Neural Architecture Search for Self-Supervised Representation Learning on Time-Series Data

Seoyoung Kim[1] , **Doguk Kim**[2]
[1]Inha University , [2]Inha University
kimsy9365@naver.com , dgkim@inha.ac.kr

Abstract

Self-supervised representation learning on time-series aims to extract informative representations of raw time-series for various downstream tasks. It is general to use a standard model architecture, but since the dynamics of different time-series vary greatly, the optimal architecture of the time-series representation learning model could also be different depending on the datasets. With the goal of finding the architecture optimal for the dynamics of each time-series, as the first attempt, we propose a neural architecture search (NAS) for self-supervised representation learning on time-series data. At first, we attempted to utilize some existing NAS methods in optimizing the architecture of the time-series representation learning model. However, despite their high performance in the image domain, the results showed their ineffectiveness when applied to this field. Among various potential reasons for this phenomenon, we sought the cause of their ineffectiveness in the scheme of bi-level optimization where DARTS has been constructed on and vulnerability to the overfitting issue of this field. Then, we suggest a new search method that operates effectively in this field. Specifically, our search method freezes the model weights just before they lose generalization and helps architecture parameters can be optimized on the basis of generalized model weights. Extensive experiments on a variety of time-series representation learning models, datasets, and downstream tasks validate the effectiveness of our method and its suitability to this field.

1 Introduction

Since time-series data can include sequential information of various phenomena, they are widely used across a lot of application domains, such as financial markets, climate modeling, and demand forecasting. Due to the ubiquity of time-series, there have been continuous attempts on how to ‘represent’ raw time-series data capturing more valuable underlying information. According to this demand, self-supervised representation learning on time-series data has become in the spotlight recently.

Self-supervised representation learning enables the model to extract informative representations without labels. Traditionally, self-supervised learning has been conducted with hand-crafted self-supervised tasks like rotation prediction [Komodakis and Gidaris, 2018], image colorization [Zhang *et al.*, 2016], and so on. SimCLR [Chen *et al.*, 2020a] proposed the concept of contrastive learning that can learn representations without hand-crafted self-supervised tasks by learning representations of given inputs using data augmentation. With the emergence of various methods that surpass the SimCLR, self-supervised representation learning has been successfully applied in various fields such as computer vision, natural language processing, voice recognition, and time-series forecasting. Especially, self-supervised representation learning on time-series data has been recently proposed and achieved SOTA performances on various benchmarks [Yue *et al.*, 2022; Zhang *et al.*, 2022; Yang and Hong, 2022]. TS2Vec [Yue *et al.*, 2022] and TS-TCC [Tonekaboni *et al.*, 2021], self-supervised representation learning techniques for time series data, perform contrastive learning utilizing data augmentation suitable for time-series data. In particular, TS2Vec conducted an ablation study on various factors such as the architecture of neural networks, data augmentation methods, and hyperparameter changes. Among these factors, it was observed that the performance changed most dramatically when the architecture of neural networks for learning the time series representation was altered. This implies that the architecture of neural networks has a significant impact on the performance of self-supervised representation learning on time series data.

Neural Architecture Search (NAS) has contributed to the enhancement of extensive research fields such as computer vision and natural language processing by enabling designing optimal neural networks automatically. However, to the best of our knowledge, there has been no research on the relationship between self-supervised representation learning on time series data and the architecture of neural networks up to date. But if we could find representation learning model architecture optimal for data-specific dynamics via NAS, we inferred that we could extract more informative representations of raw time-series. Therefore, this paper focuses on finding an optimal neural architecture for self-supervised representation learning based on NAS and improving its performance on various downstream tasks.

Among various popular NAS techniques, Differentiable Architecture Search (DARTS), drawing attention by achieving unprecedented efficiency, has been a root for various popular search methods [Zela *et al.*, 2019; Zhang *et al.*, 2020; Zhao *et al.*, 2021; Ye *et al.*, 2022; Hu *et al.*, 2022]. Specifically, DARTS relaxed the discrete neural architecture search problem to a continuous one by viewing it as an optimization problem of architectural parameters which parametrize the architecture of a neural network. Noting to its efficiency, we applied multiple effective derivatives of DARTS to several self-supervised representation learning models on time-series data. Mysteriously, most of them failed to find an optimal architecture having better test performance than the backbone architecture. Among several potential reasons, we sought the reason for the failure of existing NAS in (1) the bi-level optimisation scheme on which DARTS has been constructed and (2) the unique feature of self-supervised representation learning on time-series data.

The general form of the bi-level optimization problem is as follows:

Definition 1. *Given the outer objective function* $\mathcal{L}_{valid} : \mathbb{R}^T \times \mathbb{R}^V \rightarrow \mathbb{R}$*, the inner objective function* $\mathcal{L}_{train} : \mathbb{R}^T \times \mathbb{R}^V \rightarrow \mathbb{R}$*, the outer variable* $\alpha \in \mathbb{R}^V$*, and the inner variable* $w \in \mathbb{R}^T$*,*

$$\underset{\alpha \in \mathbb{R}^V}{min} \mathcal{L}_{valid}(\alpha, w^*(\alpha))$$

$$s.t. w^*(\alpha) \in \underset{w \in \mathbb{R}^T}{argmin} \mathcal{L}_{train}(\alpha, w)$$

