# Neural Architecture and Hyperparameter Selection Through Meta-Learning on Time Series

**Erfan Moeini[1, 2], Christopher Vox[2], Marie Anastacio[1, 3], Wadie Skaf[1], Mitra Barachi[3], Holger H. Hoos[1, 3, 4]**

[1] Chair for Artificial Intelligence Methodology (AIM), RWTH Aachen University, Aachen, Germany
[2] Volkswagen AG, Wolfsburg, Germany
[3] Leiden Institute of Advanced Computer Science (LIACS), Leiden University, Leiden, The Netherlands
[4] University of British Columbia, Vancouver, Canada

## Abstract

Active research in time series classification and forecasting has led to the development of a wide range of machine learning models. For practitioners, the selection of a suitable model among these, along with their hyperparameters, remains a challenging task. While automated machine learning offers approaches for automatic selection of models for a given task, the practical efficacy of these methods is often limited, due to the computational complexity of searching over a large design space and the high dimensionality of time series datasets that poses additional challenges on generalisation quality. To fill this gap, we propose a meta-learning framework that transfers past knowledge from previous searches to recommend an architecture and its hyperparameters; specifically, this framework utilises a joint representation of deep neural architectures and time series datasets, and predicts the performance of neural architectures along with their hyperparameters on time series datasets. Our computational experiments reveal that the configurations proposed by our meta-learned surrogate achieve a performance gain of up to $34\%$ on 4 out of the 8 forecasting datasets we considered and up to $60\%$ on 36 out of 73 of our classification datasets, whilst reducing the computational cost to $10\%$ of that required by the hyperparameter optimisation method HEBO to tune the architectures, showcasing the effectiveness of meta-learning in the time series domain.

**Extended Version** — ada.liacs.nl/papers/MoeEtAl26.pdf

## 1 Introduction

Time series analysis has been an active research area, with applications spanning a wide range of domains, such as weather forecasting (Mung and Phyu 2023) and healthcare (Skaf, Tosayeva, and Várkonyi 2022). To capture the temporal dependencies inherent to time series data, a variety of models have been proposed, with deep learning models using different architectural backbones (*e.g.*, transformers, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or Multi-Layer Perceptrons (MLPs)) showing superior results (Ismail Fawaz et al. 2019; Middlehurst, Schäfer, and Bagnall 2024).

Even though past models adopted different architecture backbones and claimed superiority over one another, it is not realistic to assume that there exists a one-size-fits-all solution, *i.e.*, each architecture might perform best on certain types of datasets and has its own advantages and disadvantages. Manual selection often requires extensive domain-specific and expert knowledge, which renders the usability of these architectures for most users limited within constrained time and computational resources. Consequently, automating or guiding the selection of appropriate architectures and hyperparameter (HP) values based on the characteristics of given time series datasets can be seen as a promising research direction.

Next to recent advances in machine learning, there has also been considerable attention on automating many tasks involved in the selection and configuration of a machine learning model that once required human intervention. This has led to the development of the subfield known as *Automated Machine Learning (AutoML)* (see, *e.g.*, Hutter, Kotthoff, and Vanschoren 2019; Baratchi et al. 2024). In particular, Hyperparameter optimisation (HPO) and Neural Architecture Search (NAS)—two key areas within AutoML—aim at identifying high performing models from a search space of possible design choices (*i.e.*, HPs, architectures), using effective search strategies. However, the practical efficacy of these methods is often restricted due to the computational cost of exploring large, interdependent HP search spaces and the high dimensionality of time series data, which impacts model generalisation. Despite the potential performance improvements that AutoML can achieve, time and computational budgets are restrictive factors to their adoption when tackling large-scale problems. This can mainly be attributed to the fact that these methods are, in essence, negligent of the previous search history, *i.e.*, the search for promising architectures and HP candidates is initiated independently for any given task.

To allow for transferability of knowledge across tasks, we propose a meta-learning-based approach by building a surrogate model for predicting model performance. The main challenge to achieve this goal lies in modelling the performance of architecture and HP pairs by simultaneously capturing the similarities between datasets and architectures. To address this challenge, our contributions are as follows:

1. We extract and compile knowledge regarding the performance of a wide range of deep learning architectures proposed for time series classification and forecasting tasks.

The performance of each architecture is recorded with different HP configurations evaluated during hyperparameter optimisation (HPO) for a fixed number of trials with HEBO (Cowen-Rivers et al. 2022), winner of the 2020 Neurip black-box optimisation challenge.

2. Using meaningful representations for architectures and datasets, our surrogate model preserves their similarity structures in the embedding space, which enables to predict task-dependent performance for new datasets, and eliminates the need to train a new surrogate from scratch for new tasks.
3. We empirically evaluate the prediction of our surrogate on 45 univariate and 29 multivariate time series classification and 8 forecasting datasets, and demonstrate its superiority over our HPO baseline HEBO by using a $k$-shot configuration proposal scheme.

The remainder of this paper is structured as follows: Section 2 introduces the key definitions and notations. Section 3 discusses related work for time series analysis and the application of AutoML in this domain. Section 4 describes our meta-learning framework, the encoding schemes for architectures, HPs and datasets, and the meta-prediction strategy. Section 5 describes our experiments to evaluate and compare our approach to HPO and Section 6 presents the results of the evaluation. Section 7 presents some overall conclusions and potential directions for future work.

## 2 Problem Definition

Time series data is generally defined as a sequence of data points $\mathbf{x_i} \in \mathbb{R}^d$ that capture temporal changes for $d$ variables, where $d \in \mathbb{N}$. Let $\mathbf{X} = [\mathbf{x_1}, \cdots, \mathbf{x_T}] \in \mathbb{R}^{T \times d}$ denote a multi-variate time series of temporal length $T \in \mathbb{N}$ with $d$ variable dimensions. In the following, we present the main problem definitions with regard to the time series $\mathbf{X}$.

**Time Series Classification.** When classifying, the aim is to find a classifier $c : \mathbb{R}^{T \times d} \rightarrow \mathcal{L}_b$ that maps a given time series $\mathbf{X} \in \mathbb{R}^{T \times d}$ to a label $l \in \mathcal{L}_b$, where $\mathcal{L}_b = \{l_1, \cdots, l_k\}$ is the set of $k$ possible labels for the given task: $l = c(\mathbf{X})$

**Time Series Forecasting.** When forecasting, the aim is to learn a function $f : \mathbb{R}^{L \times d} \rightarrow \mathbb{R}^{H \times d}$ that maps past observations within a look-back window of size $L$ to future values within a horizon of length $H$ at time-step $t$, where $L, H$ and $t$ are positive integers: $\hat{\mathbf{Y}}_{t:t+H} = f(\mathbf{X}_{t-L:t})$.

