A Systematic Evaluation of Generated Time Series and Their Effects in Self-Supervised Pretraining

Anonymous Author(s)

ABSTRACT

Self-supervised Pretrained Models (PTMs) have demonstrated remarkable performance across computer vision and natural language processing tasks. The success in these fields has prompted time series data mining researchers to design PTMs tailored to time series data. In our experiments, most self-supervised time series PTMs were surpassed by simple supervised models. We hypothesize this undesired phenomenon may be caused by data scarcity. In response, we train various time series generation methods, and compare how the use of each method's generated data in pretraining affects classification performance. Our experiment results indicate that replacing a real-data pretraining set with a greater volume of purely generated samples noticeably improves the performance.

CCS CONCEPTS

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

KEYWORDS

time series, pretraining, generative model

ACM Reference Format:

Anonymous Author(s). 2018. A Systematic Evaluation of Generated Time Series and Their Effects in Self-Supervised Pretraining. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Self-supervised Pretrained Models (PTMs) are models initially trained using self-supervised losses [29]. This initial training stage is generally known as the *pretraining*, and is unsupervised, not requiring labeled samples. This training paradigm has helped PTMs achieve promising performances in computer vision (CV) [10] and natural language processing (NLP) [32]. The unsupervised nature of the pretraining process allows PTMs to utilize a substantial amount of unlabeled data, and more robust features are extracted for the intended tasks. Motivated by these successes, numerous PTMs have been developed specifically for time series data [29].

We integrated four recently proposed self-supervised PTMs (TS2Vec [47], MixingUp [44], TF-C [49], TimeCLR [45]) with two popular network architectures (ResNet [17, 41] and Transformer [40]), but found that pretraining does not always enhance the performance of the model for classification tasks. In response, we explored pretraining with generated time series from various methods to produce samples for pretraining. Figure 1 describes our problem setting. Rather than pretraining the model with the pretraining data, we use this data to derive a time series generator, which is used to synthesize a large volume of data for pretraining. Once pretrained, we apply standard supervised training to fine-tune the model using training data. The resulting model is validated and tested using the validation and test data.

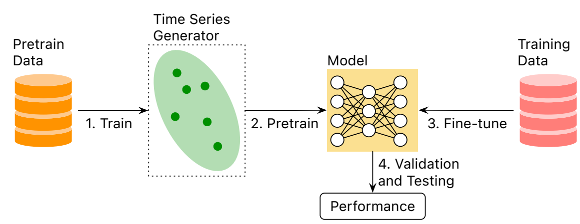

Figure 1: The time series generator can be utilized to synthesize data for self-supervised pretraining.

This work undertakes a comparative analysis to understand if and how using different generated time series during pretraining affects time series classification (TSC) performance. We experiment with combinations of one of four PTMs, one of six time series generators, and one of two network architectures. We compare the performance of each combination. Our findings suggest using generated time series in pretraining enhances the performance of PTMs compared to using an equally-sized or smaller pretraining set consisting of real data.

2 RELATED WORK

We conduct a comparative study on the interplay between time series generation, pretraining, and classification.

Generation. Time series generation has been a popular research area due to its success in CV and NLP [42]. Generative Adversarial Networks (GANs) have been used to generate time series for several domains, including music, speech, and EEG signals [48]. Variational Autoencoders (VAEs) [23] have been used for anomaly detection [14, 30], multivariate adversarial time series generation [16], and imputation [24]. Diffusion models have been used [19, 27] for time series forecasting [3, 34], imputation [3, 39], and generating waveforms [9, 25]. These models' success [22, 42] motivates us to examine outcomes using their generated time series in pretraining.

Pretraining. Research on time series PTMs is a burgeoning area of study in the field [29]. However, this task needs to be handled with care because of several complexities (inconsistent or unknown domain invariances, the wide semantic meaning across data types, varying sampling rates, differing data sources, etc. [15, 31, 49]).

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/XXXXXXX.XXXXXXX

These methods have shown promising results in a foundation model setting [45], where multiple datasets are used for pretraining. Taking inspiration from [45], we conducted experiments using four self-supervised contrastive learning pretraining methods: MixingUp [44], TimeCLR [45], TF-C [49], and TS2Vec [47].