DARTS sets an *one-shot model* where all the possible candidate architectures are contained sharing their weights. Ideally, according to the bi-level optimization scheme, DARTS aims to optimize architectural parameters(α) under the one-shot model weights(w) having optimal values that minimize the training loss. However due to expensive inner optimization, DARTS approximates $w^*(\alpha)$ by updating w with a single training step when evaluating the gradient of architectural parameters. Hence, DARTS updates w and α alternately by one step at a time. In this point, we assumed that if the one-shot model is vulnerable to overfitting, in other words, learns the noise of the training dataset easily even in short training time, the model weights(w) would be overfitted after several training steps and the architectural parameters(α) would start to be optimized on the model weights(the inner variable) overfitted to the training dataset. Also, iterative updates of architectural parameters reflecting how each candidate operation contributes to validation loss at each step would lose credibility if w loses generalization (the ability of the model not overfitted to training dataset and operate well both on training dataset and unseen datasets). According to our experimental results (Figure 1), the self-supervised representation learning model on time-series data is observed to easily overfit even during the training epochs pre-set at original papers [Yue *et al.*, 2022; Eldele *et al.*, 2021]. Generally, the overfitting problem is resolved by early stopping based on model performance on the validation dataset, but it is not appropriate in this situation since validation loss continuously decreases during the optimization of architectural parameters on the validation dataset.

Fortunately, there have been extensive researches on the positive correlation between the sharpness of local minimum and model generalization [Keskar *et al.*, 2016; Dziugaite and Roy, 2017; Foret *et al.*, 2020]. Motivated by these previous studies, we propose a new search method suitable for self-supervised representation learning on time-series data, Sharpness-aware Weight Freezing DARTS (DARTS-SWF) which guarantees the optimization of architecture conducted under generalized model weights. In this paper, we validate the effectiveness and suitability of our new search method on self-supervised representation learning on time-series data with various self-supervised representation learning models, datasets, and downstream tasks. Our contributions are summarized as:

- We propose a neural architecture search for self-supervised representation learning on time series data, and to the best of our knowledge, this is the first attempt of its kind.
- We found out that the commonly used DAG search space is not appropriate for self-supervised representation learning on time-series, and suggest a new search space suitable for this field.
- We experimentally discovered the unsuitability of representative NAS methods originally designed for image data when applied to time-series data, and propose a new NAS method that continues the optimization of architectural parameters with the model weights loaded just before loss of model generalization. Also, we validate its effectiveness and suitability for self-supervised representation learning on time-series across a variety of models, datasets, and downstream tasks.

2 Related Works

2.1 Self-supervised representation learning on time series data

Early studies in representation learning on time series focused on how to fully contain complex latent relationships in time series data that cannot be delivered well by one-hot encoding. As examples of the attempt, distributed representations with skip-gram [Choi *et al.*, 2016] and encoder-decoder architecture [Malhotra *et al.*, 2017] were suggested.

With the advancement of contrastive learning that learns representations by enforcing consistency between positive samples and heterogeneity between negative samples, a variety of works about contrastive learning on time series data have been proposed [Sermanet *et al.*, 2018; Hyvarinen and Morioka, 2016; Tonekaboni *et al.*, 2021]. Especially, [Yue *et al.*, 2022] presented a new framework named TS2Vec that encourages the same timestamps in two different augmented views as positive samples, and [Eldele *et al.*, 2021] proposed a framework named TS-TCC that encourages consistency of augmented view Furthermore, [Zhang *et al.*, 2022] enlarged the embedding space adding frequency-based embedding and introduced a strategy enforcing consistency of time-based and frequency-based representations of the same sample.

2.2 Neural Architecture Search

Neural Architecture Search (NAS) is a technique that aims to find an optimized neural network architecture for given datasets and tasks. NAS can be broadly divided into three kinds of approaches depending on the search methods; reinforcement learning (RL)-based methods [Lee *et al.*, 2020; Cai *et al.*, 2019], evolutionary algorithm (EA)-based methods [Xie and Yuille, 2017; Real *et al.*, 2017; Real *et al.*, 2019], and gradient-based methods [Liu *et al.*, 2018; Luo *et al.*, 2018; Cai *et al.*, 2019; Ye *et al.*, 2022]. RL-based methods are the earliest proposed NAS technique, which searches for the optimal neural architecture within the search space based on reinforcement learning and appropriate rewards for given tasks. They have suffered from the huge search cost to cover every possible candidate architecture. Recently, CNAS [Guo *et al.*, 2020] applied a curriculum search method to a search space and improved both the search efficiency and performance. EA-based methods use evolutionary algorithms to search optimal neural architecture, generally showing high performance but with the drawback of high computational costs. On the other hand, gradient-based methods convert the neural architecture search problem which is inherently discrete into a continuous optimization task and apply gradient-based optimization techniques. Various gradient-based approaches have also been proposed as follow-up studies of DARTS [Liu *et al.*, 2018] after DARTS got attention for its intuitive algorithm and high performance [Zela *et al.*, 2019; Zhang *et al.*, 2020; Zhao *et al.*, 2021; Ye *et al.*, 2022; Hu *et al.*, 2022]. In particular, β-DARTS [Ye *et al.*, 2022] has dramatically improved performance by simply adding beta-decay regularization to the existing DARTS. DARTS-ES [Zela *et al.*, 2019] has applied early stopping methods on DARTS when confronted dramatic increase of the largest eigenvalue of the hessian matrix of validation loss. Also, there have been observed several empirical biases with inaccurate architectural optimization and poor generalization ability due to coupled learning between sub-models [Zela *et al.*, 2019; Chen *et al.*, 2020b]. As an attempt to resolve this issue, Few-shot NAS [Zhao *et al.*, 2021] successfully improved the performance of One-Shot NAS by applying an edge-wise exhaustive partitioning schema on the One-Shot supernet and GM-NAS [Hu *et al.*, 2022] proposed effective splitting decision utilizing Gradient-Matching(GM) score on the aforementioned algorithm.

