Toward a Foundation Model for Time Series Data

Chin-Chia Michael Yeh
Xin Dai
Huiyuan Chen
Yan Zheng
Yujie Fan
Visa Research
California, USA

Audrey Der
University of California, Riverside
California, USA

Vivian Lai
Zhongfang Zhuang
Junpeng Wang
Liang Wang
Wei Zhang
Visa Research
California, USA

ABSTRACT

A foundation model is a machine learning model trained on a large and diverse set of data, typically using self-supervised learning-based pre-training techniques, that can be adapted to various downstream tasks. However, current research on time series pre-training has predominantly focused on models trained exclusively on data from a single domain. As a result, these models possess domain-specific knowledge that may not be easily transferable to time series from other domains. In this paper, we aim to develop an effective time series foundation model by leveraging unlabeled samples from multiple domains. To achieve this, we repurposed the publicly available UCR Archive and evaluated four existing self-supervised learning-based pre-training methods, along with a novel method, on the datasets. We tested these methods using four popular neural network architectures for time series to understand how the pre-training methods interact with different network designs. Our experimental results show that pre-training improves downstream classification tasks by enhancing the convergence of the fine-tuning process. Furthermore, we found that the proposed pre-training method, when combined with the Transformer, outperforms the alternatives. The proposed method outperforms or achieves equal performance compared to the second best method in ~ 93% of downstream tasks.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Temporal reasoning**.

KEYWORDS

time series, self-supervised learning, foundation model

ACM Reference Format:
Chin-Chia Michael Yeh, Xin Dai, Huiyuan Chen, Yan Zheng, Yujie Fan, Audrey Der, Vivian Lai, Zhongfang Zhuang, Junpeng Wang, Liang Wang, and Wei Zhang. 2023. Toward a Foundation Model for Time Series Data. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583780.3615155

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3615155

1 INTRODUCTION

Foundation models are a type of machine learning model that are trained through self-supervised learning on large-scale, multi-domain datasets and can be fine-tuned for a wide range of downstream tasks [3]. Although extensively studied in natural language processing and computer vision communities [3, 11, 23, 36], there has been surprisingly little work focused on building foundation models for time series data. Several papers have explored self-supervised learning with time series data [26, 32, 37, 39], but none have tested their methods on a pre-training dataset consisting of time series from multiple domains. In other words, while these methods may provide valid solutions for training foundation models for time series data, experiments involving multi-domain pre-training datasets have not yet been conducted.

To investigate this research problem, we utilized the publicly available time series dataset archive, the UCR Archive [10], and repurposed it for our experimental setup as illustrated in Fig. 1. We compiled a pre-training set comprising time series from all domains in the UCR Archive, ranging from power consumption to heartbeats and food spectrographs. Furthermore, we generated multiple mutually exclusive training, validation, and test sets for downstream classification tasks. Using the repurposed UCR Archive, we aimed to address the following three research questions: 1) Can pre-training a foundation model on a multi-domain dataset lead to improved performance on downstream single-domain classification tasks after fine-tuning? 2) What neural network architectures are most effective for the foundation model? 3) What is the most effective self-supervised learning method for building the foundation model?

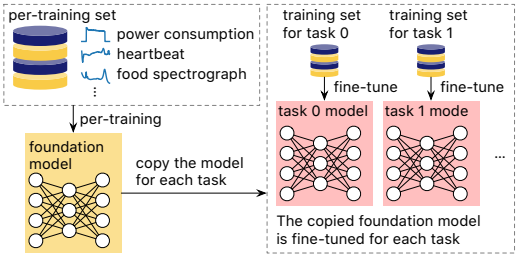

Figure 1: The foundation model is initially trained on a large dataset of time series from various domains, and is then fine-tuned using a smaller training set for each downstream task.