**Meta-Learning for CASH.** We apply meta-learning (Brazdil et al. 2022) to the combined algorithm selection and hyperparameter optimisation (CASH) problem. We aim to find an *arch-hyper* $A_{\boldsymbol{\lambda}}$ – defined by Wu et al. (2023b) as a neural network architecture $A$ from a set of neural network architectures $\mathcal{A}$ with its training and model-specific HPs $\boldsymbol{\lambda}$ from their possible values $\Lambda$ – that minimises a given loss function $\mathcal{L}$ based on the task (time series classification or forecasting):

$$A^*_{\boldsymbol{\lambda}^*} \in \underset{A \in \mathcal{A}, \boldsymbol{\lambda} \in \Lambda}{\arg\min} \mathcal{L}(A_{\boldsymbol{\lambda}}, D_{tr}, D_{val}). \quad (1)$$

We learn a meta-predictor model as surrogate for $\mathcal{L}$ to address the CASH problem for a new time series dataset $D$, split into training $D_{tr}$ and validation $D_{val}$ sets.

## 3 Related Work

Time series analysis and AutoML are key areas in machine learning, yet their intersection remains underexplored. This section reviews key contributions to deep learning for time series in addition to prior efforts to automate neural architecture search and hyperparameter optimisation.

**Deep Learning for Time Series.** Many studies have focused on designing deep learning architectures to extract intricate temporal relations from time series data. For instance, *LightTS* (Zhang et al. 2022) uses an MLP-based structure on top of two down-sampling strategies preserving the majority of information. *ModernTCN* (Donghao and Xue 2024) employs a CNN-based architecture to capture temporal and cross-variable dependencies through convolutional layers at different granularities. To address the limitation of RNNs in handling sequential data with long look-back windows and forecast horizons *SegRNN* model, (Lin et al. 2023) decomposes the time series data into multiple segments, thereby replacing point-wise iterations with segment-wise iterations. More recently, transformers (Zhou et al. 2021; Wu et al. 2021) have gained popularity due to their ability to capture long-term temporal dependencies via self-attention. Despite all the efforts to address the deficiencies of time series models, it still remains unclear what models and HP configurations would be best suited for a given dataset.

**Automatic Architecture and Hyperparameter Selection.** To decide on the HP values to be used, many general-purpose HPO methods rely on some flavour of Bayesian optimisation (BO). *SMAC* (Hutter, Hoos, and Leyton-Brown 2011) uses a random forest (or Gaussian process) surrogate with an acquisition function to efficiently explore configuration spaces. *BOHB* (Falkner, Klein, and Hutter 2018) combines Bayesian optimisation with the early-stopping mechanism and adaptive resource allocation of HyperBand.*HEBO* (Cowen-Rivers et al. 2022) models complex non-uniform noise in the objective function through input warping and output transformations, and uses a multi-objective acquisition function with evolutionary optimisers to overcome potential conflicts between different acquisition functions.

Thornton et al. (2013) defined the CASH problem, unifying algorithm selection and HPO as a single hierarchical optimisation problem, embedding the algorithm choice as an additional HP. Addressing the CASH problem with HPO approaches led to many broad-spectrum AutoML frameworks based on classic machine learning algorithms (*e.g.*, AutoWeka (Thornton et al. 2013), AutoGluon (Erickson et al. 2020)). With the large variety of possible neural network architectures, Neural Architecture Search (NAS) has become a key direction in automated machine learning, aiming to optimise architectural HPs, such as the number of layers and their operations. Consequently, many NAS approaches have been proposed (see *e.g.*, BANANAS (White, Neiswanger, and Savani 2021), DARTS (Liu, Simonyan, and Yang 2019), AutoPyTorch (Zimmer, Lindauer, and Hutter 2021)). Recently, research has been conducted to extend AutoML to be applicable to time series data. *AutoCTS* (Wu et al. 2023b) introduces a joint NAS and HPO framework

using graph isomorphism networks for time series forecasting. The prominent AutoML systems *AutoGluon* (Erickson et al. 2020) and *AutoPyTorch* (Zimmer, Lindauer, and Hutter 2021) introduced time series specific mechanisms in recent years – AutoGluon-TS (Shchur et al. 2023) and AutoPyTorch-TS (Deng et al. 2022), respectively. Despite the proven efficacy of these methods in the time series domain, they all suffer from one important limitation: the selection of appropriate architectures and HPs for a new dataset requires re-evaluation of the entire search space, disregarding the knowledge gained through observations on past datasets. To address this, Mu et al. (2023) proposed a meta-learning approach for time series classification, using decision-trees to recommend algorithms based on dataset meta-features. The method selects from 23 algorithms across 8 categories, but requires retraining to incorporate new algorithms. We address this limitation by proposing a general framework through encoding neural architectures in addition to datasets, thereby eliminating the need to retrain the surrogate for new architectures. We extend their work by also taking into account the forecasting task.

## 4 Proposed Framework

Our proposed framework relies on a meta-predictor that serves as a surrogate model of the performance of arch-hypers. This meta-predictor is later used in a meta-search step to recommend a few neural architectures along with their HP settings for a previously unseen dataset, based on performance predictions. This section explains the key components of this framework: (i) the encoding methods used to learn the joint representation of time series datasets and neural network architecture, (ii) the meta-predictor model and its structure, and (iii) the workflow for CASH using the meta-predictor.

### 4.1 Data and Architecture Embedding

Our approach relies on the ability of our meta-learning model to learn from a database of performance evaluation results of diverse architectures on different time series datasets. For this purpose, neural architectures encodings and their HP values are jointly represented with the encoding of the dataset to ensure that the performance prediction is conditioned on both dataset and architecture representations.

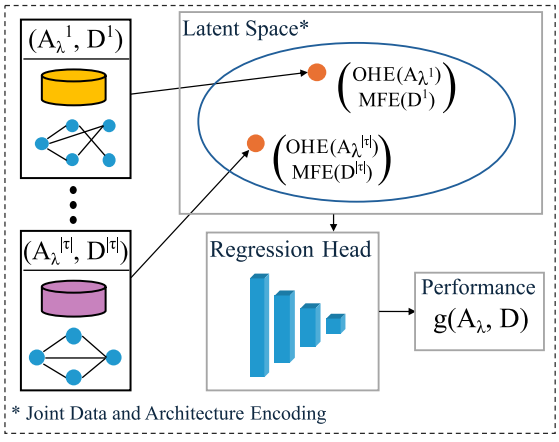


Figure 1: Overview of our meta-predictor. The regression head is a MLP with ReLU activation.