There have been several studies on NAS for time-series data. [Chen *et al.*, 2021] aimed to optimize neural network architecture for multivariate time series forecasting based on a scale-aware NAS approach. [Rakhshani *et al.*, 2020] attempted to apply NAS to time-series classification, and the proposed model called NAS-T successfully improved performance on various datasets. While these works focus on NAS for time-series classification, [Risso *et al.*, 2022] utilized NAS to enhance the performance of Temporal Convolutional Networks (TCN), which are widely used to process time-series data for various tasks. Although there are existing works on NAS for time-series data, none of them focuses on conducting NAS on self-supervised representation learning. Hence, we would like to propose NAS on self-supervised representation learning for extracting better representations on raw time-series for the first time.

3 Methodology

This paper aims to find the optimal neural network architecture for self-supervised representation learning on time-series in a dataset-specific manner via NAS. However, the experiments described in Section 4.4 imply that the DAG-based search space commonly used in the image domain for NAS is not suitable for this field, and a new search space should be designed for effective performance gain via NAS on this field. The detailed description for DAG-based search space we utilized is indicated in Appendix 3 and [Liu *et al.*, 2018]. Also, the experiments described in Table 2 suggest that new search method is needed. Hence, we propose a new NAS search method and search space for self-supervised representation learning on time-series data. As representative application cases of the proposed NAS method and search space, we applied them to the encoder of TS2Vec model [Yue *et al.*, 2022] and TS-TCC model [Eldele *et al.*, 2021], and explored their best architectures. In this section, we introduce the mechanism of our new search method and the structure of our new search space.

3.1 Preliminary

TS2Vec and TS-TCC

Contrastive learning, the basic idea of TS2Vec and TS-TCC, encourages maximizing the similarity between different views of one sample while minimizing the similarity between the views of different samples. TS2Vec is a self-supervised representation learning model on time-series that can extract contextual representations of arbitrary sub-series in all semantic levels. Given N time series $\{x_1, x_2, \cdots, x_N\}$ of N different instances, the encoder of TS2Vec aims to learn a nonlinear embedding function f_θ that maps a time series $x_i \in \mathbb{R}^{T \times F}$ to its timestamp-level representation $r_i \in \mathbb{R}^{T \times D}$, where T denotes temporal dimension and F and D represent feature dimensions. In order to train the encoder, TS2Vec encodes randomly sampled two segments, randomly masks the two representations, and performs contrastive learning for the common segment. Specifically, the contrastive learning in TS2Vec encourages *contextual consistency* which means consistency between the same timestamps of two augmented views. Detailed mechanisms and learning objectives of TS2Vec are described in Appendix 2.1 and [Yue *et al.*, 2022].

Similarly, TS-TCC is a self-supervised representation learning model on time-series data which is learned via contrastive learning. The encoder of TS-TCC aims to extract high dimensional latent representations $z \in \mathbb{R}^{T' \times F}$ from $x \in \mathbb{R}^{T \times d}$. The training process of TS-TCC is as follows. Firstly, TS-TCC extracts strongly augmented view x^s and weekly augmented view x^w from input time-series x, and encodes each of them as $z^s, z^w \in \mathbb{R}^d$. Then, the two representations are fed to temporal contrasting module, where each $z^s_{\leq t}, z^w_{\leq t} \in \mathbb{R}^d$ is summarized into a corresponding context vector $c^s_t, c^w_t \in \mathbb{R}^h$ by an autoregressive model, and c^s_t, c^w_t are forced to predict the future timestamps of the weak representation $z^w_{t+k}, z^s_{t+k} (1 \leq K)$ respectively by log-bilinear

Model	Structure
TS2Vec	Input FC $\rightarrow TimestampMaskingModule \rightarrow Res[\{GELU \rightarrow$ Conv(k=3, s=1, d=$2^{(l-1)}$, p=$2\times d+1$)$\} \times 2] \times 11$ (Channel) $C_{in} \rightarrow 64 \rightarrow 64 \rightarrow \cdots \rightarrow 64 \rightarrow 64 \rightarrow 320$
TS-TCC	[Conv(k=8, s=1, p=4) $\rightarrow BN \rightarrow ReLU \rightarrow MaxPool] \times 3$ (Channel) $C_{in} \rightarrow 32 \rightarrow 64 \rightarrow 128$

Table 1: Backbone Architectures of Encoders of TS2Vec and TS-TCC. l denotes the position number of the corresponding residual block.

model. Detailed mechanisms and learning objectives of TS-TCC are described in Appendix 2.2 and [Eldele *et al.*, 2021].

The architectures of the encoders of TS2Vec and TS-TCC are described in Table 1. We set the two encoders as architecture searching parts where the other parts of the models are maintained the same.

Relationship between sharp/flat minima and model generalization

According to [Hochreiter and Schmidhuber, 1997], a "flat minimum" is a large connected region in weight space where the error remains approximately constant. With the help of the MDL-based bayesian argument, they indicated that flat minima correspond to a low possibility of overfitting. After their research, extensive studies have analyzed the correlation under consideration of various elements including batch size [Keskar *et al.*, 2016]. Furthermore, several metrics which enable to estimate the sharpness have been proposed [Jastrzebski *et al.*, 2017; Keskar *et al.*, 2016; Wen *et al.*, 2019] (representatively, largeness of λ_{max} of hessian matrix in $\mathcal{L}_{train}$), and an optimization algorithm searching for flat minima also have been proposed for enhancement of model generalization [Foret *et al.*, 2020]. Motivated by the aforementioned research, we selected the sharpness of minima in training loss as a method to estimate whether the inner variables in the bi-level optimization paradigm are overfitted or not.