To address these research questions, we built foundation models using four widely used neural network architectures (Section 2), and trained them using four existing and one novel self-supervised learning methods (Section 3). In total, we tested 20 different configurations for building foundation models. Our experimental results showed that pre-training a foundation model with a multi-domain dataset can improve the performance of downstream single-domain classification tasks after fine-tuning. One possible reason for the improved performance using a foundation model is that it has a smoother convergence curve (i.e., better convergence) when fine-tuned compared to training a model from scratch. Furthermore, we found that the best-performing foundation model was achieved by combining the proposed self-supervised learning method with the Transformer architecture. Our contributions in this study are:

- We describe a set of steps to repurpose the UCR Archive for the study of building time series foundation models.
- We demonstrate that utilizing foundation models trained with time series from multiple domains can indeed benefit downstream classification tasks.
- We observe that the convergence curve of the foundation model during the fine-tuning stage is much smoother compared to training a model from scratch, which helps the pre-trained models achieve better performance.
- We propose an effective learning method for training time series foundation models that outperforms alternative methods.

2 MODEL ARCHITECTURE

We explored four different neural network architectures as the backbone model for the foundation models:

LSTM: The Long Short-Term Memory Network (LSTM) is a widely-used type of Recurrent Neural Network (RNN) for modeling sequential data [15, 19, 40]. We use the design illustrated in Fig. 2.c. In the figure, we use the notation `1D conv,7/2,1→64` to denote a 1D convolutional layer with a filter size of 7, a stride size of 2, an input dimension of 1, and an output dimension of 64. Similarly, we use the notation `bi-RNN,64→64` to denote a bidirectional RNN layer with an input dimension of 64 and an output dimension of 64. In this case, the two `bi-RNN` layers are bidirectional LSTM layers. Finally, we use the notation `linear,64→64` to denote a linear layer with an input dimension of 64 and an output dimension of 64.

GRU: The Gated Recurrent Unit Network (GRU) is another popular type of RNN architecture widely used for modeling sequential data [8, 19, 40]. We adopt the design depicted in Fig. 2.c with two bidirectional GRU layers.

ResNet: The Residual Network (ResNet) is a time series classification model that takes inspiration from the success of ResNet in computer vision [14, 29]. Extensive evaluations by [16] has demonstrated that ResNet is one of the strongest models for time series classification. Our design, as shown in Fig. 2.b, is based on the design proposed by [29]. We use the notation `block, 64→64` to denote a residual block (Figure2.a) with an input dimension of 64 and an output dimension of 64.

Transformer: The Transformer (XFMR) is a widely used alternative to RNNs for sequence modeling [5, 18, 19, 28, 40]. The architecture we used is shown in Fig. 2.d. We used fixed positional encoding, following [28], and included a special token `[start]`

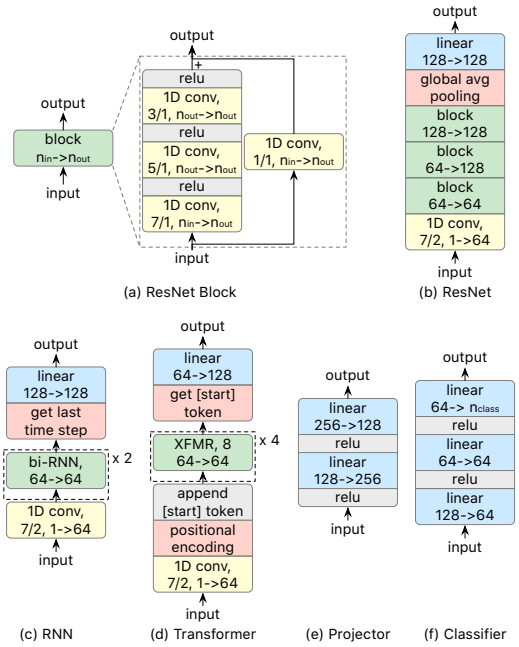

Figure 2: The designs for ResNet, RNN, and Transformer are illustrated in Fig. 2.a to d. The projector and classifier are used for pre-training and fine-tuning.

to learn the representation of the entire time series. Our Transformer architecture consists of four encoder layers, each denoted as `XFMR,8,64→64` where the number of heads is 8, the input dimension is 64, and the output dimension is 64.

