autonomous Systems*, page 104236, 2022. 4, 14

[7] Michał Bednarek, Mikołaj Łysakowski, Jakub Bednarek, Michał R. Nowicki, and Krzysztof Walas. Fast haptic terrain classification for legged robots using transformer. In *2021 European Conference on Mobile Robots (ECMR)*, pages 1–7, 2021. 4, 9, 14

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 3

[9] Russell Buchanan, Jakub Bednarek, Marco Camurri, Michał R Nowicki, Krzysztof Walas, and Maurice Fallon. Navigating by touch: haptic monte carlo localization via geometric sensing and terrain classification. *Autonomous Robots*, 45(6):843–857, 2021. 14

[10] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems*, 32, 2019. 4

[11] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020. 14

[12] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597, 2021. 3

[13] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2352–2359. International Joint Conferences on Artificial Intelligence Organization, 8 2021. 1, 3, 4, 8, 10, 13, 15

[14] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 1, 2, 3, 4, 8, 10, 13, 15

[15] Tianchen Ji, Arun Narenthiran Sivakumar, Girish Chowdhary, and Katherine Driggs-Campbell. Proactive anomaly detection for robot navigation with multi-sensor fusion. *IEEE Robotics and Automation Letters*, 7(2):4975–4982, 2022. 4

[16] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020. 8

[17] Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592, 2015. 14

[18] Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Timenet: Pre-trained deep recurrent neural network for time series classification, 2017. 8

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. 3

[20] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6:39501–39514, 2018. 3

[21] Tønnes F. Nygaard, Jørgen Nordmoen, Kai Olav Ellefsen, Charles P. Martin, Jim Tørresen, and Kyrre Glette. Experiences from real-world evolution with dyret: Dynamic robot for embodied testing. In Kerstin Bach and Massimiliano Ruocco, editors, *Nordic Artificial Intelligence Research and Development*, pages 58–68, Cham, 2019. Springer International Publishing. 9

[22] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018. 4

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 15

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 15

[25] Kihyuk Sohn, Chun-Liang Li, Jinsung Yoon, Minho Jin, and Tomas Pfister. Learning and evaluating representations for deep one-class classification. *arXiv preprint arXiv:2011.02578*, 2020. 4

[26] Pratik Somaiya, Marc Hanheide, and Grzegorz Cielniak. Unsupervised anomaly detection for safe robot operations. In *UKRAS20 Conference: "Robots Into The Real World" Proceedings*, pages 154–156. UKRAS, 2020. 4

[27] Santosh Thoduka, Juergen Gall, and Paul G. Plöger. Using visual anomaly detection for task execution monitoring. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4604–4610, 2021. 9

[28] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. In *International Conference on Learning Representations*, 2021. 1, 3, 8

[29] Julian Wiederer, Arij Bouazizi, Marco Troina, Ulrich Kressel, and Vasileios Belagiannis. Anomaly detection in multi-agent trajectories for automated driving. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1223–1233. PMLR, 08–11 Nov 2022. 1, 4, 8

[30] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series, 2021. 1, 3, 4, 8, 10, 13, 15

[31] Mikołaj Łysakowski, Michał R. Nowicki, Russell Buchanan, Marco Camurri, Maurice Fallon, and Krzysztof Walas. Unsupervised learning of terrain representations for haptic monte carlo localization. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4642–4648, 2022. 1, 8, 14

A Related Work

Autoencoder (AE) based methods have been widely used to learn time series representation by reconstructing the input series [18, 2]. In robotics, recently, Azzalini et al. [4] proposed Incr-VAE, a two-stage approach based on Variational Autoencoder (VAE) for anomaly detection in robotics, where the first stage is representation learning and the second stage is anomaly detection. Wiederer et al. [29] combined Graph Neural Networks (GNNs) with AE and presented a Spatio-temporal graph autoencoder (STGAE) to detect abnormal driving behaviour in a multi-agent system. Similar to Incr-VAE, STGAE first learns the low-dimensional representation and then performs anomalous behaviour detection. Łysakowski et al. [31] introduced an improved autoencoder (IAE) based method for representation learning and haptic localisation from force/ torque sensor data in a quadruped robot.