3.2 Search Method

In this section, we introduce a detailed mechanism of our new search method. We were motivated by the following four points. **(a)** If the continuous updates of the inner variables(model weights) make the inner variables overfitted after a certain point, then the outer variables(architecture parameters) become optimized on the basis of abnormally acting model weights after that point(Definition 1) **(b)** [Zela *et al.*, 2019] assumed that sharp minimum in $\mathcal{L}_{valid}$ leads to a large discrepancy between the one-shot model and the discretized architecture, namely overfitting of architecture parameters, which results in poor generalization. Then, they suggested the DARTS-ES method where the alternate learning of model weights and architecture parameters are all stopped when the average of the largest eigenvalue of $\nabla^2_\alpha \mathcal{L}_{valid}$ increases radically. But from the poor results of applying DARTS-ES to time-series representation learning (Table 2), we inferred that early stopping both the learning of model weights and architectural parameters at once is not effective in this field. Especially, **(c)** from several test performance-number of training iterations graphs, we could find out that time-series representation learning models are vulnerable to overfitting. We

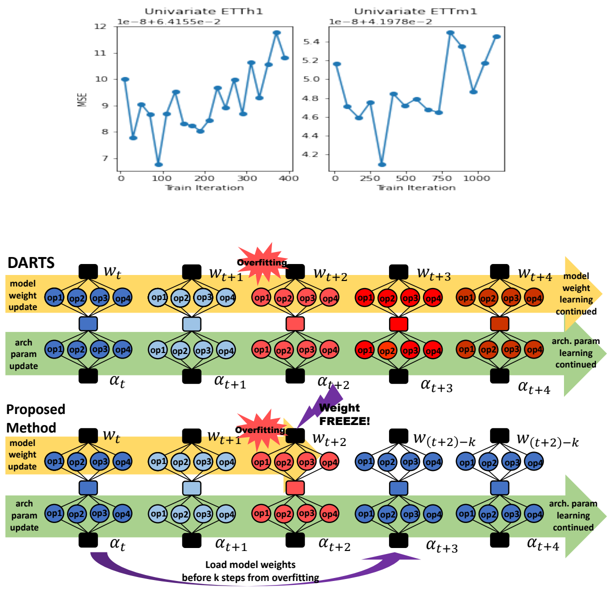

Figure 1: Test performance(MSE) log of TS2Vec along uniformly increasing train iterations(top). Overall Description of DARTS-SWF(bottom)

trained TS2Vec on univariate ETTh1 and ETTm1 datasets on 5 random seeds each for 400 and 1200 iterations which are double the number of official training iterations in the original paper [Yue *et al.*, 2022], and we could observe the overfitting occurred even before the official number of training iterations(Figure 1). Therefore, we thought that separated early stopping for model weights training and architectural parameters training is essential and consisted our new search method in the way that the model weights are frozen just before they overfit to train dataset, and the architecture parameters can be learned solely on the frozen model weights which generalize well to other datasets (does not overfit to train dataset). **(d)** We could get the idea about freezing moment from previous research on the relation between the flatness of loss function in weight space and model generalization. Consequently, we propose a new search method that would retain the generalized weight in bi-level optimization even if the architectural parameters are continuously updated.

The overall algorithm of our newly suggested search method is described in Figure 1 and Algorithm 1. Initially, we start the architecture search with alternate optimizations and computing the local average of maximum eigenvalue of $\nabla^2_w \mathcal{L}_{train}$ ($\bar{\lambda}_w$) per iteration. When $\bar{\lambda}^w_t$ / $\bar{\lambda}^w_{(t-1)}$ increases radically, the model weights are frozen as the weight values before m steps, which is a predefined value, and model weights training is terminated but architecture parameters training is continued. Then, the ratio of the local average of maximum eigenvalue of $\nabla^2_\alpha \mathcal{L}_{valid}$ ($\bar{\lambda}_\alpha$) is tracked. Following DARTS-ES, if $\bar{\lambda}^\alpha_t / \bar{\lambda}^\alpha_{(t-1)}$ jumps radically, then the training process of architecture parameters is also stopped.

3.3 Search Space

In this section, we introduce our new search space suitable for NAS for self-supervised representation learning

Algorithm 1 DARTS-SWF

```
Require: supernet model weights $w$, architecture parameters $\alpha$, number of maximum iterations T, weight-freezing flag $F$, predefined arch ratio $k_\alpha$, predefined weight ratio $k_w$, size of local average window $n$, size of weight-loading step $m$
 1: for $t = 1...T$ do
 2:   if not F then:
 3:     $\bar{\lambda}^w_{bef} \leftarrow \bar{\lambda}^w_{cur}$
 4:     Compute $\bar{\lambda}^w_{cur}$ as maximum eigenvalue of $\nabla^2_w \mathcal{L}_{train}$
 5:     Compute $\bar{\lambda}^w_{cur}$ as a local average of maximum eigenvalues of $\nabla^2_w \mathcal{L}_{train}$ at $(t-n), ..., t$
 6:     if $\bar{\lambda}^w_{cur} / \bar{\lambda}^w_{bef} > k_w$ then:
 7:       F ← True
 8:       Save model weights at $t = t - m$ as w'
 9:     else:
10:       Update $w$ by descending $\nabla_w \mathcal{L}_{train}(w, \alpha)$
11:     end if
12:   else:
13:     $\bar{\lambda}^\alpha_{bef} \leftarrow \bar{\lambda}^\alpha_{cur}$
14:     Compute maximum eigenvalue of $\nabla^2_\alpha \mathcal{L}_{valid}$
15:     Compute $\bar{\lambda}^\alpha_{cur}$ averaging maximum eigenvalues of $\nabla^2_\alpha \mathcal{L}_{valid}$ at $(t-n), ..., t$
16:     if $\bar{\lambda}^\alpha_{cur} / \bar{\lambda}^\alpha_{bef} > k_\alpha$ then:
17:       Terminate architecture search
18:     else:
19:       Update $\alpha$ by descending $\nabla_\alpha \mathcal{L}_{valid}(w', \alpha)$
20:     end if
```