Classification. There is an extensive body of work on TSC. ROCKET and its variants [37] extract features using convolutional kernels. High-performing deep learning models such as FCN [41], ResNet [41], and InceptionTime [21] all utilize convolution-based designs. Fawaz et al. [20] have demonstrated ResNet [41] to be a competitive baseline in TSC, and use ResNet [41] as the representative convolution-based architecture in our experiments. We also use transformers in our experiments, another popular choice for time series [2, 43, 46].

To our knowledge, no other systematic evaluation assesses the interaction between the time series generation methods and PTMs.

3 ARCHITECTURE COMPONENTS

3.1 Pretraining Methods

We consider four constrastive learning-based PTMs.

TimeCLR: TimeCLR is based on SimCLR [38, 45], which was originally proposed as a self-supervised pretraining method based on contrastive learning for computer vision [10], and was later extended to human activity time series by [38]. During pretraining, random scaling and negation augmentations [11, 49] are randomly applied to a batch of time series X to generate two augmented batches X_0 and X_1. Both X_0 and X_1 are then processed by the backbone model (Section 3.3) and the projector (Figure 2.c). The output feature vectors are denoted as H_0 and H_1, respectively.

The NT-Xent loss function [10, 38] 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 series in X, the loss for the positive pair (h_i, h_j) is computed as follows:

$$\mathcal{L} = -\log \frac{\exp\left(\mathrm{sim}(h_i, h_j)/\tau\right)}{\sum_{h_k \in H_0 + H_1} \mathbb{1}_{[h_k \neq h_i \& h_k \neq h_j]} \exp\left(\mathrm{sim}(h_i, h_k)/\tau\right)}$$

The cosine similarity is computed between the input vectors in $\mathrm{sim}(\cdot, \cdot)$, where 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 projector are optimized using the NT-Xent loss. We add a classifier (Figure 2.d) on top of the projector, as shown in Figure 2.a, to fine-tune the model for the classification task. The backbone, projector, and classifier are updated using the cross-entropy loss.

TS2Vec: The TS2Vec [47] uses the contextual consistency to generate positive pairs by randomly cropping two overlapping subsequences from a time series. It computes the contrastive loss in a heirarchial fashion under multiple levels of time granularity, where the temporal contrastive loss and instance-wise contrastive loss are computed at each time granularity. 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. To fine-tune the model for classification, we add a classifier model (Figure 2.d) on top of the projector (Figure 2.c); this is shown in Figure 2.a. Then the backbone, projector, and classifier are updated using the cross-entropy loss.

MixingUp: Given a pair of time series (x_i, x_j), this method [44] generates a mixed time series x_k by computing $\lambda x_i + (1 - \lambda) x_j$. The mixing parameter λ is randomly drawn from a beta distribution and determines the contribution of x_i and x_j in x_k. The pretraining self-supervised task predicts 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 using the backbone and projector model. The loss function adopted by [44] is a modified version of the NT-Xent loss. To fine-tune, we add a classifier to the projector (Figure 2.a), and update the backbone, projector, and classifier model using the cross-entropy loss.

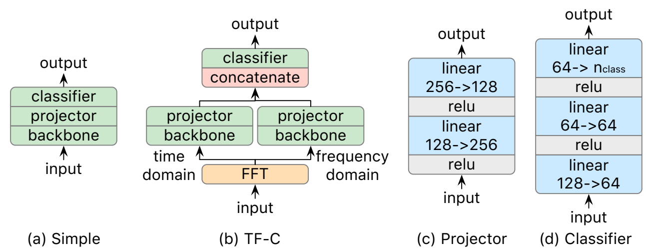

Figure 2: There are two ways to use backbone models for pretraining and finetuning. All pretraining methods but TF-C use Figure 2.a. TF-C uses Figure 2.b. The projector and classifier are shown in Figures 2.c and 2.d.

TF-C: TF-C extends the idea of contrastive learning to the frequency domain [49]. In the open-source implementation [11, 49], the self-supervised learning procedure transforms 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 uses jittering to augment the time series in the time domain, and adds and removes 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. During fine-tuning, the output of the projector in time and frequency domain is concatenated before being fed into the classifier (Figure 2.b), and is performed using cross-entropy loss.

3.2 Generative Models

In this paper, we examine three simple time series generators and three generative models.

Random Walk (RW): This generator produces random walk time series of specified length and dimensionality. Random walk is useful for simulating real-world time series, including fluctuating stock prices, Brownian motion, and the unpredictable paths of animals searching for food in the wild.