In recent years, self-supervised contrastive learning-based methods have gained attention for generalised time series representation learning [28, 13, 14, 30]. Franceschi et al. [14] presented T-Loss, a time series representation learning method based on triplet loss and time-based negative sampling. T-Loss suggested that using the encoder alone significantly reduces the computation cost while learning superior representations. In TS-TCC [13], the authors proposed the cross-view prediction technique for temporal contrasting and an additional contextual contrasting for learning discriminative representations. TNC [28] introduced an unsupervised representation learning method for non-stationary time series and presented the concept of temporal neighbourhood. Another recent work, called TS2Vec [30], proposed a universal time series representation learning framework, where they applied temporal-level and instance-level contrastive loss hierarchically at different time scales to learn a fine-grained representation of time series. TS2Vec also obtained state-of-the-art performance on downstream tasks and outperformed the unsupervised methods for time series forecasting, classification and anomaly detection. In the context of robotics, CuRL [16] applies instance contrastive loss on images and shows significant improvement in reinforcement learning tasks.

B Visulisation: Positive and Negative sampling

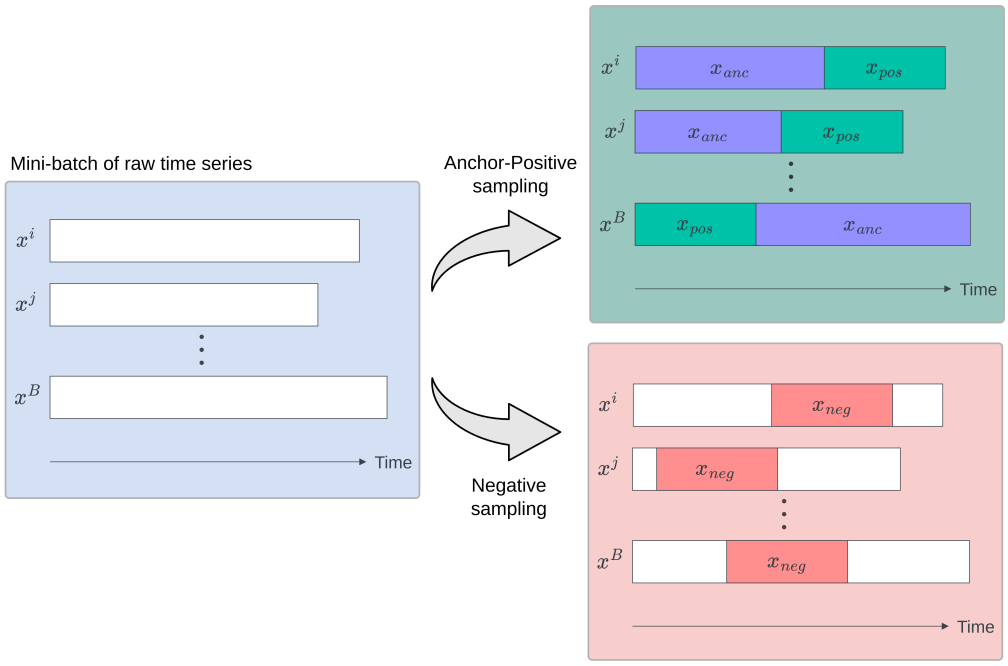

Figure 2: Anchor x_{anc}, Positive x_{pos} and Negatives x_{neg} sampling from a mini-batch of size B.

C Datasets

We validate our proposed method against state-of-the-art approaches on four different publicly available robotic datasets: (i) the Manipulation dataset [27], collected during a manipulative task of a robot placing books in a shelf, (ii) the Water Monitoring Robot dataset [4], representing runs performed by an unmanned surface vehicle to collect water samples, (iii) the QCAT dataset [1], recorded with quadruped robot DyRET [21] walking over different type of terrains, and (iv) the PUTany dataset [7], was recorded with the ANYmal robot walking over samples from different real-world terrains. We are now going to describe each individual dataset in detail.