**Time Series Dataset Encoding.** Following similar works in the literature (Mu et al. 2023; Talkhi et al. 2024; Uddin and Lu 2024), we encode time series datasets using a set of meta-features, which describe their statistical and temporal properties. This choice is further motivated in Section 5.

**Architecture Encoding.** During our preliminary analysis (see Section 5), we identified that the one-hot structural encodings of neural architectures and HPs (Akhauri and Abdelfattah 2024; White et al. 2020; Wu et al. 2023b) performs well for our task. To one-hot encode a neural architecture $A$, we extract the associated computational directed acyclic graph (DAG) during a forward pass simulation. In this DAG representation, nodes represent operations and edges information flow between them. The adjacency matrix $M \in \mathbb{R}^{N \times N}$ of the corresponding DAG containing $N \in \mathbb{N}$ nodes is flattened and concatenated with the operations vector $O \in \mathbb{R}^N$ to derive a final representation for the architecture. The operations vector holds for each node the associated numerical value obtained from categorical encoding of the set of all available operations in the candidate architectures. We encode the information about HP values with the min-max normalised HP vector $\lambda \in \mathbb{R}^K$. The categorical HPs are initially encoded using a categorical encoding scheme; these include training HPs, such as learning rate, and model-specific HPs, such as the embedding dimension in transformers. To ensure similar dimensionality for the embedding vectors of different neural architectures, the corresponding adjacency matrix and operation vector are padded with zeros. We fixed the dimension of HP vector and the position of each HP across time series models, assigning reserved values to indicate irrelevance for the current architecture. In summary, using $\|$ to denote concatenation, the one-hot representation of a given architecture is obtained as:

$$\mathrm{OHE}(A_{\boldsymbol{\lambda}}) = \mathrm{FLATTEN}(M) \,\|\, O \,\|\, \lambda. \tag{2}$$

### 4.2 Meta-Predictor Model

The meta-predictor model is the key element in our framework. As illustrated in Figure 1, it is defined as a function $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ that takes the embedding vector representations of a time series dataset $D$ and an arch-hyper $A_\lambda$ as input , and learns to map their joint representation to a final performance metric $v$ that is determined by the downstream time series task. Our approach for arch-hyper representation is inspired by *AutoCTS* (Wu et al. 2023b). We further extend their work by introducing model-specific HPs in addition to training HPs. Inspired by similar works in this domain (Lee, Hyung, and Hwang 2021; Shala et al. 2023), the concatenated data and arch-hyper encoding vectors are fed into a batch of four fully connected linear layers with ReLU activation to obtain a final performance value, as illustrated in Figure 1. HPs of the meta-predictor were tuned with 200 iterations of a simple BO approach. Let $\mathcal{T}$ denote the meta-database containing triplets of the form $\tau = (D, A_\lambda, v)$. Throughout the training phase, the learnable weights of the surrogate of the meta-predictor are updated by minimising a

loss function $\hat{\mathcal{L}}$ over the predicted and true performance values $v$ for each record $\tau$ sampled from the source database:

$$g^* \in \arg\min_g \sum_{(D,A_\lambda,v)\in\mathcal{T}} \hat{\mathcal{L}}(v, g(A_{\boldsymbol{\lambda}}, D)) \tag{3}$$

To decide which loss function to use, we consider that the objective of the meta-predictor is twofold. First, it should accurately estimate the performance of a given arch-hyper on time series datasets. Second, we need to preserve their relative ranking based on their performance. $\hat{\mathcal{L}}$ should penalise both the difference between the estimated and true performance value, and the incorrect ordering of arch-hypers based on those estimated performances. Therefore, we combined the Mean Squared Error (MSE) and Margin Ranking Loss (MRL) as the objective function for training the meta-predictor (Hwang et al. 2024). This objective function, referred to as Combined Ranking Loss (CRL), is defined as:

$$CRL(\mathbf{y}, \hat{\mathbf{y}}) = MSE(\mathbf{y}, \hat{\mathbf{y}}) + MRL(\mathbf{y}, \hat{\mathbf{y}}) \tag{4}$$

$$\text{where } MSE(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{|\mathcal{T}|} \cdot \sum_{i=1}^{|\mathcal{T}|} (y_i - \hat{y}_i)^2,$$

$$\text{and } MRL(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{y_i > y_j} \max(0, \delta - (\hat{y}_i - \hat{y}_j)),$$

where $\mathbf{y} = \left[y_1, \ldots, y_{|\mathcal{T}|}\right]$ and $\hat{\mathbf{y}} = \left[\hat{y}_1, \ldots, \hat{y}_{|\mathcal{T}|}\right]$ with $y_i$ and $\hat{y}_i$ being the ground-truth and predicted performance for the arch-hyper $i$, respectively, and $\delta$ denotes the margin parameter in MRL that indicates how strict the loss function should be when encountering a wrong ordering. Note that each $y_i$ corresponds to an earlier introduced $v$.

Section 5 provides a detailed comparison of the impact of different objective functions on the performance of meta-predictors.

### 4.3 Meta-Search Workflow

The aim of the meta-predictor is to replace the time-consuming steps of selecting an architecture and optimising its HPs. Figure 2 illustrates the proposed framework. The steps of the search strategy are formalised in Algorithm 1. Given a new time series dataset, the meta-search algorithm starts by searching for the most similar dataset according to the $L^1$-norm distance between meta-feature vectors of two datasets (line 1). Then, it searches for the top-$N$ architectures based on past performance evaluations on this dataset (line 2). For each of these $N$ architectures, the meta-predictor predicts the expected performance of a grid of HP values and returns the top-$K$ performing ones (line 6).

## 5 Experimental Setup

The main aim of our experiments is to evaluate the effectiveness of our approach against the widely used approach of tuning the HPs of architectures on the task at hand. To do so, we compare the performance of arch-hypers suggested by our framework to those obtained by independently tuning the HPs of each architecture. In this section, we present the choice of architectures and datasets, the HPO baseline, the data collection approach and the design choices for the meta-predictor.