We use layer normalization [1] for all normalization layers, as it is both effective and widely used in modeling sequential data [1, 28].

3 PRE-TRAINING METHOD

We evaluated a total of five pre-training methods. The first four methods listed below serve as the baseline methods, while the fifth method is the proposed method.

SimCLR: The SimCLR method was originally proposed as a self-supervised pre-training method based on contrastive learning for computer vision [6], and was later extended to human activity time series by [26]. The pre-training procedure for SimCLR involves a batch of time series X that undergoes a sequence of randomized data augmentation methods to generate two augmented batches X_0 and X_1. The data augmentation methods used are random scaling and negation, following the open-sourced implementation [9, 39]. Both X_0 and X_1 are then processed by the backbone model (Section 2) and the projector (Fig. 2.e); the output feature vectors are denoted as H_0 and H_1, respectively.

Next, the NT-Xent loss function [6, 26] is used to compute the loss. If $h_i \in H_0$ and $h_j \in H_1$ are features extracted from the augmented versions of the same time series in X, the loss for the positive pair (h_i, h_j) is computed as follows: $\mathcal{L} = -\log \frac{\exp(\text{sim}(h_i,h_j)/\tau)}{\sum_{h_k \in H_0+H_1} \mathbb{1}_{[h_k \neq h_i \& h_k \neq h_j]} \exp(\text{sim}(h_i,h_k)/\tau)}$. Here, $\text{sim}(\cdot,\cdot)$ is a function that computes the cosine similarity between the input vectors, h_k is a feature vector from H_0 or H_1 that is not h_i nor h_j, and τ is the temperature parameter. Both the backbone model and

the projector are optimized using the NT-Xent loss. To fine-tune the model for the downstream classification task, we add a classifier model (Fig. 2.f) on top of the projector as shown in Fig. 3.a, and update the backbone, projector, and classifier using the cross-entropy loss.

TS2Vec: The TS2Vec method is another pre-training method that utilizes contrastive learning [37]. What differentiates the TS2Vec method from SimCLR is that: 1) the positive pairs are generated using the contextual consistency strategy and 2) the contrastive loss is calculated in a hierarchical fashion. The contextual consistency strategy generates positive pairs by randomly cropping two overlapping subsequences from a time series. The TS2Vec method computes the contrastive loss under multiple levels of time granularity. At each time granularity, two contrastive losses are computed: temporal contrastive loss and instance-wise contrastive loss. Both are modified NT-Xent losses where the former helps the model learn discriminative representations over time, and the latter helps the model learn discriminative representations over samples.

Note that TS2Vec requires the backbone model to generate representation for multiple time steps. To achieve this, we modify the RNN-based models (Fig. 2.c) by removing the `get last time step` block and apply the subsequent linear layer and projector model independently at each time step to generate the per-time-step representation. For ResNet and Transformer (Fig. 2.b and Fig. 2.d), we remove the `global avg pooling` and `get [start] token` block, respectively, and apply the subsequent linear layer and projector model independently at each time step to generate the per-time-step representation. Because the first layer for all the backbone models is a $1D$ convolutional layer with a stride size of 2, the length of the output sequence is halved from the input time series. To fine-tune the model for classification, we first re-add the removed blocks back to the model, add a classifier model (Fig. 2.f) on top of the projector as shown in Fig. 3.a, and update the backbone, projector, and classifier using the cross-entropy loss.

MixingUp: The MixingUp method was also developed based on the idea of contrastive learning [32]. Given a pair of time series (x_i, x_j), it generates a mixed time series x_k by computing $\lambda x_i + (1-\lambda)x_j$, where λ is the mixing parameter that determines the contribution of x_i and x_j in x_k. The mixing parameter is randomly drawn from a beta distribution. The self-supervised task for pre-training is to predict λ from the representations associated with x_i, x_j, and x_k. The representations are generated by processing x_i, x_j, and x_k with the backbone and projector model. The loss function adopted by [32] is a modified version of the NT-Xent loss. Similar to the previous two methods, fine-tuning for the classification task is achieved by adding a classifier model on top of the projector (Fig. 3.a) and updating the backbone, projector, and classifier model using the cross-entropy loss.