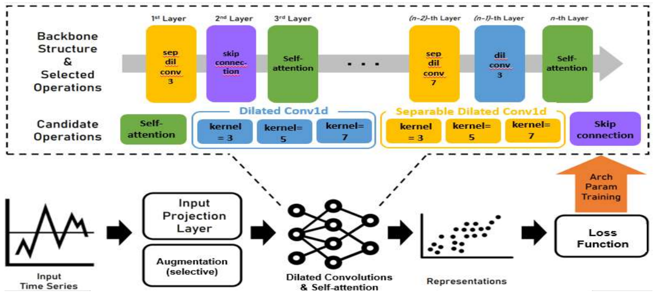

Figure 2: Overall Description of our search space

on time-series data. At first, we tried to utilize a well-known search space originally designed for NAS on computer vision tasks, a stacked DAG-based cell structure, but we observed poor results when performing NAS with the DAG-based search space and our newly proposed search method(Table 5). Thus, throughout several trials transforming the stacked DAG-based cell structure, we succeeded in creating a new search space appropriate for time-series representation learning. Motivations and a detailed derivation process are described in Section 4.4. Figure 2 describes the overall configuration of our proposed search space.

According to the results in Section 4.4, we composed the backbone of our search space as a straight structure where none of the multiple input elements is entered into every choice block and each choice block has its own freedom of operation choice. Firstly, the input time-series passes through the input projection layer consists of a fully-connected layer, and augmentation strategies are applied to the output of input projection layer selectively. In our application cases, we maintained the augmentation modules of TS2Vec and TS-TCC. Then, it goes through l layers of choice blocks each containing 8 candidate operations. Each choice block delivers the weighted sum of 8 outputs each from one of the candidate operations. The weight assigned to each operation when they are summed together is each operation's corresponding architecture parameter.

Considering the importance of dilated convolutions mentioned in [Yue *et al.*, 2022] and the complementary effect of self-attention and convolution operations in time series [Li *et al.*, 2019], we designed our candidate operation set as follows. Each choice block contains 8 candidate operations: (a) three types of residual blocks, each including two 1-D dilated convolution layers with kernel sizes 3 or 5 or 7; (b) three types of residual blocks, each with two 1-D separable dilated convolution layers with kernel sizes of 3 or 5 or 7; (c) self-attention module; (d) a skip-connection block. We included the dilated parameter D^l (for l-th layer) for all the convolution operations noting its verified ability to secure large receptive field in [Yue *et al.*, 2022]. We verified the effectiveness of the structure and detailed candidate operations of our search space in Section 4.4.

4 Experiments

4.1 Experimental Setup

In this section, we evaluate the proposed method by applying our DARTS-SWF and other baseline search methods to TS2Vec and TS-TCC, and comparing the test performances of the searched architectures on a variety of datasets and downstream tasks. Specifically, following the original papers of the two models [Yue *et al.*, 2022; Eldele *et al.*, 2021], we evaluated search methods applied to TS2Vec through time-series classification, time-series forecasting, time-series anomaly detection, and the search methods applied to TS-TCC through time-series classification task. In case of application on TS2Vec, following [Yue *et al.*, 2022], we adopted three ETT datasets [Zhou *et al.*, 2021] for forecasting task, 128 UCR archive [Dau *et al.*, 2019] and 29 UEA archive [Bagnall *et al.*, 2018] for classification task, and KPI dataset [Ren *et al.*, 2019] for anomaly detection task. In the case of application on TS-TCC, following [Eldele *et al.*, 2021], we selected SleepEDF [Goldberger *et al.*, 2000] dataset and HAR dataset [Anguita *et al.*, 2013] for the classification task. We could not get experimental results on *MotorImagery* in the UEA archive, *Yahoo* [Laptev *et al.*, 2015], and *Epilepsy* [Andrzejak *et al.*, 2001] due to the memory limit of our GPU. For our baseline search methods, we selected DARTS [Liu *et al.*, 2018], β-DARTS [Ye *et al.*, 2022], DARTS-ES [Zela *et al.*, 2019] as examples of one-shot gradient-based NAS methods, GM-DARTS [Hu *et al.*, 2022] as an example of few-shot gradient-based NAS method, and CNAS [Guo *et al.*, 2020] as an example of RL-based NAS method. Also, we added the performance of the original backbone architecture. It is well-known that large size of a model has positive effect on its performance. Thus, to exclude its effects and maintain a similar model size to the original backbone architecture, we limited the number of layers to 3 in TS2Vec and 2 in TS-TCC considering the original backbone structure(Table 1). Except for the encoder, we maintained the rest of the parts of models same as the backbones.

The overall organization of our evaluation is as follows. In the first part, we searched the optimal architecture for each dataset on our search space and evaluated a trained encoder composed of that architecture on the same dataset. In the second part, we conducted several experiments to test the transferability of searched architectures in different tasks and datasets. Since labels and downstream tasks are assumed to be unknown in self-supervised representation learning, tuning hyperparameters such as learning epochs is challenging [Yue *et al.*, 2022]. Thus, for our representation learning models, we utilized fixed values for hyperparameters empirically without considering downstream tasks or labels. Detailed values of hyperparameters are described in Appendix 4.3. All the experiments were conducted using NVIDIA RTX 6000.

