Joint Embeddings Go Temporal

Sofiane Ennadir*
KTH
Stockholm, Sweden

Siavash Golkar
New York University
New York, USA

Leopoldo Sarra
Flatiron Institute
New York, USA

Abstract

Self-supervised learning has seen great success recently in unsupervised representation learning, enabling breakthroughs in natural language and image processing. However, these methods often rely on autoregressive and masked modeling, which aim to reproduce masked information in the input, which can be vulnerable to the presence of noise or confounding variables. To address this problem, Joint-Embedding Predictive Architectures (JEPA) has been introduced with the aim to perform self-supervised learning in the latent space. To leverage these advancements in the domain of time series, we introduce Time Series JEPA (TS-JEPA), an architecture specifically adapted for time series representation learning. We validate TS-JEPA on both classification and forecasting, showing that it can match or surpass current state-of-the-art baselines on different standard datasets. Notably, our approach demonstrates a strong performance balance across diverse tasks, indicating its potential as a robust foundation for learning general representations. Thus, this work lays the groundwork for developing future time series foundation models based on Joint Embedding.

1 Introduction

The paradigm of self-supervised learning (SSL) has emerged as a crucial technique for the development of Foundation Models [3]. These models are built on large unlabeled datasets and then fine-tuned for specific tasks with smaller labeled datasets. This approach has been very successful for language and vision tasks, and has been recently applied also to time series [5, 9]. SSL methodologies can be broadly categorized into two families. The first is based on contrastive approaches [21], such as SimCLR [4] developed for computer vision, which learns representations by juxtaposing positive and negative pairs of samples. Building upon these principles, researchers have also introduced novel adaptations [20, 21, 22] that account for the specific properties of time series. The second family encompasses predictive or generative-based methods. These techniques, such as the Masked Auto-Encoder (MAE) [11], task the model to reconstruct or predict missing segments of the input. This approach can be particularly effective for time series, given their inherent temporal ordering and often cyclical structure, such as in seasonal retail sales [7]. A variation of MAE is the Autoregressive Encoder, where the model's objective becomes the prediction of the next time steps. This adaptation aligns particularly well with the forward-looking nature of many time series applications, enhancing the capacity to capture temporal dependencies specifically in the forecasting task.

However, since masked models are designed to reconstruct missing parts in the input space, they can be susceptible to the presence of noise and other non-predictable confounding factors. Indeed, a good reconstruction requires to model the entire input, and such elements may prevent the extraction of meaningful and predictive features, because the model will focus on this noise rather than the underlying patterns within the data [12]. Recently, the Joint-Embedding Predictive Architecture (JEPA) [12] has been introduced as a way to only maintain the relevant information in the representation, leading

*Work done during internship at Flatiron Institute - Corresponding Author: ennadir@kth.se

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

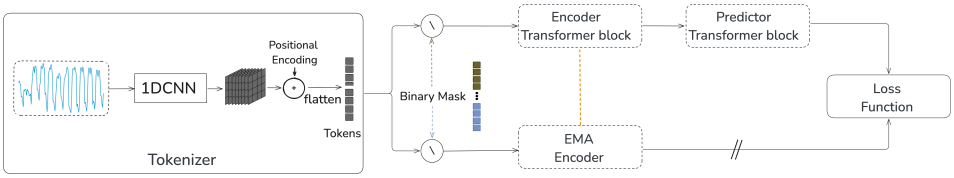

Figure 1: Illustration of TS-JEPA: it consists of (1) a tokenizer, (2) an encoder that processes the non-masked patches, (3) a Predictor that generates the target predictions from the encoder's output and (4) the EMA-Encoder, which encodes the target masked patches.

to competitive results in both the image [1] and video domains [2]. The JEPA approach first encodes the input into a latent space and then performs masked reconstruction in this space. Latent space reconstruction makes this technique robust to the presence of confounding variables and noise in the input. Inspired by the success of JEPA in addressing confounding factors in images and videos, we believe that this approach can similarly benefit time series data, which are often inherently noisy.