TF-C: The TF-C method extends the idea of contrastive learning to the frequency domain [39]. Based on the open-source implementation [9, 39], the self-supervised learning procedure is as follows: The method involves transforming a time series from the time domain to the frequency domain using the Fast Fourier transform (FFT). Positive pairs are generated through augmentation functions applied to the time series in both the time and frequency domains. TF-C employs the jittering technique to augment the time series in the time domain and the technique of adding/removing frequency components in the frequency domain. The input time series are processed in both the time and frequency domains using their respective backbone models and projectors. The self-supervised learning loss is calculated using a modified NT-Xent loss function, which involves intermediate representations from both the time and frequency domains. Since the TF-C method uses two backbone models and two projector models, the width of these models is halved so that it has the same number of parameters as the other methods. The structure of the model for fine-tuning is shown in Fig. 3.b, where the output of the projector in time and frequency domain is concatenated before being fed into the classifier. The fine-tuning is once again achieved with cross-entropy loss.

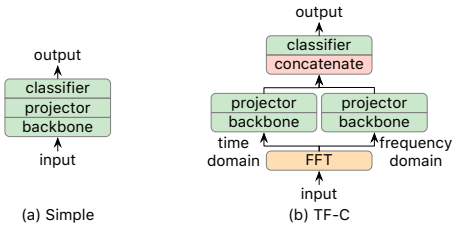

Figure 3: There are two ways to use backbone models for pre-training and fine-tuning. Fig. 3.a is used for all methods except TF-C, which uses Fig. 3.b.

TimeCLR: This is our proposed contrastive learning pre-training method, building on the simplest existing method, SimCLR [26]. The first enhancement we introduce is the incorporation of additional time series data augmentation techniques based on recent survey papers [17, 30]. Our method employs the following data augmentation techniques: jittering, smoothing, magnitude warping, time warping, circular shifting, adding slope, adding spike, adding step, masking, and cropping. These additional techniques enhance the pre-training process by allowing the model to learn more invariance properties. For instance, jittering and smoothing aid the model in being invariant to Gaussian noise, and magnitude and time warping help the model become warping invariant in both aspects. Circular shifting, adding slope/spike/step, and masking make the model invariant to corresponding noise types, while cropping helps the model learn contextual consistency similar to TS2Vec [37].

Our second enhancement involves using a single augmentation function for generating positive pairs instead of using all augmentation functions. This is due to the fact that we incorporate more data augmentation functions in our method compared to the existing SimCLR implementation [9, 39]. Applying all augmentation methods to the input time series could result in the augmented time series bearing little visual similarity to the original. For example, when we apply all the data augmentation methods to a simple pattern ( ) , the resulting time series ( ) loses many of the properties of the original time series. It is possible to achieve better performance by using more than one augmentation when generating positive pairs, but we left this exploration for future work. Apart from these two enhancements, both the pre-training and fine-tuning processes are identical to those of SimCLR. For fine-tuning, we add a classifier model on top of the projector (as

shown in Fig. 3.a) and update the backbone, projector, and classifier model using the cross-entropy loss.

4 EXPERIMENT

To ensure the reproducibility of our experiments, we have created a website [27] that contains supplementary materials, including hyper-parameter settings, source code, and complete experimental results. We first explain how we repurposed the datasets in the UCR archive [10]. We combined the original training set with the test set for each of the 128 datasets in the UCR archive. Next, we created a pre-training set by randomly extracting 50% of the samples from each of the 128 datasets in the UCR archive. In other words, the pre-training set contains time series samples from all 128 datasets. We then split the remaining time series samples in each of the 128 datasets into training, validation, and test sets with a ratio of 3:1:1. As a result, we have a single pre-training set, 128 training sets, 128 validation sets, and 128 test sets. It is important to note that all split sets are mutually exclusive, and there is no label information available in the pre-training set.