**Algorithm 1** Meta-Search Algorithm

**Input**: Trained meta-predictor $g$; Set of historical datasets $\mathcal{D}$; Input dataset $D_n$; data similarity function $s$; Historical observations $\mathcal{T}$; HP search space $\Lambda$; Number of top architectures $N$; Number of HP configurations $K$ per architecture; Evaluation function $F$

**Output**: List of top architecture and HP configurations $\mathcal{O}$.

$D_s \leftarrow \arg\min_{D\in\mathcal{D}} s(D_n, D)$; ▷ Find most similar dataset to D

2: $\mathcal{A}^\star \leftarrow \arg\max_{\tau\in\mathcal{T}} F(\tau, D_n, N)$; ▷ Find top-$N$ architectures on the similar dataset

**for all** $A \in \mathcal{A}^\star$ **do**

4: $A_{\boldsymbol{\lambda}} = A \parallel \boldsymbol{\lambda}$; ▷ Create candidate arch-hypers with each architecture and HP configuration in the search space

$\mathbf{v} \leftarrow g(A_{\boldsymbol{\Lambda}}, D_n)$; ▷ Predict performance values

6: $A_{\boldsymbol{\Lambda}^*} \leftarrow \arg\max_K(\mathbf{v})$; ▷ Find top-$K$ HP configurations

$\mathcal{O} \leftarrow \mathcal{O} \cup A_{\boldsymbol{\Lambda}^*}$; ▷ Update the set of output

8: **end for**

10: **return** $\mathcal{O}$

**Architectures.** We considered 20 architectures with different backbones that are frequently used in time series literature. Among CNN-based architectures, we chose ModernTCN (Donghao and Xue 2024), TS2Vec (Yue et al. 2022) and TimesNet (Inception, ResNeXt) (Wu et al. 2023a). Among RNN-based architectures, we use FiLM (Zhou et al. 2022), LMU (Voelker, Kajic, and Eliasmith 2019), Mamba (Gu and Dao 2024), SegRNN (Lin et al. 2023), and S4 (Hasani et al. 2023). Among MLP-based architectures, we included DLinear (Zeng et al. 2023), LightTS (Zhang et al. 2022), and TSMixer (Chen et al. 2023). Finally, among transformer-based architectures, we considered Autoformer (Wu et al. 2021), Crossformer (Zhang and Yan 2023), ETSformer (Woo et al. 2022), Informer (Zhou et al. 2021), PatchTST (Nie et al. 2023), Reformer (Kitaev, Kaiser, and Levskaya 2020), Transformer (Vaswani et al. 2017) and iTransformer (Liu et al. 2024).

Each of these architectures has two types of hyperparameters: (i) training HPs and (ii) model-specific HPs. While the training HPs are similar across all architectures, the model-specific HPs depend on the structure and the underlying functional units. Moreover, some HPs can only be considered depending on the downstream task (*e.g.* decoder component of transformers for regression tasks). A complete list of HPs can be found in the supplementary materials.

**Datasets.** For time series classification, we used the *UEA* multivariate classification datasets (Bagnall et al. 2018), excluding 2 out of 30 datasets: (i) *InsectWingBeats*, due to its extensive training cost, and (ii) *EigenWorms*, due to out-of-memory conditions with both HEBO and the meta-predictor. From the *UCR* univariate classification datasets (Dau et al. 2019), we sampled uniformly at random one third of each subject category specified by the authors (*e.g.*, *Motion*, *Au-*

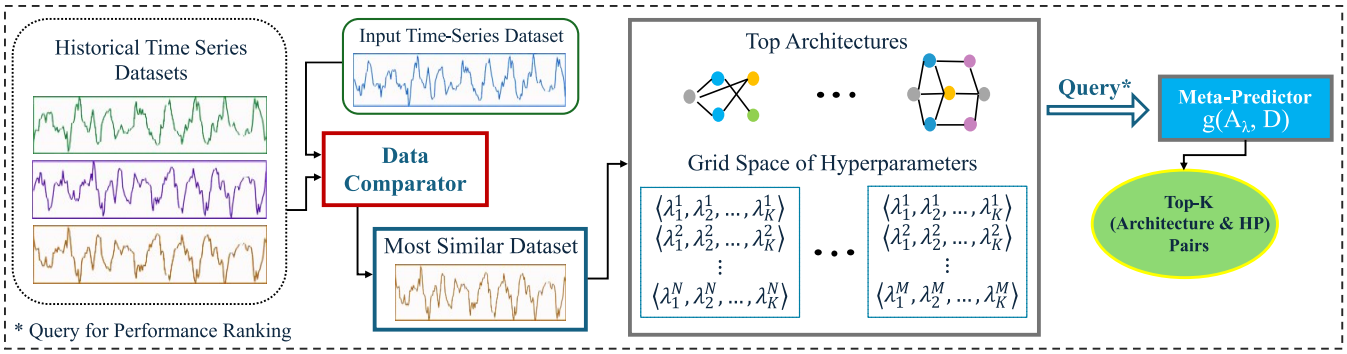


Figure 2: Overview of the meta-search strategy. Once the most similar historical dataset is identified, the meta-predictor predicts the performance values of the top-performing architectures and their grid space of hyperparameters and ranks them accordingly.

*dio*, *Sensor*), resulting in a total of 45 datasets. For time series forecasting, we selected eight datasets frequently used in time series literature. An overview of the selected datasets can be found in the supplementary material.

**Baseline and Data Collection.** For our HPO baseline, we used the *RayTune* implementation of HEBO, winner of the NeurIPS 2020 black-box optimisation competition (Turner et al. 2021), based on preliminary experiments comparing the performance of top HP configurations suggested by several HPO algorithms, including BANANAS (White, Neiswanger, and Savani 2021), HyperOpt (Bergstra et al. 2015) and random search (Bergstra and Bengio 2012), with 128 iterations per algorithm. To train the meta-predictor, we need the performance of a subset of arch-hypers over time series datasets. A typical approach would have been to run a grid search over the HP space of each architecture and observe their performance on each dataset. However, due to the high dimensionality of the search spaces, this approach is inefficient. Instead, we kept track of the performance evaluations carried out during optimisation with HEBO over 128 trials per model-dataset pair. When running a BO algorithm – HEBO in our case – over the HP space of a neural architecture, the HP values are initially sampled uniformly at random, which introduces diversity in the collected samples. As the tuning continues, the Gaussian process surrogates are adapted based on past evaluations and thereby better performing candidates are sampled. This introduces a sampling bias, which is desirable in the context of our application since we are more interested in being accurate in high-performing areas of the HP space. Consequently, we obtained a tabular output indicating the performance of each sampled arch-hyper during the search process. We then replaced each arch-hyper with the corresponding embedding vector and fed it to our meta-predictor.