Here, we propose TS-JEPA, an adaptation of the JEPA architecture specifically tailored to the unique characteristics of temporal sequences. We validate our model through experiments on various standard datasets for time series classification and short and long-term forecasting. TS-JEPA exhibits promising performance compared to other baselines with the induced representation performing competitively in the forecasting task while outperforming significantly on classification. While a few papers, have already appeared mentioning JEPA in the context of time series, they were either a specific application to encoded frames [8], or in combination with other techniques for in-context prediction [19]. Ours is the first systematic study of the JEPA architecture to time series, investigating whether this method can compete against the more common autoregressive approach. Our results positions TS-JEPA as a promising building block to be used for time series foundation models.

2 Joint Embedding Predictive Architecture for Time Series

We consider the task of representation learning for time series. We focus on the univariate case, but our architecture is easily adaptable to multivariate time series. Formally, let $\mathcal{X} = \{x_1, x_2, ..., x_N\}$ represent a set of N time series, where each x_j is of length T. We aim to find a function $f : \mathcal{X} \to \mathcal{H}$ mapping the input time series to a representation $h \in \mathcal{H} \subset \mathbb{R}^m$, where m is the hidden representation's dimension. Once the self-supervised representation learning process has been completed, the learned representations can be used for downstream tasks such as classification or forecasting.

The fundamental idea of JEPA is to use non-masked parts of the input to predict other masked parts, but in an *abstract latent space* rather than in the original input space. This approach allows the model to capture underlying patterns and relationships in the data that may not be immediately apparent in the raw input. Our architecture is composed of four primary components (Fig. 1):

(i) Tokenizer: serves as the initial processing stage. It takes as input a time series $x \in \mathcal{X}$ and transforms it into a sequence of non-overlapping patches $p_i \in \mathcal{P}$. To embed these patches, we use a one-dimensional convolutional neural network (1D-CNN). The aim is to capture local patterns and features along the temporal dimension within each patch. In addition, to preserve the temporal information, we incorporate a positional encoding mechanism. While alternative positional encoding schemes could easily be incorporated in our architecture, we have adopted an absolute sin-cos positional embedding [18] for simplicity. The sequence of patches are split into a set of masked $\mathcal{P}_\mathcal{M}$ and non-masked patches $\mathcal{P}_\mathcal{N}$ following a uniform masking strategy.

(ii) Encoder E_θ: is the cornerstone of our architecture, transforming the patches into a useful representation that can be used afterwards for downstream tasks. We use a standard transformer architecture incorporating self-attention mechanisms. The encoder takes as input the non-masked patches $\mathcal{P}_\mathcal{N} = \{p_i \in \mathcal{P} \mid i \in \mathcal{N}\}$, where $\mathcal{N}$ represents the indices of non-masked elements. The inputs are transformed into the latents $z_\mathcal{N} = E_\theta(\mathcal{P}_\mathcal{N})$.

(iii) Predictor P_β: takes the output of the encoder $z_\mathcal{N}$ as input. The goal is to learn a mapping from the encoded observed (i.e. non-masked) tokens to the encoded unobserved (masked) ones. Similar to the Encoder, we use a transformer-based architecture with self-attention mechanisms. Specifically, the output of the predictor can be formulated as $z'_\mathcal{M} = P_\beta(E_\theta(\mathcal{P}_\mathcal{N}))$.

(iv) EMA-Encoder $E_{\bar{\theta}}$: The Exponential Moving Average encoder is an instance of the encoder with different weights. Instead of them being directly optimized through backpropagation like in the main encoder, the weights of the EMA-encoder are updated as an exponential moving average of the main encoder's weights. This has been shown to prevent collapse during training [12, 10]. This component takes as input the set of masked patches $\mathcal{P}_{\mathcal{M}}$ and produces representations $t_{\mathcal{M}} = E_{\bar{\theta}}(\mathcal{P}_{\mathcal{M}})$, which serve as the target for the predictor. Additional details are provided in Appendix.