We combine the five pre-training methods (SimCLR, TS2Vec, MixingUp, TF-C, and TimeCLR) with the four neural network architectures (LSTM, GRU, ResNet, and Transformer), resulting in 20 different configurations for the pre-training methods. First, we train the foundation model using each configuration with the pre-training set. Next, we copy each trained foundation model for each of the 128 downstream tasks, where we train with the corresponding training set for the task. We perform model selection[1] with the task-specific validation set and measure the performance (i.e., accuracy) with the associated test set.

We also include a *no pre-training* baseline for each network architecture. For this baseline, we perform model training, model selection, and evaluation using only the task-specific dataset. In addition to the neural network-based methods, we also include one-nearest-neighbor with Euclidean Distance (ED) and Dynamic Time Warping distance (DTW) as baselines in our experiment. ED and DTW are considered simple yet effective baselines for time series classification problems [2, 10, 24]. The experimental results across 128 downstream tasks are summarized in Table 1. We report the average rank among the tested methods in the table.

Table 1: The average ranks of different methods across 128 datasets in UCR Archive are reported, with lower ranks indicating better performance. XFMR means Transformer. For each backbone model, we highlight the best performance in bold and the second best performance with an underline.

Average rank (↓)	ED	DTW	LSTM	GRU	ResNet	XFMR
No pre-train	15.14	14.08	21.33	20.96	11.16	11.38
SimCLR	-	-	21.38	20.98	11.12	11.77
TS2Vec	-	-	<u>13.71</u>	16.07	10.56	<u>11.37</u>
MixingUp	-	-	14.18	<u>12.16</u>	11.19	11.38
TF-C	-	-	14.53	13.24	<u>10.55</u>	11.48
TimeCLR	-	-	**12.43**	**9.51**	**10.04**	**9.30**

When we consider the method without any pre-training, we can see that ResNet and Transformer outperform the other methods. The superior performance of ResNet is not surprising as the results are in agreement with [16]. The success of Transformer is also expected, as Transformers have achieved state-of-the-art performance in natural language processing [4, 20, 28], computer vision [7, 12, 13], and other time series problems [21, 31, 38]. On the other hand, the performance of RNN-based models (LSTM and GRU) is worse than simple baselines like ED and DTW.

When comparing the performance of various pre-trained models with their non-pre-trained counterparts, we can observe that the foundation model is beneficial in 14 out of 20 configurations. In particular, aside from SimCLR (which was designed specifically for human activity time series [26]), the performance of RNN-based models improved greatly relative to the configurations with ResNet or Transformer, such that the RNN-based models had comparable performance with the other two model architectures. Generally speaking, the foundation model can improve the performance of downstream tasks after fine-tuning. Additionally, when comparing the proposed TimeCLR method with other pre-training methods, we can see that the proposed TimeCLR achieves the best performance regardless of the type of architecture used for the backbone model. The overall best performance is achieved by combining the TimeCLR method with the Transformer backbone model.

In Fig. 4, we compare the convergence curves of non-pre-trained and TimeCLR pre-trained models on two datasets. The ArrowHead dataset is considered "harder" to converge compared to CBF. Pre-trained models on the harder dataset exhibit smoother convergence curves during fine-tuning compared to training from scratch, despite both experiments using the same model architecture and optimization settings. In contrast, for the easier dataset, both models behave similarly, and either model can be trained to achieve good performance. This difference in convergence for harder datasets is likely the main reason why the foundation model improves the overall performance of downstream tasks.

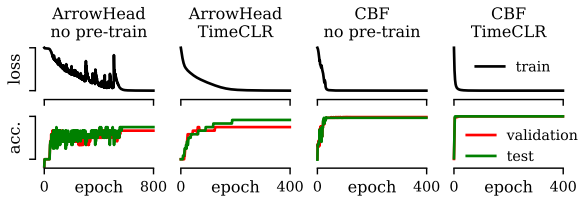

Figure 4: The convergence curves of both the non-pre-trained ResNet model and TimeCLR pre-trained ResNet model on the ArrowHead and CBF dataset.