Table 4: Datasets used in this study and their properties.

Dataset	Properties				Data Splits	
	Samples	Max len	Features	Classes	Train	Test
Manipulation [27]	121	640	84	5	100	21
Water Monitoring Robot [4]	1000	440	6	9	800	200
QCAT [1]	2880	662	22	6	2304	576
QCAT-6 [1]	480	400	22	6	384	96
PUTany [7]	5739	160	6	8	3443	1147

C.1 Manipulation

The manipulation dataset [27] was originally developed for visual anomaly detection for a mobile manipulator placing a book on a shelf. It contains runs of varied time lengths and each run includes joint position, velocity, and acceleration for all the 7 joints in joint space, and signals from the force-torque sensor mounted on the end-effector of the manipulator, which results in 84 features in total. The dataset originally presents annotations for anomalous behaviours such as books falling down, the robot being disturbed by collision, etc; however, these events and annotations are at a timestep level. In our approach, we require the whole time series to have a single label and so we reannotated the dataset based on the shelf the robot is placing the book onto.

C.2 Water Monitoring Robot

The water monitoring dataset [4] describes a water drone performing a coverage and sampling task, and it has been collected as part of the H2020 EU project INTCATCH. Each run in the dataset is described by 6 features: latitude, longitude, sine of heading, cosine of heading, and power signals to the left and right motors. A visualisation of the dataset classes is presented in Fig. 3.

C.3 QCAT

The QCAT dataset [1] was recorded during walking sessions of the DyRET [21] quadruped robot on different types of terrains. The dataset includes measurements from force-torque sensors mounted on all four feet of the robot, and the IMU sensor mounted on the body. The walking sessions include DyRET traversing over 6 different types of terrains at 6 different speeds. The terrains include grass, concrete, gravel, dirt, mulch, and sand. We additionally create QCAT-6 datasets from QCAT by using data from only one speed, which results in 480 samples.

C.4 PUTany

The PUTany dataset [7] was collected by the ANYmal quadruped robot traversing on 8 types of terrain samples, namely carpet, artificial grass, sand, rocks, rubber, ceramic tiles, foam, and PVC. Each sample consists of force/ torque feedback from the feet of the robot. For a fair comparison, we follow baselines in section F.3 to split the dataset and use 3443 samples for training the network and keep 1147 in the test. We do not use the remaining 1149 in this case.