The learning task consists of predicting the encoding of the masked tokens from those of the non-masked tokens in a latent space, specifically by minimizing

$$\mathcal{L} = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mid\mid z_i' - t_i \mid\mid_1 = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \mid\mid P_\beta(E_\theta(\mathcal{P}_{\mathcal{N}}))_i - E_{\bar{\theta}}(p_i) \mid\mid_1 .$$

3 Experimental Setup

We first pretrain TS-JEPA using all available training samples for each dataset. Subsequently, we evaluate the learned representation using a frozen evaluation protocol. Specifically, we keep the pre-trained encoder fixed and train a small classification/regression heads for the downstream tasks. This allows us to assess the quality and transferability of the learned representations. For simplicity, we maintain the same architecture for both our encoder and predictor, i.e. a transformer with 2 attention heads and an embedding dimension of 128. All the models have been trained using the AdamW optimizer [15], and the learning rates have been tuned to maximize the performance of each method. We report the specific details in the Appendix.

Baselines. We compare the performance of TS-JEPA against three alternative methods: (i) TS2Vec [20], based on contrastive learning, (ii) MAE [11], which employs a mask-and-reconstruct paradigm in the input space and (iii) an autoregressive approach, which predicts future values based on past observations. We always use the same architecture as the one used in TS-JEPA's encoder (in terms of dimensions and number of attention heads) to ensure the fairness of the comparison. To assess the effectiveness of the encoder's pretraining, we also report results obtained by training only the classification/regression head on a frozen, randomly initialized (non-pretrained) encoder. This comparison helps determine whether the encoder has truly captured the underlying patterns of the dataset or if the task is so simple that a linear layer alone can achieve good performance. We additionally consider the fully supervised case, where a transformer and a CNN are trained end-to-end on the downstream tasks. These supervised baselines provide an upper bound to gauge how close our approach can get to fully supervised learning.

Datasets. For classification, we use FordA, FordB [6], FaultDetectionA and FaultDetectionB [13], all of which comprise outputs from various sensors. Additionally, we include ECG500 [6], consisting of single-sensor electrocardiogram recordings. For forecasting tasks, we employ the Weather dataset [16], which contains recordings of diverse meteorological indicators. We also consider ETT-Small [23], representing electricity transformer temperature data, and the Electricity dataset [17], which measures electric power consumption in a single household.

Evaluation tasks. We focus on classification and forecasting. For classification, we consider two settings. The first involves using the same dataset for both pre-training and evaluation, while the second assesses the transferability of the pretrained model to similar datasets with similar tasks. Specifically, we pretrain on FordA (resp. FaultDetectionA) and evaluate on FordB (resp. FaultDetectionB). We evaluate the performance on the forecasting task in two ways. We first focus on short-term forecasting, where, given the immediate context, we predict the next patch in the sequence. We then consider longer-term forecasting, where we predict a horizon window by autoregressively rolling out the input.

4 Experimental Results

Table 1 presents the average prediction accuracy and standard deviations (over 10 runs) for the classification downstream task. Our experiments show that TS-JEPA outperforms both contrastive and autoregressive approaches in the majority of classification tasks, while exhibiting comparable performance to the Masked Auto-Encoder approach. Notably, the resulting accuracy closely approximates that of a transformer trained in a fully supervised manner.

Table 1: Downstream Classification accuracy ($\pm$ standard deviation) using real-world datasets.

Method	Fully Supervised		Pre-trained					
	CNN	Transformers	Trained On	Classification Head	TS2Vec (Contrastive)	MAE	Auto-Regressive	TS-JEPA
FordA	86.8 ± 0.4	91.8 ± 0.5	✓	46.3 ± 0.2	86.4 ± 0.2	85.1 ± 0.6	69.6 ± 0.4	91.5 ± 0.1
FordB	70.5 ± 0.8	74.8 ± 1.1	×	51.2 ± 0.6	72.4 ± 0.7	59.6 ± 0.5	61.9 ± 0.3	73.8 ± 0.3
FaultDetectionA	98.4 ± 0.3	91.8 ± 0.8	✓	54.3 ± 2.1	83.9 ± 0.4	90.4 ± 0.3	81.6 ± 0.2	85.8 ± 0.1
FaultDetectionB	63.9 ± 1.2	54.3 ± 0.4	×	40.2 ± 3.7	53.9 ± 0.6	54.3 ± 0.4	51.4 ± 0.7	50.6 ± 1.3
ECG5000	87.3 ± 0.6	89.9 ± 1.8	✓	58.4 ± 0.1	86.9 ± 0.3	91.6 ± 0.7	87.5 ± 0.2	89.5 ± 0.1