## Design Choices

In the following, we present the result of a set of experiments conducted to select our data encodings and objective function. Therefore, we split the data such that 70% of the records corresponding to each (model, dataset) pair are used for training, and the remaining 30% are equally distributed between the validation and testing sets. We evaluate the meta-predictor based on MSE and Spearman's rank correlation coefficient (SRC) between the true and predicted values, to account for both of our objectives (have an accurate prediction and keep the ranking intact).

| Encoding | | Spearman's Rank | MSE |
|---|---|---|---|
| Data | Architecture | Correlation | |
| MFE | arch2vec | 0.848 ±0.003 | 0.018 ±0.001 |
| MFE | AdjOp+GCN | 0.841 ±0.005 | 0.020 ±0.001 |
| MFE | AdjOp+GIN | 0.856 ±0.004 | 0.017 ±0.001 |
| MFE | OHE | 0.868 ±0.003 | 0.015 ±0.001 |
| MFE | OHE+LSTM | 0.866 ±0.006 | 0.015 ±0.001 |
| MFE | OHE+Transformer | 0.868 ±0.002 | 0.015 ±0.001 |
| VAE | arch2vec | 0.838 ±0.007 | 0.019 ±0.001 |
| VAE | AdjOp+GCN | 0.837 ±0.001 | 0.020 ±0.000 |
| VAE | AdjOp+GIN | 0.854 ±0.003 | 0.018 ±0.001 |
| VAE | OHE | 0.862 ±0.005 | 0.016 ±0.001 |
| VAE | OHE+LSTM | 0.863 ±0.002 | 0.016 ±0.000 |
| VAE | OHE+Transformer | 0.854 ±0.022 | 0.018 ±0.004 |

Table 1: Performance of meta-predictors trained with different encodings; the best performance is underlined.

**Architecture and Dataset Encoding.** We analysed the impact of different encoding methods on the performance of our meta-predictor. For the dataset encoding, we considered two options: (i) meta-feature encoding (MFE) and (ii) variational autoencoder (VAE). For the architecture encoding, we evaluated 6 methods: (i) one-hot encoding (OHE), (ii) AdjOp + Transformer, (iii) AdjOp + LSTM, (iv) AdjOp + Graph Isomorphism Network (GIN), (v) AdjOp + Graph Convolutional Network (GCN), and (vi) *arch2vec*. This resulted in a total of 12 possible combinations. We trained a meta-predictor for each (data, architecture) encoding pair with 3 different seeds, and compared the performance of the obtained meta-predictors to determine the effectiveness of these encoding schemes. Table 1 summarises the performance results of all encoding combinations. MFE/OHE and MFE/OHE+Transformer resulted in the best relative performance in terms of both regression and ranking criteria. However, due to the computational overhead of transformers, we

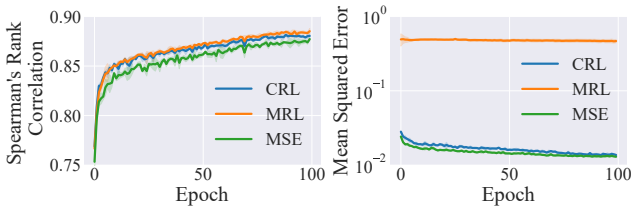


Figure 3: Impact of different loss functions on meta-predictor ranking and regression performance.

| Dataset | HEBO | Meta | Change (%) |
|---|---|---|---|
| ETTh1 | 0.309 ±0.005 | 0.345 ±0.022 | 11.7 ↑ |
| ETTh2 | 0.410 ±0.007 | 0.268 ±0.017 | 34.6 ↓ |
| ETTm1 | 0.210 ±0.005 | 0.293 ±0.003 | 39.1 ↑ |
| ETTm2 | 0.238 ±0.013 | 0.162 ±0.013 | 32.0 ↓ |
| Electricity | 0.134 ±0.001 | 0.140 ±0.004 | 4.6 ↑ |
| Exchange | 0.109 ±0.010 | 0.076 ±0.002 | 30.3 ↓ |
| Traffic | 0.558 ±0.005 | 0.439 ±0.059 | 21.3 ↓ |
| Weather | 0.105 ±0.008 | 0.144 ±0.002 | 37.3 ↑ |

Table 2: HPO and meta-prediction performance on forecasting problems in terms of MSE; the best performance is underlined.

selected MFE/OHE for the encodings used in all other experiments.

**Choice of Objective Function.** We also investigated the impact of the objective used in the meta-predictor on its performance. For this purpose, we trained 3 different meta-predictors, each with one of the objective functions introduced in Section 4. We set the margin parameter $\delta$ to 1.0 based on preliminary experiments with $\delta \in \{0.0, 0.5, 1.0, 2.0\}$. Figure 3 shows the performance of the 3 trained meta-predictors. MRL excelled in terms of the ranking metric, MSE in regression, and CRL found a strong compromise between the two objectives.

## 6 Experiments

We conducted independent experiments for time series classification and forecasting tasks. We run each task with three different seeds, applied to both training the meta predictor and conducting the search strategy.

**Setup.** For each task and each corresponding time series dataset $D$, we trained a meta-predictor over the performance data collected for all other datasets in the same task. During the training phase, we excluded the dataset on which we aimed to evaluate the meta-predictor and split the remaining data such that 70% were used for training and 15% for validation. The remaining 15% were used as a test set in Section 5, and are not used in this set of experiments. The performance data on the excluded dataset is used to evaluate the generalisation of our method. Our trained meta-predictor is integrated into our meta-search strategy to predict high-performing architecture(s) and their HP settings for each dataset $D$. Our method can suggest $N$ architectures and $K$ HP settings for each architecture, with $N$ and $K$ being configurable parameters of our method. To ensure the robustness of our results and reduce sensitivity to outliers, we keep the top-5 HP settings ($K = 5$) for the best found architecture ($N = 1$) (other values were considered in supplementary material). The performance comparison between the suggested architecture(s) and those obtained through HPO is shown in Tables 2-4 separately for each task.