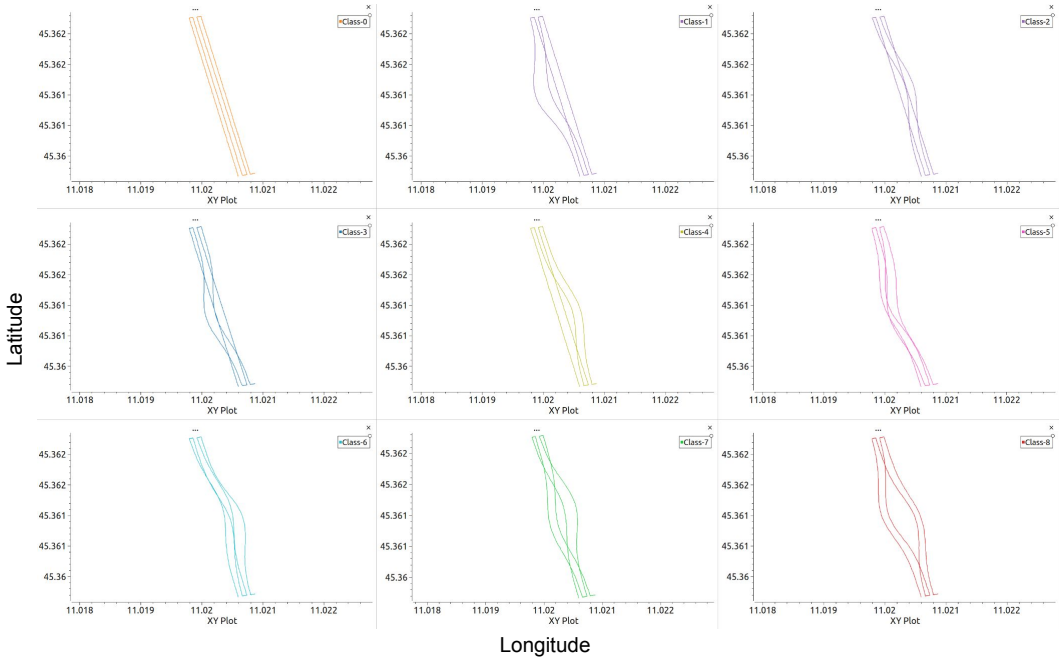

Figure 3: Visualisation of the Water Monitoring Robot dataset with Longitude and Latitude features.

D Choice of Encoder

Our network consists of stacked causal dilated convolutional layers, weight normalisation, Leaky ReLU, and residual connections. The output from these stacked layers is first passed through the global pooling layer to squeeze the time dimension, and then through the linear layer to apply the linear transformation.

Previous studies in [14, 30] show that the dilated CNN-based networks, such as the temporal convolutional network (TCN) [5], are better at capturing the long-term dependencies in time series than the Recurrent Networks or Transformer based networks. In our experiments, we observed that the T-Loss [14] learns better representations (i.e., clusterable) than the other self-supervised and VAE-based methods such as [30, 13, 4] and so, we kept the network architecture similar to T-Loss [14].

E Evaluation Metrics

E.1 Clusterability

To evaluate the clustering performance on the learned representation, we use three commonly used metrics, namely the Silhouette score and Davies-Bouldin index (DBI) and Normalized Mutual Information (NMI).

Normalized Mutual Information (NMI): Normalized Mutual Information (NMI) measures the agreement between the true and predicted cluster assignments. The NMI is bounded between 0 and 1, and the higher number represents a better agreement between the true and predicted cluster assignments. NMI is given by:

$$NMI(y_{\text{true}}, y_{\text{pred}}) = \frac{MI(y_{\text{true}}, y_{\text{pred}})}{\frac{1}{2}[H(y_{\text{true}}) + H(y_{\text{pred}})]} \tag{2}$$

where y_{true} is ground-truth labels, y_{pred} is predicted labels, MI is mutual information metric and H is entropy.

Silhouette score: The Silhouette score measures how similar an instance is to the other instances within the same cluster compared to other clusters. The Silhouette score is between -1 and 1, and the higher number indicates the sample is well-matched with its own cluster and poorly matches the other clusters. The Silhouette score for any sample i is given as:

$$\text{Silhouette score } s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}, \tag{3}$$

where $a(i)$ is the mean distance between the i^{th} sample and all other samples in the same cluster and is given by:

$$\text{mean intra-cluster distance } a(i) = \frac{1}{|C_k| - 1} \sum_{j \in C_k, i \neq j} d(i, j) \tag{4}$$

where C_k is cluster k, $|C_k|$ represents the number of samples in the cluster k and $d(i, j)$ denotes distance between the sample i and j.

$b(i)$ is the mean distance between the i^{th} sample and all other samples in the nearest or neighbouring cluster and is written as:

$$\text{mean nearest-cluster distance } b(i) = \min_{l \neq k} \frac{1}{|C_l|} \sum_{j \in C_l} d(i, j) \tag{5}$$

Davies-Bouldin index (DBI): The Davies-Bouldin index or DBI measures how separated all clusters are and how close samples are within the cluster. The lower number for DBI indicates low intra-cluster and high inter-cluster distances. DBI is given as,

$$DBI \equiv \frac{1}{K} \sum_{i=1}^{K} \max_{i \neq j} R_{ij} \tag{6}$$

where K is the number of clusters and R_{ij} is defined as,

$$R_{ij} = \frac{s_i + s_j}{m_{ij}} \tag{7}$$

In Equation 7, s is the average distance of all points in the cluster to the cluster centre, and subscript i and j denote the cluster number. Here, s_i and s_j measure the cluster scatter in cluster i and j, respectively. m_{ij} measures the separation between the cluster i and j and can be calculated by measuring the distance between the cluster centroids.