Sinusoidal Wave (SW): For each dimension, this method generates and combines two sinusoidal time series of a specified length with randomly sampled frequencies, amplitudes, and offsets to form each dimension of the output time series. Sinusoidal waves may serve as an ideal stand-in for real periodic time series.

Multivariate Gaussian (MG): The data is modeled as a multivariate Gaussian distribution in frequency domain, computing the mean and variance for each frequency from the pretraining set. Time series are generated by sampling this distribution. We do not consider the covariance between different frequencies.

Generative Adversarial Network (GAN): Our training framework takes inspiration from the BiGAN [13] with one difference: given our time series application, we use a 1D (rather than 2D) convolutional network. In each iteration of training, we train the encoder and generator with mean squared error reconstruction loss. Where X denotes training data, $G(\cdot)$ the generator, and $E(\cdot)$ the encoder:

$$\mathcal{L}_{\text{reconstruction}} = \text{MSE}(X, G(E(X)))$$

Where Z denotes random vectors and $D(\cdot)$ the discriminator, we train the discriminator with a critic loss [4]:

$$\mathcal{L}_{\text{critic}} = \text{mean}(D((G(Z)) - \text{mean}(D(X)))$$

A generator loss [4] is used to train $G(\cdot)$:

$$\mathcal{L}_{\text{generator}} = -\text{mean}(D(G(Z)))$$

We use the critic and generator losses described in [4].

β-Variational Autoencoder (β-VAE): The β-Variational Autoencoder (VAE) [18] utilizes the reparameterization trick, which aids in learning the data distribution within the latent space [23]. Rather than generating a fixed latent representation for each input sample, the VAE generates parameters linked to a specific distribution (typically a Gaussian distribution). The β-VAE introduces a hyper-parameter, β, which controls the relative importance of different loss terms.

The β-VAE is comprised of an encoder and a decoder. Our encoder design is similar to the $1D$ convolutional design in the GAN, with the exception the encoder in our β-VAE differs needs to generate both the mean μ and the log variance $\log \sigma^2$ for the Gaussian distribution. Consequently, our β-VAE encoder has two parallel output `Linear,2048→2048` layers[1]. We use the standard β-VAE training procedure [18] to train the encoder and decoder.

Diffusion Model (Diff.): Diffusion models [19] learn the data distribution by simulating the diffusion of data points through the latent space by gradually eliminating Gaussian noise within a Markov chain. Our design is inspired by the U-Net [35], and uses a $1D$ convolutional network. The diffusion model also utilizes downsample and upsample blocks, and we incorporate a skip connection akin to [35]. The input to each upsample block is concatenated with the intermediate representation transmitted via the skip connection, which help the network capture localized features [35].

3.3 Model Backbone Architecture

We focus on two widely utilized model architectures for time series data: the *ResNet* and *Transformer*.

ResNet: The Residual Network (ResNet) is a TSC model that takes inspiration from the success of ResNet as it was first introduced in computer vision [17, 41]. Extensive evaluation by [20] have demonstrated ResNet is one of the strongest models for TSC. The specific design we use is based on the design proposed by [41].

Transformer: The Transformer is a widely used architecture for sequence modeling [8, 25, 26, 40, 50]. We used fixed positional encoding (following [40]) and included a special token, `[start]`, to learn the representation of the entire time series. Our Transformer architecture consists of four encoder layers, where the number of heads is 8, the input dimension is 64, and the output dimension is 64. The Transformer block is composed of a multihead self-attention stage and a feed-forward stage. Both stages incorporate skip connections, ensuring that the input is added to the output at each stage. We use layer normalization [5] for all normalization layers, as it is both effective and widely used in modeling sequential data [5, 40].

[1] One layer generates μ, while the other produces $\log \sigma^2$.

4 EXPERIMENT

During all training stages, the optimization executes for 400 epochs using the AdamW optimizer [28] with a batch size of 64. Our learning rate scheduler follows the 1cycle learning rate policy [36]. All code, data, full experiment results, details, hyperparameters, and architecture discussion is available on our companion website [1].

Datasets. We use the UCR Archive [12] and the UEA Archive [6] in our experiments.[2] The UCR Archive contains 128 univariate classification datasets, and the UEA Archive contains 30 multivariate classification datasets. For each of the datasets in each archive, we randomly extract the following data splits: pretraining (50%), training (30%), validation (10%), test (10%). All splits are mutually exclusive, and the pretraining set does not contain labels.