To further validate the efficacy of our TS-JEPA approach, we conduct an additional evaluation in the context of fine-tuning with limited labeled data. In this experiment, we mimic a realistic scenario in which supervised labels are sparse. Specifically we only consider a fraction of labels from our labeled data, with varying size from 5% to 20% of the total labels and treat the rest of the dataset as unlabeled samples used for pretraining. Figure 2 illustrates the performance of TS-JEPA compared to a fully-supervised transformer model across different amounts of available labeled data. TS-JEPA shows superior sample efficiency, achieving higher performance with fewer labeled examples. As expected, we observe that the performance gap between TS-JEPA and fully supervised method narrows as the amount of labeled data increases, with the gap shrinking with more labels.

Table 2 displays the average MSE and MAE for the short-term forecasting strategy (confidence level provided in the Appendix), while Figure 3 illustrates the cumulated MSE for long-term forecasting. As anticipated, the autoregressive approach outperforms TS-JEPA in short-term forecasting tasks, which is consistent with the training paradigm of these models. In long-term forecasting, both approaches exhibit the inherent uncertainty amplification effect characteristic of the roll-out strategy. However, TS-JEPA demonstrates superior performance to autoregressive strategies in two out of three datasets (i.e. ETT and Electricity), suggesting an enhanced stability.

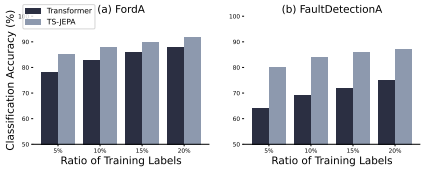

Figure 2: Performance of TS-JEPA against a full-supervised Transformer when subject to less training labels on the FordA dataset (a) and FaultDetectionA dataset (b).

Table 2: MSE and MAE of short-term forecasting.

Dataset	ETT-Small		Weather		Electricity	
	MSE	MAE	MSE	MAE	MSE	MAE
Auto-regressive	0.009	0.083	0.022	0.108	0.010	0.076
JEPA	0.017	0.110	0.015	0.109	0.014	0.086

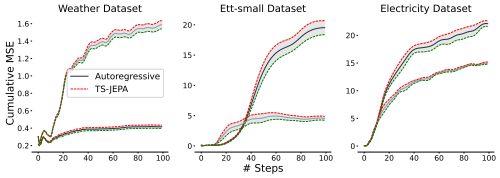

Figure 3: Comparison of the Cumulative Mean Square Error on the long-term forecasting task.

Overall, these results show promising performance for TS-JEPA in both classification and forecasting tasks. While the method does not consistently outperform all baselines on every dataset and task, by maintaining competitive forecasting capabilities and outperforming on classification, it offers a compelling trade-off between the two tasks for using a single architecture.

5 Conclusion

We introduce TS-JEPA, an adaptation of the JEPA Architecture tailored specifically for self-supervised learning in time series analysis. The experimental results demonstrate that TS-JEPA achieves a great balance between performance in classification and forecasting downstream tasks. This balanced capability sets TS-JEPA apart from current state-of-the-art methods, particularly the widely-used autoregressive approach, which often excels in one task (e.g. forecasting) at the expense of the other. The versatility of TS-JEPA makes it a promising candidate for developing adaptable foundation models for time series analysis. Next steps will include the exploration of scaling strategies for TS-JEPA, with the ultimate goal of establishing a new paradigm for time series foundation models.

Acknowledgements

We thank Alberto Bietti, Francois Lanusse, Rudy Morel for various discussions. The computations in this work were run at facilities at the Flatiron Institute, a division of the Simons Foundation, and we are thankful to the Scientific Computing Core and Shirley Ho for their support.

References

[1] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023.

[2] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-JEPA: Latent video prediction for visual representation learning, 2024.