E.2 Anomaly Detection

To calculate F1-score, we learn Gaussian distribution over raw scores of the nominal time series obtained using the OCSVM and log/ store the mean and standard deviation. During inference, we check if the predicted score is within the three standard deviations of the training scores. If it is inside the range, we predict it as nominal, and if out of range, we predict it as anomalous.

We additionally use AUROC and FPR at 95% TPR, which are directly calculated with the raw prediction scores.

We first explain True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) before going into evaluation metrics.

True Positive (TP): actual positive (anomalous) that is classified correctly as a positive (anomalous).

False Positive (FP): actual negative (nominal) that is misclassified as a positive (anomalous).

True Negative (TN): actual negative (nominal) that is correctly classified as a negative (nominal).

False Negative (FN): actual positive (anomalous) that is misclassified as a negative (nominal).

F1-score: F1-score is often used to evaluate a classifier's performance. It combines precision and recall into a single metric and is the harmonic mean of precision and recall. The range for the F1-score is 0 to 1. F1-score is defined as,

$$\text{F1-score} = 2\,\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{8}$$

and precision and recall are given by,

$$\text{Precision} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}} \tag{9}$$

$$\text{Recall} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} \tag{10}$$

where #TP and #FP are the number of true and false positives, respectively, and #FN is the number of false negatives.

Area under the receiver operating characteristic (AUROC): AUROC is obtained by calculating the area under the receiver operating characteristic (ROC) curve. The ROC is a probability curve obtained by calculating the false positive rate (FPR) and true positive rate (TPR). Here, FPR and TPR are calculated with different classification threshold values, and a graphical plot of TPR vs FPR at these thresholds is called the ROC curve. The area under this ROC curve is measured to evaluate the classifier's performance and is referred to as AUROC. It ranges from 0 to 1. FPR and TPR are given by,

$$\text{False positive rate (FPR)} = 1 - \frac{\#\text{TN}}{\#\text{TN} + \#\text{FP}} \tag{11}$$

$$= \frac{\#\text{FP}}{\#\text{TN} + \#\text{FP}} \tag{12}$$

$$\text{True positive rate (TPR)} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} \tag{13}$$

where #TN is the number of true negatives.

TPR is also known as sensitivity, and FPR is known as $1-$ specificity, where specificity shows the proportion of correctly classified negatives.

FPR at 95% TPR: FPR at 95% TPR measures the probability of misclassifying a negative (normal data) as positive (anomalous) when the TPR is as high as 95%. The range for FPR at 95% TPR is from 0 to 1.

E.3 Terrain Classification

We follow the baselines and use the classification accuracy to measure the performance. Accuracy is defined as,

$$\text{Accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}} \tag{14}$$

Mathematically, it is written as,

$$\text{Accuracy}(y_{\text{true}}, y_{\text{pred}}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(y^i_{\text{true}}, y^i_{\text{pred}}) \tag{15}$$

In Equation 15, $\mathbb{1}$ is an indicator function that returns 1 when y^i_{true} is the same as y^i_{pred}, and 0 otherwise.

Accuracy is reported in percentage ($\%$) and is in the range of 0 to 100%.

F Additional Results

F.1 Training Time

We report training time for each method on all three datasets in Table 5, and it shows that, on average, TS-Rep is the fastest method to train among the baselines.

Table 5: Training time (in hours) for Manipulation, Water Monitoring Robot, and QCAT-6 datasets.

	Manipulation	Water Monitoring Robot	QCAT-6	Total
	Training time (hrs)	Training time (hrs)	Training time (hrs)	Training time (hrs)
Incr-VAE[4]	0.14	0.50	**0.22**	0.86
T-Loss[14]	0.40	2.28	2.14	4.83
TS-TCC[13]	1.03	1.03	1.03	3.09
TS2Vec[30]	0.12	0.31	0.29	0.72
TS-Rep	**0.05**	**0.29**	0.30	**0.64**

F.2 Clusterabilty Analysis

A qualitative evaluation of representations generated by TS-Rep and baselines on Manipulation, Water Monitoring Robot and QCAT-6 is given in Fig. 4.