Experiment Setup. We perform a four-stage experimental pipeline for each dataset: 1) Pretraining, 2) Fine-Tuning, 3) Validation, and 4) Testing. Methods that do not require pretraining skip Stage 1, and the pretraining set is ignored. In methods that do not use generated time series, we use the pretraining set to pretrain the model. When methods involve a trainable data generator, we use the pretraining set to train the generative model. We use the generator to generate n_{gen} time series to pretrain the model with.

While an unlimited quantity of data can be generated, but remain mindful of computational costs. We consider the average number of time series in each dataset of the source archive when determining n_{gen}.[3] If the size of the pretraining set is below this threshold, we generate the threshold number of time series. Otherwise, we generate the same number of time series in the pretraining set.

Experiment Results. Table 1 are the average ranks of each method in each Archive. We include three baselines: models with no pretraining, 1NN Euclidean Distance (ED), and 1NN Dynamic Time Warping (DTW) distance. The 1NN baselines are considered simple, yet effective for TSC problems [7, 12, 33]. A total of 60 methods and baselines are compared.[4] Table 2 consolidates the results in Table 1, presenting the top-10 methods in each Archive according to their average rank. We observe from Table 2 that:

(1) Most of the top-10 methods are pretrained with generated time series, indicating generators generally improve PTMs.
(2) The TimeCLR, TS2Vec, and MixingUp PTMs demonstrate superior performance over TF-C. The difference in average rank performance between the complex generation methods (GAN, β-VAE, Diff.) and the simple ones (RW, SW, MG) is marginal.

[2] The University of East Anglia (UEA) has contributed several datasets to the UCR Archive, and this archive is often referred to as the "UCR/UEA Archive". We refer to the UCR/UEA Archive as the "UCR Archive" to avoid confusion in our discussion.
[3] For the UCR Archive this threshold is 1494, and for the UEA Archive it is 3398.
[4] *NG* column indicates where "no generator" was used: in cases of pretraining, the pretraining set was used, and where no pretraining was conducted the PTM is "N/A".

Table 1: The values are the average ranks for each configuration of experiments, testing combinations of the backbone models, PTMs, and data generators. Ranks are computed for each archive separately, and are highlighted by row. We compare data generators by fixing the backbone and PTM. The top ranking method is in bold, and the runner-up is underlined.

Archive	Backbone	PTM	NG	RW	SW	MG	GAN	β-VAE	Diff.
UCR	1NN ED	N/A	40.41	-	-	-	-	-	-
	1NN DTW	N/A	32.84	-	-	-	-	-	-
	ResNet	N/A	24.16	-	-	-	-	-	-
		TimeCLR	34.56	23.90	23.27	**22.21**	22.73	22.38	22.38
		TS2Vec	23.44	23.23	21.95	23.23	**21.71**	22.04	22.94
		MixingUp	37.89	22.61	23.47	**21.82**	22.30	22.88	23.72
		TF-C	42.03	42.12	**40.30**	42.77	41.05	41.57	42.31
	Transformer	N/A	32.87	-	-	-	-	-	-
		TimeCLR	32.89	29.91	28.60	28.16	**27.30**	28.58	28.09
		TS2Vec	27.59	29.17	29.70	28.50	29.66	**26.45**	28.27
		MixingUp	29.75	29.54	28.41	28.17	29.41	30.35	**26.73**
		TF-C	42.38	42.03	43.21	42.23	43.80	42.88	**41.16**
UEA	1NN ED	N/A	43.87	-	-	-	-	-	-
	1NN DTW	N/A	36.42	-	-	-	-	-	-
	ResNet	N/A	30.13	-	-	-	-	-	-
		TimeCLR	27.38	25.60	25.57	28.15	**25.08**	28.73	30.87
		TS2Vec	26.75	26.73	26.00	28.82	32.32	25.42	**25.27**
		MixingUp	24.92	29.25	27.07	**22.53**	24.53	27.53	24.75
		TF-C	**38.92**	41.43	45.12	43.10	41.02	44.93	41.57
	Transformer	N/A	26.92	-	-	-	-	-	-
		TimeCLR	29.43	**26.22**	27.58	29.68	28.30	28.72	27.67
		TS2Vec	27.57	27.63	29.58	29.43	28.30	29.97	**25.43**
		MixingUp	30.35	26.95	29.12	26.90	26.72	26.98	**26.03**
		TF-C	33.88	34.13	32.75	38.25	**32.52**	34.67	38.55

Table 2: The 10 best methods on the UCR Archive and UEA Archive classification tasks.