| Dataset | HEBO | Meta | Change (%) |
|---|---|---|---|
| ArticularyWordRecognition | 98.4 ±0.2 | 98.5 ±0.6 | 0.1 ↑ |
| AtrialFibrillation | 73.3 ±0.0 | 63.8 ±10.7 | 12.9 ↓ |
| BasicMotions | 100.0 ±0.0 | 97.5 ±0.0 | 2.5 ↓ |
| CharacterTrajectories | 98.6 ±0.1 | 99.2 ±0.1 | 0.6 ↑ |
| Cricket | 96.9 ±0.6 | 97.4 ±1.2 | 0.5 ↑ |
| DuckDuckGeese | 64.0 ±0.0 | 74.9 ±0.0 | 17.0 ↑ |
| ERing | 95.8 ±0.2 | 93.9 ±0.9 | 2.0 ↓ |
| Epilepsy | 92.2 ±0.4 | 97.1 ±0.8 | 5.4 ↑ |
| EthanolConcentration | 31.1 ±0.2 | 44.1 ±3.8 | 41.8 ↑ |
| FaceDetection | 70.5 ±0.1 | 74.8 ±0.6 | 6.1 ↑ |
| FingerMovements | 62.6 ±1.3 | 66.5 ±5.6 | 6.3 ↑ |
| HandMovementDirection | 68.4 ±0.7 | 66.0 ±4.5 | 3.5 ↓ |
| Handwriting | 37.0 ±0.4 | 30.7 ±18.5 | 17.2 ↓ |
| Heartbeat | 79.8 ±0.6 | 79.9 ±1.5 | 0.1 ↑ |
| JapaneseVowels | 98.0 ±0.2 | 98.5 ±0.4 | 0.5 ↑ |
| LSST | 44.2 ±1.5 | 66.2 ±3.0 | 49.7 ↑ |
| Libras | 86.8 ±0.2 | 87.4 ±5.6 | 0.7 ↑ |
| MotorImagery | 68.4 ±1.5 | 65.4 ±0.8 | 4.4 ↓ |
| NATOPS | 96.1 ±1.3 | 93.6 ±1.0 | 2.6 ↓ |
| PEMS-SF | 88.9 ±0.5 | 85.2 ±4.2 | 4.1 ↓ |
| PenDigits | 98.7 ±0.1 | 98.8 ±0.1 | 0.1 ↑ |
| PhonemeSpectra | 16.0 ±0.0 | 25.6 ±3.1 | 60.2 ↑ |
| RacketSports | 90.1 ±0.5 | 89.4 ±2.7 | 0.8 ↓ |
| SelfRegulationSCP1 | 91.5 ±0.0 | 93.6 ±0.6 | 2.3 ↑ |
| SelfRegulationSCP2 | 59.3 ±1.2 | 62.8 ±5.4 | 5.9 ↑ |
| SpokenArabicDigits | 99.1 ±0.1 | 98.5 ±0.2 | 0.6 ↓ |
| StandWalkJump | 66.7 ±0.0 | 62.8 ±2.6 | 5.8 ↓ |
| UWaveGestureLibrary | 87.8 ±0.2 | 87.3 ±0.8 | 0.6 ↓ |

Table 3: HPO and meta-prediction performance on UEA classification problems in terms of accuracy; the best performance is underlined.

**Results.** We report the performance of the meta-predictor and HEBO. Overall, we would like to verify if the meta-predictor can recommend arch-hypers at least as competitive as those proposed by HEBO without performing the time-consuming HPO procedure. On our forecasting datasets (Table 2), the meta-predictor was able to find a significantly better arch-hyper on half of the datasets, whilst on the other half, the opposite was observed. On UEA (Table 3), the meta-predictor outperformed HEBO on 16 out of 28 datasets, with up to 60% improvement on `PhonemeSpectra` and an average improvement of 12.3%. On the others, HEBO performed on average 4.8% better. On UCR (Table 4), the meta-predictor outperformed HEBO on 20 out of 45 datasets with an average improve-

| Dataset | HEBO | Meta | Change (%) |
|---|---|---|---|
| Adiac | 80.1 ±0.3 | 73.0 ±6.0 | 8.8 ↓ |
| BME | 100.0 ±0.0 | 99.4 ±0.5 | 0.6 ↓ |
| Beef | 83.3 ±0.0 | 85.1 ±2.7 | 2.2 ↑ |
| CBF | 99.9 ±0.1 | 98.6 ±1.0 | 1.4 ↓ |
| Car | 84.7 ±0.8 | 86.1 ±2.2 | 1.7 ↑ |
| Coffee | 100.0 ±0.0 | 100.0 ±0.0 | 0.0 ↕ |
| CricketX | 80.4 ±0.9 | 82.1 ±2.5 | 2.2 ↑ |
| CricketY | 79.0 ±0.6 | 79.7 ±2.4 | 0.8 ↑ |
| Crop | 76.6 ±0.1 | 80.8 ±1.6 | 5.5 ↑ |
| DiatomSizeReduction | 99.7 ±0.0 | 97.1 ±0.7 | 2.6 ↓ |
| DodgerLoopWeekend | 98.6 ±0.0 | 98.9 ±0.2 | 0.3 ↑ |
| ECG200 | 92.4 ±0.5 | 93.2 ±1.6 | 0.9 ↑ |
| ECG5000 | 94.3 ±0.0 | 97.1 ±0.1 | 2.9 ↑ |
| EOGVerticalSignal | 52.0 ±0.9 | 52.9 ±4.3 | 1.7 ↑ |
| ElectricDevices | 75.4 ±0.3 | 78.9 ±1.7 | 4.7 ↑ |
| FaceAll | 91.2 ±0.1 | 85.0 ±5.2 | 6.8 ↓ |
| FaceFour | 96.1 ±1.5 | 90.5 ±1.3 | 5.8 ↓ |
| Fish | 94.9 ±0.4 | 93.8 ±1.8 | 1.1 ↓ |
| FreezerRegularTrain | 99.8 ±0.1 | 99.2 ±0.2 | 0.6 ↓ |
| Fungi | 96.1 ±0.7 | 96.1 ±1.9 | 0.0 ↕ |
| GestureMidAirD1 | 70.8 ±0.5 | 69.9 ±2.5 | 1.3 ↓ |
| GesturePebbleZ2 | 92.3 ±1.4 | 87.8 ±2.4 | 4.9 ↓ |
| GunPointAgeSpan | 100.0 ±0.0 | 99.4 ±0.3 | 0.6 ↓ |
| GunPointMaleVersusFemale | 100.0 ±0.0 | 99.8 ±0.2 | 0.2 ↓ |
| HouseTwenty | 99.2 ±0.0 | 95.6 ±2.2 | 3.6 ↓ |
| InlineSkate | 47.1 ±0.1 | 39.0 ±4.7 | 17.3 ↓ |
| InsectEPGSmallTrain | 91.2 ±0.9 | 89.5 ±2.9 | 1.8 ↓ |
| Lightning2 | 86.6 ±0.6 | 83.8 ±3.1 | 3.2 ↓ |
| Lightning7 | 85.6 ±5.6 | 94.5 ±1.4 | 10.4 ↑ |
| Mallat | 81.5 ±0.2 | 83.2 ±1.2 | 2.2 ↑ |
| MedicalImages | 97.3 ±0.1 | 97.5 ±0.2 | 0.2 ↑ |
| MelbournePedestrian | 84.6 ±0.6 | 85.9 ±1.1 | 1.5 ↑ |
| MiddlePhalanxOutlineCorrect | 94.0 ±0.1 | 94.0 ±0.7 | 0.0 ↕ |
| MixedShapesRegularTrain | 88.0 ±0.0 | 80.2 ±4.0 | 8.8 ↓ |
| PickupGestureWiimoteZ | 100.0 ±0.0 | 100.0 ±0.0 | 0.0 ↕ |
| Plane | 98.3 ±0.0 | 100.0 ±0.0 | 1.7 ↑ |
| PowerCons | 97.0 ±0.3 | 94.1 ±1.4 | 3.0 ↓ |
| SemgHandGenderCh2 | 89.9 ±0.3 | 92.2 ±1.1 | 2.6 ↑ |
| ShapesAll | 78.9 ±0.2 | 79.6 ±1.4 | 0.9 ↑ |
| SmallKitchenAppliances | 97.7 ±0.1 | 97.8 ±0.1 | 0.1 ↑ |
| StarLightCurves | 97.0 ±0.2 | 97.1 ±0.4 | 0.0 ↕ |
| Strawberry | 94.4 ±0.2 | 95.9 ±0.7 | 1.6 ↑ |
| SwedishLeaf | 77.9 ±0.3 | 82.6 ±0.3 | 6.1 ↑ |
| UWaveGestureLibraryZ | 99.9 ±0.0 | 99.7 ±0.1 | 0.2 ↓ |
| Wafer | 99.9 ±0.0 | 99.7 ±0.1 | 0.2 ↓ |