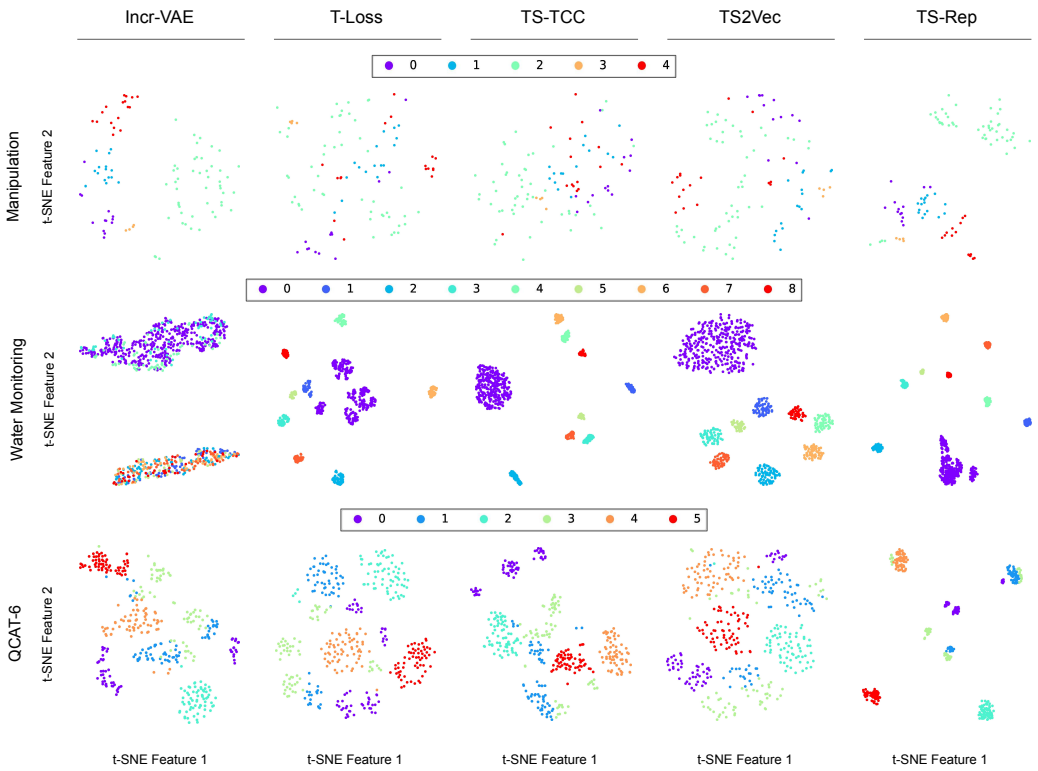

Figure 4: Qualitative analysis of the representation obtained by our method TS-Rep and the other baselines using t-SNE projection, for the manipulation dataset (top), the water monitoring robot dataset (center), and the QCAT-6 (bottom).

The t-SNE projection of the Manipulation dataset with shelves is presented in Fig. 5.

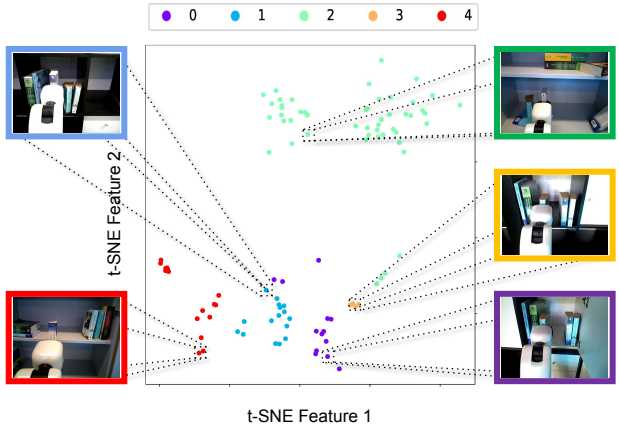

Figure 5: Learned representations on the manipulation dataset by TS-Rep. Here, the manipulator is placing books on different shelves, denoted by 0 to 4 in different colors. Our approach is able to separate different classes from the robot's sensory data in a self-supervised fashion without any labels.