	UCR Archive	UEA Archive
1	ResNet+TS2Vec+GAN	ResNet+MixingUp+MG
2	ResNet+MixingUp+MG	ResNet+MixingUp+GAN
3	ResNet+TS2Vec+SW	ResNet+MixingUp+Diff
4	ResNet+TS2Vec+β-VAE	ResNet+MixingUp+NG
5	ResNet+TimeCLR+MG	ResNet+TimeCLR+GAN
6	ResNet+MixingUp+GAN	ResNet+TS2Vec+Diff
7	ResNet+TimeCLR+Diff.	ResNet+TS2Vec+β-VAE
8	ResNet+TimeCLR+β-VAE	Transformer+TS2Vec+Diff
9	ResNet+MixingUp+RW	ResNet+TimeCLR+SW
10	ResNet+TimeCLR+GAN	ResNet+TimeCLR+RW

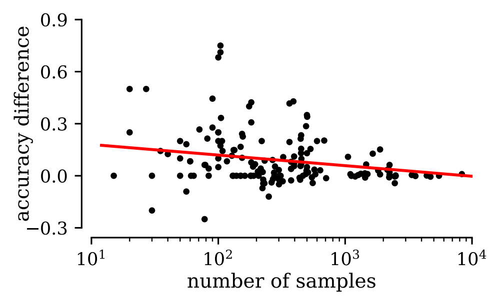

Figure 3: The x-axis is the size of allocated pretraining set and the y-axis the difference in accuracy of the two models. Positive values indicate where the MG variant outperforms the NG variant. The fitted line (red) shows the inverse correlation between dataset size and the performance gain with the data generator. Each point is a dataset from either Archive.

(3) In Table 1, we observe in 10/16 cases that pretraining models with real data may have led to degradation in performance. This may indicate the negative impact stems from data scarcity, and may be alleviated using generative models.

(4) Most of the top-10 methods utilize the ResNet backbone.

We examine the correlation between the size of the pretraining set and the relative accuracy between methods pretrained with generated data and those pretrained with real data. We are interested in ResNet+MixingUp+MG, as it ranks highly (2nd in the UCR Archive and 1st in the UEA Archive), and plot the accuracy difference between ResNet+MixingUp+**MG** and ResNet+MixingUp+**NG** in Figure 3. Overall, there is a trend indicating the use of a data generator is beneficial, but has diminishing returns with an increase in pretraining set size. Thus, the adoption of data generators *does* help alleviate the issue of data scarcity.

5 CONCLUSION

We explore improving the performance of pretraining methods by utilizing time series generative models. By generating vast quantities of time series data for pretraining, we can improve the classification accuracy on the both UCR Archive and UEA Archive by addressing the data scarcity issue. We consider different types of generative models used in the pretraining, and have included a systematic study of combining different data generators and pretrained methods. Future work includes improvements to pretraining through superior generative models, or an ensemble of such models. Another area of exploration is the feasibility of constructing a universal time series generator for pretraining, which encapsulates some commonnality in distributions across time series domains.