Table 4: HPO and meta-prediction performances on UCR classification problems in terms of accuracy; the best performance is underlined.

ment of 2.5%, on 5 datasets it reached the same performance, and on the remaining 20 it was outperformed by an average of 3.6%. To assess the overall statistical significance of these differences, we conducted a Wilcoxon signed-rank test with a standard significance threshold of 0.05, which detected no significant difference between the performance of the two methods for any of the two tasks. This confirms that our proposed framework is competitive against HEBO.

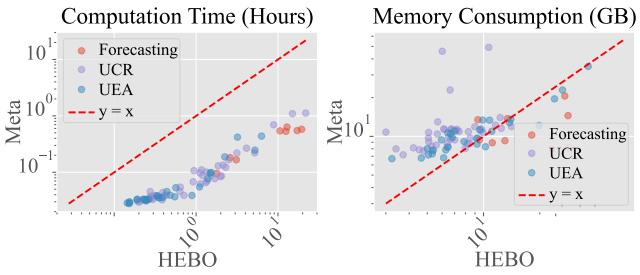


Figure 4: Efficiency comparison between HPO and meta-prediction approach in terms of computation time (hours) and memory consumption (gigabytes).

Figure 4 compares the time and memory consumption of HEBO against our meta-search strategy, demonstrating the clear advantages of our approach. Similar performing arch-hypers, if not better ones, were found using on average 5% for forecasting and 10% for classification of the time required for HEBO. On classification datasets, HEBO is more efficient in terms of memory usage – probably due to the memory required to store the encodings of the whole search space of arch-hypers in our method –, but values are still comparable as seen in the right plot in Figure 4. On forecasting, the meta-predictor uses in average 78% of the amount of memory required by HEBO.

## 7 Conclusion and Future Work

In this work, we examined the ability of a meta-learning approach to automatically determine effective deep learning models along with their hyperparameters for time series classification and forecasting tasks. Our empirical results clearly indicate that incorporating past knowledge from historical observations is viable in finding potentially high-performing architectures and HP configurations for new datasets, using on average 10% of the computational cost. To build our meta-predictor, we evaluated three possible objective functions and designed the CRL representing the best compromise between the ranking and regression metrics used to train the meta-predictor. Similarly, we evaluated the impact of different architectures and dataset encoding methods on the performance of the meta-predictor. Moreover, we analysed the role of different encoding schemes and showed that a simple one-hot encoding for architectures and meta-feature encoding for datasets contains enough information for the meta-predictor to predict performance. Future work should explore alternative encoding schemes that capture information flow during forward and backward passes (Akhauri and Abdelfattah 2024; Hwang et al. 2024), preserving similarities between structurally different but computationally similar architectures. Additionally, since our meta-predictor can, in theory, predict the performance of previously unseen architectures, it might provide value as a surrogate model within a NAS method.

## Acknowledgements

We acknowledge support through an Alexander von Humboldt Professorship in Artificial Intelligence held by Holger Hoos. This work has been conducted during an internship at Volkswagen Aktiengesellschaft.

**Disclaimer** The results, opinions and conclusions expressed in this publication are not necessarily those of Volkswagen Aktiengesellschaft.

## References

Akhauri, Y.; and Abdelfattah, M. S. 2024. Encodings for Prediction-based Neural Architecture Search. In *Proceedings of the International Conference on Machine Learning*.

Bagnall, A.; Dau, H. A.; Lines, J.; Flynn, M.; Large, J.; Bostrom, A. G.; Southam, P.; and Keogh, E. J. 2018. The UEA multivariate time series classification archive, 2018. arXiv:1811.00075v1.

Baratchi, M.; Wang, C.; Limmer, S.; Van Rijn, J. N.; Hoos, H.; Bäck, T.; and Olhofer, M. 2024. Automated machine learning: past, present and future. *Artificial Intelligence Review*.

Bergstra, J.; and Bengio, Y. 2012. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*.

Bergstra, J.; Komer, B.; Eliasmith, C.; Yamins, D.; and Cox, D. D. 2015. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery*.

Brazdil, P.; van Rijn, J.; Soares, C.; and Vanschoren, J. 2022. *Metalearning: Applications to Automated Machine Learning and Data Mining*. Springer.

Chen, S.-A.; Li, C.-L.; Arik, S. O.; Yoder, N. C.; and Pfister, T. 2023. TSMixer: An All-MLP Architecture for Time Series Forecast-ing. *Transactions on Machine Learning Research*.

Cowen-Rivers, A. I.; Lyu, W.; Tutunov, R.; Wang, Z.; Grosnit, A.; Griffiths, R. R.; Maraval, A. M.; Jianye, H.; Wang, J.; Peters, J.; and Bou-Ammar, H. 2022. HEBO: Pushing The Limits of Sample-Efficient Hyper-parameter Optimisation. *Journal of Artificial Intelligence Research*.