5 CONCLUSION

In this paper, we have demonstrated the benefits of using self-supervised learning to train a foundation model for time series data from multiple domains. The proposed TimeCLR method outperformed the alternatives when combined with transformer models. For future work, it would be interesting to explore other ways of combining the benefits of different pre-training methods [26, 32, 37, 39], the possibility of using the data compression idea in self-supervised learning [22, 25, 34, 35], and testing alternative backbone model like the ResNet 2*D* [33].

[1]The model selection process selects the model from the best epoch and initialization strategy (i.e., random or pre-train) based on validation accuracy.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery* 31, 3 (2017), 606–660.

[3] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] Huiyuan Chen, Yusan Lin, Menghai Pan, Lan Wang, Chin-Chia Michael Yeh, Xiaoting Li, Yan Zheng, Fei Wang, and Hao Yang. 2022. Denoising Self-Attentive Sequential Recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 92–101.

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

[7] Xinlei Chen, Saining Xie, and Kaiming He. 2021. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9640–9649.

[8] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[9] Contributions. 2023. GitHub repository for Self-Supervised Contrastive Pre-Training For Time Series via Time-Frequency Consistency. https://github.com/mims-harvard/TFC-pretraining.

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

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[13] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. 2022. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence* 45, 1 (2022), 87–110.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[16] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery* 33, 4 (2019), 917–963.

[17] Brian Kenji Iwana and Seiichi Uchida. 2021. An empirical survey of data augmentation for time series classification with neural networks. *Plos one* 16, 7 (2021), e0254841.

[18] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems* 32 (2019).

[19] Bryan Lim and Stefan Zohren. 2021. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A* 379, 2194 (2021), 20200209.

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[21] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *arXiv preprint arXiv:2211.14730* (2022).

[22] Zongyue Qin, Yunsheng Bai, and Yizhou Sun. 2020. GHashing: Semantic Graph Hashing for Approximate Similarity Search in Graph Databases. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2062–2072.

[23] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[24] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 262–270.

[25] Qiaoyu Tan, Ninghao Liu, Xing Zhao, Hongxia Yang, Jingren Zhou, and Xia Hu. 2020. Learning to hash with graph neural networks for recommender systems. In *Proceedings of The Web Conference 2020*. 1988–1998.

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

[27] The Author(s). 2023. Supplementary Material. https://sites.google.com/view/timeclr.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[29] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*. IEEE, 1578–1585.

[30] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. 2020. Time series data augmentation for deep learning: A survey. *arXiv preprint arXiv:2002.12478* (2020).

[31] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2022. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125* (2022).

[32] Kristoffer Wickstrøm, Michael Kampffmeyer, Karl Øyvind Mikalsen, and Robert Jenssen. 2022. Mixing up contrastive learning: Self-supervised representation learning for time series. *Pattern Recognition Letters* 155 (2022), 54–61.

[33] Chin-Chia Michael Yeh, Huiyuan Chen, Xin Dai, Yan Zheng, Wang Junpeng, Vivian Lai, Yujie Fan, Audrey Der, Zhongfang Zhuang, Liang Wang, Wei Zhang, and Jeff M. Phillips. 2023. An Efficient Content-based Time Series Retrieval System. In *Proceedings of the 32nd ACM International Conference on Information & Knowledge Management*.

[34] Chin-Chia Michael Yeh, Mengting Gu, Yan Zheng, Huiyuan Chen, Javid Ebrahimi, Zhongfang Zhuang, Junpeng Wang, Liang Wang, and Wei Zhang. 2022. Embedding Compression with Hashing for Efficient Representation Learning in Large-Scale Graph. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4391–4401.

[35] Chin-Chia Michael Yeh, Yan Zheng, Junpeng Wang, Huiyuan Chen, Zhongfang Zhuang, Wei Zhang, and Eamonn Keogh. 2022. Error-bounded approximate time series joins using compact dictionary representations of time series. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*. SIAM, 181–189.

[36] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. 2021. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432* (2021).

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

[38] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2114–2124.

[39] Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. 2022. Self-supervised contrastive pre-training for time series via time-frequency consistency. *arXiv preprint arXiv:2206.08496* (2022).

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