REFERENCES

- [1] 2023. Paper Companion Website (Code, Data, Documentation). https://sites.google.com/view/tsgenandptm/
- [2] Sabeen Ahmed, Ian E Nielsen, Aakash Tripathi, Shamoon Siddiqui, Ravi P Ramachandran, and Ghulam Rasool. 2023. Transformers in time-series analysis: A tutorial. *Circuits, Systems, and Signal Processing* (2023), 1–34.
- [3] Juan Miguel Lopez Alcaraz and Nils Strodthoff. 2023. Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models. *arXiv preprint* (2023).
- [4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 214–223.
- [5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint* (2016).
- [6] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. 2018. The UEA multivariate time series classification archive, 2018. *arXiv preprint* (2018).
- [7] 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.
- [8] 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.
- [9] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. 2020. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint* (2020).
- [10] 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.
- [11] Contributions. 2023. GitHub repository for Self-Supervised Contrastive Pre-Training For Time Series via Time-Frequency Consistency. https://github.com/mims-harvard/TFC-pretraining.
- [12] 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.
- [13] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2016. Adversarial feature learning. *arXiv preprint* (2016).
- [14] Yifan Guo, Weixian Liao, Qianlong Wang, Lixing Yu, Tianxi Ji, and Pan Li. 2018. Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach. In *Asian Conference on Machine Learning*. PMLR, 97–112.
- [15] Priyanka Gupta, Pankaj Malhotra, Jyoti Narwariya, Lovekesh Vig, and Gautam Shroff. 2020. Transfer learning for clinical time series analysis using deep neural networks. *Journal of Healthcare Informatics Research* 4, 2 (2020), 112–137.
- [16] Samuel Harford, Fazle Karim, and Houshang Darabi. 2021. Generating adversarial samples on multivariate time series using variational autoencoders. *IEEE/CAA Journal of Automatica Sinica* 8, 9 (2021), 1523–1538.
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.
- [18] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*.
- [19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.
- [20] 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.
- [21] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.
- [22] Brian Kenji Iwana and Seiichi Uchida. 2020. Time series classification using local distance-based features in multi-modal fusion networks. *Pattern Recognition* 97 (2020), 107024.
- [23] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint* (2013).
- [24] Junying Li, Weijie Ren, and Min Han. 2021. Variational auto-encoders based on the shift correction for imputation of specific missing in multivariate time series. *Measurement* 186 (2021), 110055.
- [25] 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).
- [26] 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.
- [27] Lequan Lin, Zhengkun Li, Ruikun Li, Xuliang Li, and Junbin Gao. 2023. Diffusion models for time series applications: A survey. *arXiv preprint* (2023).
- [28] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint* (2017).
- [29] Qianli Ma, Zhen Liu, Zhenjing Zheng, Ziyang Huang, Siying Zhu, Zhongzhong Yu, and James T Kwok. 2023. A Survey on Time-Series Pre-Trained Models. *arXiv preprint* (2023).
- [30] Pedro Matias, Duarte Folgado, Hugo Gamboa, and André V Carreiro. 2021. Robust Anomaly Detection in Time Series through Variational AutoEncoders and a Local Similarity Score.. In *Biosignals*. 91–102.
- [31] Amiel Meiseles and Lior Rokach. 2020. Source model selection for deep learning in the time series domain. *IEEE Access* 8 (2020), 6190–6200.
- [32] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* 63, 10 (2020), 1872–1897.
- [33] 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 *ACM SIGKDD*. 262–270.
- [34] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. 2021. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*. PMLR, 8857–8868.
- [35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*.
- [36] Leslie N Smith and Nicholay Topin. 2019. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, Vol. 11006. SPIE, 369–386.
- [37] Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I Webb. 2022. MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery* 36, 5 (2022), 1623–1646.
- [38] Chi Ian Tang, Ignacio Perez-Pozuelo, Dimitris Spathis, and Cecilia Mascolo. 2020. Exploring contrastive learning in human activity recognition for healthcare. *arXiv preprint* (2020).
- [39] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems* 34 (2021), 24804–24816.
- [40] 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).
- [41] 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.
- [42] 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* (2020).
- [43] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2022. Transformers in time series: A survey. *arXiv preprint* (2022).
- [44] 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.
- [45] Chin-Chia Michael Yeh, Xin Dai, Huiyuan Chen, Yan Zheng, Yujie Fan, Audrey Der, Vivian Lai, Zhongfang Zhuang, Wang Junpeng, Liang Wang, and Wei Zhang. 2023. Toward a Foundation Model for Time Series Data. In *Proceedings of the 32nd ACM International Conference on Information & Knowledge Management*.
- [46] Yuan Yuan and Lei Lin. 2020. Self-supervised pretraining of transformers for satellite image time series classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2020), 474–487.
- [47] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2022. Ts2vec: Towards universal representation of time series. In *AAAI*, Vol. 36. 8980–8987.
- [48] Da Zhang, Ming Ma, and Likun Xia. 2022. A comprehensive review on GANs for time-series signals. *Neural Computing and Applications* 34, 5 (2022), 3551–3571.
- [49] Xiang Zhang, Ziyuan Zhao, Theodoros Tsiligkaridis, and Marinka Zitnik. 2022. Self-supervised contrastive pre-training for time series via time-frequency consistency. *arXiv preprint* (2022).
- [50] 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 *AAAI*, Vol. 35. 11106–11115.