[3] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, et al. On the opportunities and risks of foundation models, 2021.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[5] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv:2310.10688*, 2023.

[6] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[7] Yasaman Ensafi, Saman Hassanzadeh Amin, Guoqing Zhang, and Bharat Shah. Time-series forecasting of seasonal items sales using machine learning–a comparative analysis. *International Journal of Information Management Data Insights*, 2(1):100058, 2022.

[8] Abanoub M Girgis, Alvaro Valcarce, and Mehdi Bennis. Time-series jepa for predictive remote control under capacity-limited networks. *arXiv:2406.04853*, 2024.

[9] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

[10] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[12] Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

[13] Christian Lessmeier, James Kuria Kimotho, Detmar Zimmer, and Walter Sextro. Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification. In *PHM Society European Conference*, volume 3 1, 2016.

[14] Ziyu Liu, Azadeh Alavi, Minyi Li, and Xiang Zhang. Self-supervised learning for time series: Contrastive or generative? In *Workshop on Artificial Intelligence for Time Series, IJCAI 2023*, 2023.

[15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[16] Max Planck Institute for Biogeochemistry. Weather data, 2024.

[17] Artur Trindade. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86.

[18] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[19] Stijn Verdenius, Andrea Zerio, and Roy LM Wang. Lat-pfn: A joint embedding predictive architecture for in-context time-series forecasting. *arXiv:2405.10093*, 2024.

[20] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36 8, pages 8980–8987, 2022.

[21] Chaoning Zhang, Chenshuang Zhang, Junha Song, John Seon Keun Yi, Kang Zhang, and In So Kweon. A survey on masked autoencoder for self-supervised learning in vision and beyond. *arXiv:2208.00173*, 2022.

[22] Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. *Advances in Neural Information Processing Systems*, 35:3988–4003, 2022.

[23] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, May 2021.

Appendix: Joint Embedding go Temporal.

1 Pre-training Details

Architecture. Our encoder and decoder share an identical architecture: a transformer with 2 attention heads and an embedding dimension of 128. While we have not conducted an ablation study on this configuration, it has proven sufficient to yield satisfactory results in downstream tasks. We defer a more thorough investigation of architectural choices to future work. To ensure a fair comparison, all considered baselines utilize the same architecture as our TS-JEPA, allowing us to isolate the effects of the self-supervised task. Across all datasets and tasks, we segment each time series into 10 patches and employ a batch size of 32. For TS-JEPA, we apply a masking ratio of 70%, while for MAE, we use 75%, consistent with the implementation in [14].

Optimization. We train all models using the AdamW optimizer [15], with learning rates fine-tuned to maximize the performance of each method. Specifically, we explore learning rates within the range [1e-03, 1e-04, 1e-05, 1e-06]. Table 3 and Figure 4 and 5 illustrate the impact of learning rate on both long-term and short-term forecasting downstream tasks for both Autogressive approach and TS-JEPA.

On the EMA-Encoder The use of two instances of the same encoder has been shown to lead to a representation collapse, where the encoder learns the trivial solution of a constant output regardless of the input. [12]. Typically, this solution minimizes the reconstruction loss but fails to capture any meaningful information about the data. By using the EMA-encoder, we introduce a slowly moving target for the predictor, helping to stabilize training and encourages the model to learn more robust and meaningful representations. The use of an EMA-encoder has been validated in previous work, including the Bootstrap Your Own Latent (BYOL) [10] method for self-supervised learning, and in JEPA architectures applied to images [1] and videos [2]. For our implementation we have used set the update parameter $m = 0.998$, which we have seen to emperically give good results while avoid the collapse.

Table 3: Effect of Learning Rate on both JEPA and Autoregressive approach.