F.3 Terrain Classification

We evaluate TS-Rep on the PUTany dataset through 10-Fold CV and on a hold-out test set as in the existing methods. For 10-Fold CV we use the same baselines as QCAT as in section 3.3. For test set evaluation, we compare our method against two non-deep learning methods DTW-KNN [17] and ROCKET [11], supervised methods TCN [5], HAPTR [7], HAPTR2 [6] and CNN-RNN [9] and an unsupervised method IAE [31].

We present results on the test dataset in Table 6. Our method achieves the accuracy of 87.4% and performs better than the non-deep learning methods DTW-KNN (74.0%) and ROCKET (84.9%), unsupervised method IAE (74.2%) and performs close to the supervised method TCN (87.5%). However, TS-Rep is still behind the other supervised methods HAPTR (91.7%), HAPTR2 (92.7%) and CNN-RNN (93.0%). Similar to TS-Rep, IAE is also a two-stage learning method, namely the representation learning stage and the classification stage. For a better comparison and to study the effect of output dimension L, we increase the L to 512 as in IAE. We achieve 90.2% accuracy, which is a $+2.8\%$ gain from the previous result with $L = 100$.

Table 6: Comparison of Terrain classification accuracy on the PUTany Test dataset.

		Acc (%)
Supervised	DTW-KNN [17]	74.0
	ROCKET [11]	84.9
	TCN [5]	87.5
	HAPTR [7]	91.7
	HAPTR2 [6]	92.7
	CNN-RNN [9]	**93.0**
Unsupervised/ Self-supervised	IAE [31] (L=512)	74.2
	TS-Rep (L=100)	87.4
	TS-Rep (L=512)	90.2

Table 7: Comparison of Terrain classification accuracy on the PUTany dataset with 10-Fold Cross-Validation.

		Acc (%)
Supervised	HAPTR2[6]	**93.85±0.82**
	RNN-FCL[1]	93.20±0.89
Self-supervised	TS-Rep (L=100)	89.49±0.49
	TS-Rep (L=512)	92.01±0.41

We present 10-Fold CV results in Table 7, and TS-Rep (89.49 ± 0.49) achieves accuracy close to the supervised methods HAPTR2 (93.85 ± 0.82) and RNN-FCL (93.20 ± 0.89). Again, when $L = 512$, the performance improves from 89.49 ± 0.49 to 92.01 ± 0.41.

G Ablation Studies

We additionally perform studies to understand the importance of different components in TS-Rep. As shown in Table 8, in our experiments, we first change the positive selection method from non-overlapping subseries to overlapping subseries, similar to T-Loss, and we see a drop in performance in all three datasets in all three metrics. In the second case, we randomly sample negatives from the train set similar to T-Loss, as opposed to batch negatives in TS-Rep, and the results are very close to the TS-Rep results except the training time increased significantly (roughly 7 times). Third, we remove the nearest neighbour strategy, and we see notice the drop in performance in all three datasets, which validates our intuition that the nearest neighbours provide a more diverse set of positives. In the end, we perform experiments without any padding or resampling and use the varying length time series directly, and we see that TS-Rep has gained in NMI ($+0.03$) in Manipulation and Silhouette score ($+0.03$) in WMR but a loss in Silhouette score (-0.01) in Manipulation and overall training time increases. This suggests that TS-Rep can be directly used in the case of varying length time series and does not require any padding strategies.

Table 8: Ablation studies with clusterability evaluation.