4.2 NAS on self-supervised representation learning

To appraise only the effectiveness of architectural optimization, we followed the same evaluation process of TS2Vec and TS-TCC. Specifically, in the case of time-series forecasting, the future H observations $x_{t+1}, \dots, x_{t+T}$ are predicted by a linear regression model on top of the encoder with an L_2 norm penalty, using the representation of the last timestamp t as input, and then MSE and MAE are calculated for 5 different prediction lengths. In case of time-series classification used in TS2Vec, max pooling is applied along the temporal axis on each representation delivered from the encoder, and the subsequent instance-level representations are classified by a trained SVM classifier. For the anomaly detection task, the anomaly score is defined from L_1 distance between the representations of the subject timestamp extracted from masked and unmasked time series, and then the F1 score, precision, and recall are measured. For the time-series classification in TS-TCC, the encoder is evaluated as the performance of a linear classifier on top of the frozen pre-trained encoder.

We applied these unified evaluation processes to the architectures searched by a variety of baseline search methods and our search method, and the subsequent results are described in Table 2. All the experiments in Table 2 are conducted for three different random seeds and the mean value for each metric is specified in Table 2. Specifically, in Table 2 the accuracy values of UCR and UEA datasets are the average of accuracy values obtained from 128 UCR datasets and 29 UEA datasets. The indicated MSE values for time-series forecasting tasks are the average of five MSE values obtained from 5 experiments with different prediction length conditions. We added the detailed experimental settings and results including the standard deviation of results and individual results from the unit dataset in UCR and UEA archives in Appendix 4 and 5. According to the Table 2, our search method achieved the best performance on most of the datasets and downstream tasks except for time-series classification on the sleepEDF dataset, precision in anomaly detection, and multivariate time-series forecasting on the ETTm1 dataset. However, our search method got the third or second best score for the three tasks.

Also, we conducted the Nemenyi test to inspect whether there exist statistically significant differences between the performances of model architectures each searched by our search method and one of the baseline search methods. In the case of classification tasks, we considered a mean value of three accuracies on a dataset in UCR or UEA archive measured at 3 random seeds as a single sample and conducted the Nemenyi test for 157 samples for every possible pair (DARTS-SWF, one of the baseline methods). The subsequent p-values for all the possible pairs were analyzed as Figure 3, which implied the statistically significant difference between the performances of model architectures searched by all the baseline search methods and our search method. We also conducted the same test on other datasets and the results are added in Appendix 1.

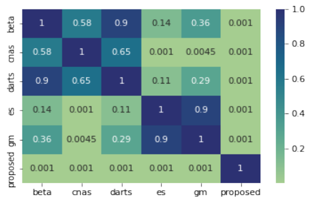

Figure 3: Heatmap of p-values from Nemenyi test on 128 UCR and 29 UEA datasets

4.3 Architectural Transferability Analysis

In this section, we conducted extensive experiments to estimate architectural transferability when the dataset for architecture search and the dataset for training and evaluating the searched model are different. Especially, we aimed to inspect the architectural transferability of our search method when the downstream tasks for the aforementioned two datasets are different. Thus, we selected one dataset among the datasets which achieved better performance than all the other search methods and backbone for three downstream tasks, and measured the performance of the architecture searched in one dataset when transferring it to the other two datasets and downstream tasks on a single random seed. The evaluation results for each (target task, architectural task) pair are described in Table 3. The tasks listed on the vertical axis('Target Task') denote the new downstream tasks where the test performance of transferred architecture is evaluated, and the ones listed on the horizontal axis denote the downstream task that the dataset where the architecture was searched belongs to. The three datasets were selected as AllGestureWiimoteY, univariate ETTh1, and KPI. We observed consistent performance gains for all three datasets and corresponding downstream tasks when the encoder has the architecture searched on the AllGestureWiimoteY dataset. The architectures searched each on univariate ETTh1 and KPI dataset also got improvement compared to the original TS2Vec model in one of the two transferred downstream tasks.

4.4 Discussions

Observations on Various Hyperparameters

In this section, we analyzed the influences of the value of weight-freezing threshold k_w and the size of the averaging window on λ^w. We measured the test performances of the model architecture searched by our DARTS-SWF on the