Dataset	**Learning Rate:**	$1e-03$		$1e-04$		$1e-05$		$1e-06$	
	Metric:	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT-Small	Auto-regressive	0.035	0.123	0.028	0.113	0.009	0.083	0.042	0.152
	JEPA	0.031	0.133	0.031	0.114	0.017	0.110	0.019	0.114
Weather	Auto-regressive	0.054	0.153	0.039	0.160	0.022	0.108	0.030	0.124
	JEPA	0.017	0.101	0.053	0.172	0.037	0.147	0.015	0.109
Electricity	Auto-regressive	0.013	0.089	0.022	0.114	0.010	0.076	0.048	0.145
	JEPA	0.014	0.086	0.058	0.198	0.022	0.121	0.026	0.140

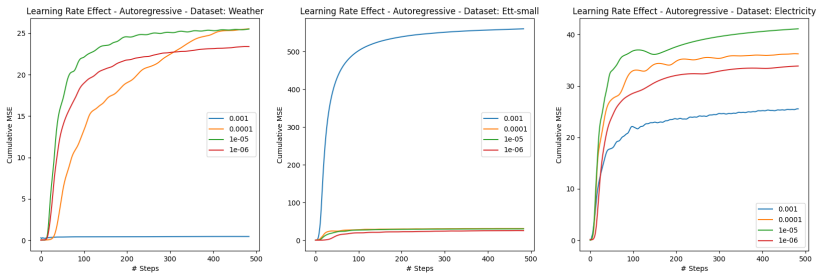

Figure 4: Effect of Learning Rate on the long-term forecasting for autoregressive models.

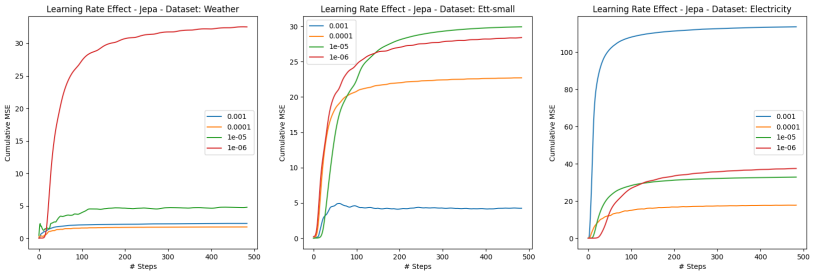

Figure 5: Effect of Learning Rate on the long-term forecasting for JEPA.

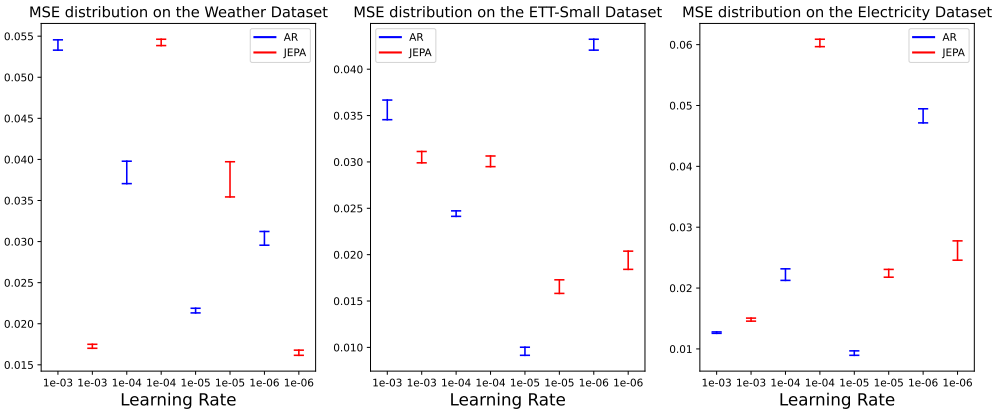

Figure 6: MSE distributions for each experiment.

2 Datasets and Implementation Details

Implementation details. The experimental implementation of TS-JEPA is available on github[2]. It is built using the open-source PyTorch library. We use the official, publicly available implementation of TS2Vec [20] and the MAE implementation provided by [14]. All experiments were conducted on an NVIDIA V100 GPU.

Table 4: Statistics of the classification datasets used in our experiments.

Dataset	#Training Point	#Test Points	#Length	#Classes
FordA	3601	1320	500	2
FordB	3636	810	500	2
FaultDetectionA	10912	2728	5120	3
FaultDetectionB	10912	2728	5120	3
ECG500	500	4500	140	5

[2]https://github.com/Sennadir/TS_JEPA