Configuration	Training	Manipulation			Water Monitoring Robot			QCAT-6		
	Time (hrs)	NMI ↑	Silhouette ↑	DBI ↓	NMI ↑	Silhouette ↑	DBI ↓	NMI ↑	Silhouette ↑	DBI ↓
TS-Rep (fixed length)	0.64	0.65	0.31	1.26	1.00	0.70	0.50	0.80	0.41	1.10
TS-Rep (varying length)	0.95	0.68	0.30	1.26	1.00	0.73	0.50	-	-	-
W/o Non-overlapping subseries	0.71	0.52	0.20	1.83	0.85	0.35	1.26	0.76	0.19	2.10
W/o Batch Negatives	4.93	0.66	0.31	1.24	1.00	0.68	0.53	0.81	0.43	1.09
W/o NN	0.71	0.63	0.24	1.51	1.00	0.67	0.55	0.79	0.36	1.17

H Implementation Details

As a pre-processing step, we standardise all the datasets such that the mean of each feature in the dataset is close to 0 and the standard deviation is close to 1. Additionally, for a fair comparison across all methods, we convert varying length time series into fixed length time series. We perform zero-padding in Water Monitoring Robot and QCAT datasets, data resampling in the Manipulation dataset and we use QCAT-6 and PUTany datasets as it is. We use a batch size of 8, 64, and 64 for the training of Manipulation, Water Monitoring Robot, and QCAT-6, respectively. Furthermore, we fix the output representation dimension L to 100 for all datasets and approaches. We use the Adam optimizer with a learning rate of 0.001 and with decay rates of $(0.9, 0.999)$. To crop time series into 2 non-overlapping subseries, for each crop, we set the minimum and maximum length to 30% and 70% of the original length respectively, which was chosen without any tuning. We follow T-Loss [14] and TS2Vec [30] to obtain the instance-level representation by performing max pooling over all timesteps. For the nearest neighbour, we have a warm-up period of 30 epochs for all the datasets, and we use the L1 distance and probability $p = 0.6$ for the Bernoulli distribution. The mean and standard deviation for all experiments are obtained through 10 independent runs with random initialization of weights.

To train the network on QCAT and PUTany, we use a batch size of 256. For comparison with IAE in Table 6,in addition to standard configuration of mini-batch size $B = 256$ and output representation dimension $L = 100$, we also use $B = 64$ and $L = 512$ (bottom row in Tables 6 & 7).

For clustering, we use Gaussian Mixture Model (GMM). We use implementations from scikit-learn [24] for GMM, OCSVM and SVM. For OCSVM, we use the default parameters. In the case of SVM, we follow [14, 30] and perform a grid search using training labels to find the best classifier.

The implementation of our method is based on Python 3, PyTorch [23] 1.10 with CUDA 11.1 on Nvidia GeForce GTX 1080 Ti GPU (11GB).

For T-Loss [14] and TS2Vec [30], we use the default parameters as described in the original papers. For TS-TCC [13], we refer to their default parameters used for the Human Activity Recognition (HAR) dataset.

Further details are reported in Table 9. Our implementation is publicly available at `https://github.com/imprs/TS-Rep`.

Table 9: TS-Rep configuration and implementation details.

	Manipulation	Water Monitoring Robot	QCAT-6	PUTany	QCAT
Pre-processing					
	Standardisation (mean 0 and standard deviation 1)				
	Resampling	Zero-padding	-	-	Zero-padding
Network					
Number of channels in the intermediate layers	40				
Number of layers/ depth of the network	10				
Kernel size of convolutions	3				
Negative slope for leaky ReLU	0.01				
Number of output channels from the causal network (before max pooling)	80				
Dimensions of output representation (L)	100				
Training params					
Batch size (B)	8	64	64	256	256
Number of iterations	2000				
Optimizer	Adam, leaning rate α = 0.001 with decay rates β = $(0.9, 0.999)$				
Nearest neighbour	Warm-up for 30 epochs, Distance = $L1$, Bernoulli dist. probability $p = 0.6$				
Hardware and Software					
	PyTorch 1.10 with CUDA 11.1				
	Nvidia GeForce GTX 1080 Ti GPU (11GB)				