Dau, H. A.; Bagnall, A.; Kamgar, K.; Yeh, C.-C. M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C. A.; and Keogh, E. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*.

Deng, D.; Karl, F.; Hutter, F.; Bischl, B.; and Lindauer, M. 2022. Efficient Automated Deep Learning for Time Series Forecasting. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases Conference*.

Donghao, L.; and Xue, W. 2024. ModernTCN: A Modern Pure Convolution Structure for General Time Series Analysis. In *Proceedings of The International Conference on Learning Representations*.

Erickson, N.; Mueller, J. W.; Shirkov, A.; Zhang, H.; Larroy, P.; Li, M.; and Smola, A. 2020. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. arXiv:2003.06505v1.

Falkner, S.; Klein, A.; and Hutter, F. 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the International Conference on Machine Learning*.

Gu, A.; and Dao, T. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *Proceedings of the First Conference on Language Modeling*.

Hasani, R. M.; Lechner, M.; Wang, T.; Chahine, M.; Amini, A.; and Rus, D. 2023. Liquid Structural State-Space Models. In *Proceedings of the International Conference on Learning Representations*.

Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the International Conference on Learning and Intelligent Optimization*.

Hutter, F.; Kotthoff, L.; and Vanschoren, J. 2019. *Automated Machine Learning: Methods, Systems, Challenges*. Springer.

Hwang, D.; Kim, H.; Kim, S.; and Shin, K. 2024. FlowerFormer: Empowering Neural Architecture Encoding Using a Flow-Aware Graph Transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; and Muller, P.-A. 2019. Deep learning for time series classification: a review. *Data Mining Knowledge Discovery*.

Kitaev, N.; Kaiser, L.; and Levskaya, A. 2020. Reformer: The Efficient Transformer. In *Proceedings of the International Conference on Learning Representations*.

Lee, H.; Hyung, E.; and Hwang, S. J. 2021. Rapid Neural Architecture Search by Learning to Generate Graphs from Datasets. In *Proceedings of The International Conference on Learning Representations*.

Lin, S.; Lin, W.; Wu, W.; Zhao, F.; Mo, R.; and Zhang, H. 2023. SegRNN: Segment Recurrent Neural Network for Long-Term Time Series Forecasting. arXiv:2308.11200v1.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *Proceedings of the International Conference on Learning Representations*.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *Proceedings of The International Conference on Learning Representations*.

Middlehurst, M.; Schäfer, P.; and Bagnall, A. 2024. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*.

Mu, T.; Wang, H.; Zheng, S.; Liang, Z.; Wang, C.; Shao, X.; and Liang, Z. 2023. TSC-AutoML: Meta-learning for Automatic Time Series Classification Algorithm Selection. In *Proceedings of the IEEE International Conference on Data Engineering*.

Mung, P. S.; and Phyu, S. 2023. Time Series Weather Data Forecasting Using Deep Learning. In *Proceedings of the IEEE Conference on Computer Applications*.

Nie, Y.; H. Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *Proceedings of the International Conference on Learning Representations*.

Shala, G.; Elsken, T.; Hutter, F.; and Grabocka, J. 2023. Transfer NAS with Meta-learned Bayesian Surrogates. In *Proceedings of the International Conference on Learning Representations*.

Shchur, O.; Turkmen, A. C.; Erickson, N.; Shen, H.; Shirkov, A.; Hu, T.; and Wang, B. 2023. AutoGluon–TimeSeries: AutoML for Probabilistic Time Series Forecasting. In *Proceedings of the International Conference on Automated Machine Learning*.

Skaf, W.; Tosayeva, A.; and Várkonyi, D. T. 2022. Towards Automatic Forecasting: Evaluation of Time-Series Forecasting Models for Chickenpox Cases Estimation in Hungary. In *Proceedings of the International Conference on Intelligent Systems Design and Applications*.

Talkhi, N.; Akhavan Fatemi, N.; Jabbari Nooghabi, M.; Soltani, E.; and Jabbari Nooghabi, A. 2024. Using meta-learning to recommend an appropriate time-series forecasting model. *BMC Public Health*, 24(1): 148.

Thornton, C.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2013. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*.

Turner, R.; Eriksson, D.; McCourt, M.; Kiili, J.; Laaksonen, E.; Xu, Z.; and Guyon, I. 2021. Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*.

Uddin, S.; and Lu, H. 2024. Dataset meta-level and statistical features affect machine learning performance. *Scientific Reports*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In *Proceedings of Advances in Neural Information Processing Systems*.

Voelker, A.; Kajic, I.; and Eliasmith, C. 2019. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In *Proceedings of the International Conference on Neural Information Processing Systems*.

White, C.; Neiswanger, W.; Nolen, S.; and Savani, Y. 2020. A Study on Encodings for Neural Architecture Search. In *Proceedings of Advances in Neural Information Processing Systems*.

White, C.; Neiswanger, W.; and Savani, Y. 2021. BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Woo, G.; Liu, C.; Sahoo, D.; Kumar, A.; and Hoi, S. 2022. ETSformer: Exponential Smoothing Transformers for Time-series Forecasting. arXiv:2202.01381v2.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2023a. TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis. In *Proceedings of the International Conference on Learning Representations*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: decomposition transformers with auto-correlation for long-term series forecasting. In *Proceedings of the International Conference on Neural Information Processing Systems*.

Wu, X.; Zhang, D.; Zhang, M.; Guo, C.; Yang, B.; and Jensen, C. S. 2023b. AutoCTS+: Joint Neural Architecture and Hyperparameter Search for Correlated Time Series Forecasting. In *Proceedings of the ACM on Management of Data*.

Yue, Z.; Wang, Y.; Duan, J.; Yang, T.; Huang, C.; Tong, Y.; and Xu, B. 2022. TS2Vec: Towards Universal Representation of Time Series. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhang, T.; Zhang, Y.; Cao, W.; Bian, J.; Yi, X.; Zheng, S.; and Li, J. 2022. Less Is More: Fast Multivariate Time Series Forecasting with Light Sampling-oriented MLP Structures. arXiv:2207.01186v1.

Zhang, Y.; and Yan, J. 2023. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. In *Proceedings of the International Conference on Learning Representations*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhou, T.; Ma, Z.; Wang, X.; Wen, Q.; Sun, L.; Yao, T.; Yin, W.; and Jin, R. 2022. FiLM: Frequency improved Legendre Memory Model for Long-term Time Series Forecasting. In *Proceedings of the International Conference on Neural Information Processing Systems*.

Zimmer, L.; Lindauer, M.; and Hutter, F. 2021. Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.