	Task	Dataset	Metric	Backbone		DARTS		β-DARTS		DARTS-ES		GM-DARTS		CNAS		Proposed	
				mean	param	mean	param	mean	param	mean	param	mean	param	mean	param	mean	param
TS2Vec	cls.	UCR	Acc.	0.7752	0.6372	0.7746	0.6703	0.7708	0.7774	0.7639	0.6167	0.7587	0.6136	0.7756	0.6074	**0.8124**	0.5928
		UEA	Acc.	0.7045	0.6373	0.6737	0.6215	0.6719	0.7328	0.6584	0.5744	0.6622	0.9976	0.7087	0.7105	**0.7236**	0.5943
	mul. for.	ETTh1	MSE	0.7828	0.6381	0.8284	0.9611	0.8289	0.9530	0.8132	0.6989	0.8286	0.7988	0.8083	0.8038	**0.7365**	0.9811
		ETTh2	MSE	1.6567	0.6381	1.6988	0.9647	1.7082	0.9530	1.7688	0.6989	1.6274	0.7988	1.5446	0.8038	**1.4220**	0.9811
		ETTm1	MSE	0.6442	0.6381	0.5696	0.9574	0.5652	0.6107	0.6343	0.9489	0.7131	0.5783	**0.5537**	0.8256	0.5649	0.5158
	uni. for.	ETTh1	MSE	0.1125	0.6372	0.1196	0.9607	0.1153	0.5859	0.1297	0.7514	0.1164	0.7333	0.1194	0.8034	**0.0980**	0.768
		ETTh2	MSE	0.1701	0.6372	0.1828	0.9727	0.1986	0.9807	0.1876	0.9891	0.1895	0.9231	0.1728	0.8034	**0.1634**	0.7349
		ETTm1	MSE	0.06548	0.6372	0.0668	0.4564	0.0689	0.7349	0.0666	0.9971	0.0825	0.9231	0.0616	0.8256	**0.0524**	0.5859
	ano.	KPI	F1	0.6360		0.6042		0.5905		0.6149		0.5482		0.6500		**0.6903**	
			Prec	0.9188	0.6372	0.9108	0.5242	0.9113	0.7428	0.9210	0.5850	0.9251	0.7872	**0.9280**	0.7962	0.9235	0.5928
			Rec	0.4864		0.4527		0.4371		0.4515		0.3967		0.5007		**0.5513**	
TS-TCC	cls.	sleepEDF	Acc.	0.8524	0.0857	**0.9428**	0.1706	0.8966	0.1698	0.9000	0.1576	0.893	0.1212	0.8492	0.0715	0.9385	0.1461
		HAR	Acc.	0.8965	0.0985	0.8794	0.2236	0.8895	0.213	0.8848	0.1783	0.8574	0.1829	0.8353	0.1753	**0.9010**	0.2028

Table 2: NAS Results in Various Models, Datasets, and Downstream Tasks

Target Task	Metric	NAS Task			Backbone
		Cls.	Forecast.	Anomaly.	
Cls.	Acc.	$\mathbf{0.7529}$	$\mathbf{0.7186}$	$\mathbf{0.7342}$	0.7057
Forecast.	MSE	$\mathbf{0.0952}$	$\mathbf{0.0875}$	0.1162	0.1138
Anomaly.	F_1	0.5953	$\mathbf{0.6579}$	$\mathbf{0.6924}$	0.6272
	Prec.	0.9293	$\mathbf{0.9264}$	$\mathbf{0.9245}$	0.9199
	Recall	0.4388	$\mathbf{0.5101}$	$\mathbf{0.5534}$	0.4759

Table 3: Results of Architectural Transferability Analysis

window	weight-freezing threshold				
	2.0	2.5	3.0	3.5	4.0
2	0.6771	0.6785	0.7032	0.6814	0.6750
3	0.7247	0.7247	$\mathbf{0.7529}$	0.3928	0.3542
4	0.7257	0.3542	0.3828	0.3542	0.3542

Table 4: Experimental Results in Different Values of Weight-freezing Threshold and Weight Average Window. The bolded setting corresponds to the setting in our main experiment.

AllGestureWiimoteY dataset with 5 different values of k_w and 3 different sizes of averaging window, which is described in Table 4.

In our search method, the greater the value of k_w and the size of the local averaging window are, the more difficult a criterion of whether model weight training should be stopped becomes, which corresponds to the right-bottom section of Table 4. Since there appears a pattern that tests performance decreases with too hard weight-freezing criterion, it implies that weight freezing at an appropriate moment has positive influence on searching for good generalized architecture.

Derivation of our search space

In this section, we introduce the derivation process of our search space in the form of an ablation study. At first, we attempted the commonly-used DAG-based search space where multiple input connections of nodes and operations of a single cell are both optimized, and the optimized cell structure is stacked as multiple layers. Detailed configuration of DAG-based search space is described in Appendix 3. 'Multiple input connection' refers to the structure where outputs of multiple nodes are entered to a single node. But, since there observed degraded performance with the DAG-based search space, we took several other trials, described in Table 5. Even if the cells get the independence of optimization, the results remain inferior to the original model. Surprisingly, there was observed improvement when the multiple input connection structure was eliminated. To specify this process, we took experiments in the form of an ablation study on search space as time-series classification on the AllGestureWiimoteY dataset. The search method was unified as our DARTS-SWF. The experiment results in Table 5 complement our search space in detail. According to the experiments, the absence of multiple inputs and dilated separable convolution operation are the most essential elements in our search space. Also, there existed a performance drop for every candidate operation when each of them was eliminated from our search space.

Structure Search Space	Accuracy	param. size
Original TS2Vec	0.7057	
stacked DAG cells	0.5614	0.7776 MB
independent DAG cells (w/o stackedness)	0.6671	0.4900MB
independent straight cells(ours) (w/o stackedness, multiple inputs)	0.7529	0.6234MB
Candidate Operations	**Accuracy**	**param. size**
w/o dilated conv.	0.71	0.5759MB
w/o dilated sep. conv.	0.5114	0.0958MB
w/o skip connection	0.7442	0.7625MB
w/o self-attention	0.7185	0.9969MB

Table 5: Ablation Study on Search Space

5 Conclusion

Self-supervised representation learning enables the model to extract informative representations from raw data without labels. It is general that a standard architecture is proposed and utilized across various datasets, but the optimal architecture extracting representations could be different depending on dataset-specific dynamics. Hence, we propose a neural architecture search for self-supervised representation learning on time-series data, and to the best of our knowledge, this is the first attempt of its kind. Our experiments with a variety of existing search methods implied that applying existing NAS methods and search spaces of which effectiveness is well-known in the image domain does not necessarily lead to performance improvement. Among a variety of potential reasons, we focused on the scheme of bi-level optimization and vulnerability to overfitting of this field, and devised a new search method that could help to relieve this vulnerability. Specifically, our search method freezes the model weight just before it overfits to train dataset and helps architecture parameters to be optimized on the basis of generalized model weights. Also, we propose a new search space suitable for NAS in this field. Extensive experiments on a variety of time-series representation learning models, datasets, and downstream tasks validate the effectiveness of our method and its suitability to this field.

References

[Andrzejak *et al.*, 2001] Ralph G Andrzejak, Klaus Lehnertz, Florian Mormann, Christoph Rieke, Peter David, and Christian E Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001.

[Anguita *et al.*, 2013] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.

[Bagnall *et al.*, 2018] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

[Cai *et al.*, 2019] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *International Conference on Learning Representations*, 2019.

[Chen *et al.*, 2020a] 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.

[Chen *et al.*, 2020b] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Drnas: Dirichlet neural architecture search. *arXiv preprint arXiv:2006.10355*, 2020.

[Chen *et al.*, 2021] Donghui Chen, Ling Chen, Zongjiang Shang, Youdong Zhang, Bo Wen, and Chenghu Yang. Scale-aware neural architecture search for multivariate time series forecasting. *arXiv preprint arXiv:2112.07459*, 2021.

[Choi *et al.*, 2016] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1495–1504, 2016.

[Dau *et al.*, 2019] 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.

[Dziugaite and Roy, 2017] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[Eldele *et al.*, 2021] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. *arXiv preprint arXiv:2106.14112*, 2021.

[Foret *et al.*, 2020] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[Goldberger *et al.*, 2000] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[Guo *et al.*, 2020] Yong Guo, Yaofo Chen, Yin Zheng, Peilin Zhao, Jian Chen, Junzhou Huang, and Mingkui Tan. Breaking the curse of space explosion: Towards efficient nas with curriculum search. In *International Conference on Machine Learning*, pages 3822–3831. PMLR, 2020.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[Hu *et al.*, 2022] Shoukang Hu, Ruochen Wang, Lanqing Hong, Zhenguo Li, Cho-Jui Hsieh, and Jiashi Feng. Generalizing few-shot nas with gradient matching. *arXiv preprint arXiv:2203.15207*, 2022.

[Hyvarinen and Morioka, 2016] Aapo Hyvarinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. *Advances in neural information processing systems*, 29, 2016.

[Jastrzebski *et al.*, 2017] Stanisław Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[Keskar *et al.*, 2016] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[Komodakis and Gidaris, 2018] Nikos Komodakis and Spyros Gidaris. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations*, 2018.

[Laptev *et al.*, 2015] Nikolay Laptev, Saeed Amizadeh, and Ian Flint. Generic and scalable framework for automated time-series anomaly detection. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1939–1947, 2015.

[Lee *et al.*, 2020] Heung-Chang Lee, Do-Guk Kim, and Bohyung Han. Efficient decoupled neural architecture search by structure and operation sampling. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4222–4226. IEEE, 2020.

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

[Liu *et al.*, 2018] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

[Luo *et al.*, 2018] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Advances in Neural Information Processing Systems*, pages 7827–7838, 2018.

[Malhotra *et al.*, 2017] Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Timenet: Pre-trained deep recurrent neural network for time series classification. *arXiv preprint arXiv:1706.08838*, 2017.

[Rakhshani *et al.*, 2020] Hojjat Rakhshani, Hassan Ismail Fawaz, Lhassane Idoumghar, Germain Forestier, Julien Lepagnot, Jonathan Weber, Mathieu Brévilliers, and Pierre-Alain Muller. Neural architecture search for time series classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[Real *et al.*, 2017] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017.

[Real *et al.*, 2019] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[Ren *et al.*, 2019] Hansheng Ren, Bixiong Xu, Yujing Wang, Chao Yi, Congrui Huang, Xiaoyu Kou, Tony Xing, Mao Yang, Jie Tong, and Qi Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3009–3017, 2019.

[Risso *et al.*, 2022] Matteo Risso, Alessio Burrello, Francesco Conti, Lorenzo Lamberti, Yukai Chen, Luca Benini, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. Lightweight neural architecture search for temporal convolutional networks at the edge. *IEEE Transactions on Computers*, 2022.

[Sermanet *et al.*, 2018] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[Tonekaboni *et al.*, 2021] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. Unsupervised representation learning for time series with temporal neighborhood coding. *arXiv preprint arXiv:2106.00750*, 2021.

[Wen *et al.*, 2019] Yeming Wen, Kevin Luk, Maxime Gazeau, Guodong Zhang, Harris Chan, and Jimmy Ba. An empirical study of large-batch stochastic gradient descent with structured covariance noise. *arXiv preprint arXiv:1902.08234*, 2019.

[Xie and Yuille, 2017] Lingxi Xie and Alan Yuille. Genetic cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1379–1388, 2017.

[Yang and Hong, 2022] Ling Yang and Shenda Hong. Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion. In *International Conference on Machine Learning*, pages 25038–25054. PMLR, 2022.

[Ye *et al.*, 2022] Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. b-darts: Beta-decay regularization for differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10874–10883, 2022.

[Yue *et al.*, 2022] 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, pages 8980–8987, 2022.

[Zela *et al.*, 2019] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. *arXiv preprint arXiv:1909.09656*, 2019.

[Zhang *et al.*, 2016] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

[Zhang *et al.*, 2020] Yuge Zhang, Zejun Lin, Junyang Jiang, Quanlu Zhang, Yujing Wang, Hui Xue, Chen Zhang, and Yaming Yang. Deeper insights into weight sharing in neural architecture search. *arXiv preprint arXiv:2001.01431*, 2020.

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

[Zhao *et al.*, 2021] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. Few-shot neural architecture search. In *International Conference on Machine Learning*, pages 12707–12718. PMLR, 2021.

[Zhou *et al.*, 2021] 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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.