Few-Shot Forecasting of Time-Series with Heterogeneous Channels

Lukas Brinkmeyer*✉[0000−0001−5754−1746], Rafael Rego
Drumond*✉[0000−0002−6607−3208], Johannes Burchert, and Lars
Schmidt-Thieme[0000−0001−5729−6023]

University of Hildesheim, Germany
{brinkmeyer,radrumond,burchert,schmidt-thieme}@ismll.uni-hildesheim.de

Abstract. Learning complex time series forecasting models usually requires a large amount of data, as each model is trained from scratch for each task/data set. Leveraging learning experience with similar datasets is a well-established technique for classification problems called few-shot classification. However, existing approaches cannot be applied to time-series forecasting because i) multivariate time-series datasets have different channels, and ii) forecasting is principally different from classification. In this paper, we formalize the problem of few-shot forecasting of time-series with heterogeneous channels for the first time. Extending recent work on heterogeneous attributes in vector data, we develop a model composed of permutation-invariant deep set-blocks which incorporate a temporal embedding. We assemble the first meta-dataset of 40 multivariate time-series datasets and show through experiments that our model provides a good generalization, outperforming baselines carried over from simpler scenarios that either fail to learn across tasks or miss temporal information.

Keywords: Few-shot Learning · Time-series Forecasting · Meta-Learning.

1 Introduction

Time-series research is a central area in the field of machine learning and is widely present in real-life problems and applications ranging from health to the financial sector [24, 23], with time-series data being an essential modality in all of the industry. In particular, time-series forecasting has been in focus of research as it strives to forecast variables over a future time horizon which applies to most data currently being collected. Forecasts can be made for a complete horizon or just a single point in time. Forecasting on univariate time-series, meaning a signal that varies over time and comes without covariates that contain additional information, e.g., the day of the week, is a well-researched area, spanning decades of work with classical approaches being well-studied for all kinds of problem settings [3, 41]. Recently, deep learning approaches are becoming more

---

* Equal contribution

popular in this area, showing to outperform classical approaches when a sufficient amount of training data is available [28]. However, often this is not the case as many time-series datasets are limited in size, giving classical approaches the edge [29]. Specifically, in the case of multivariate time-series data, this is a common problem because datasets have different sets of covariates, making it impossible to learn joint model attributes. Research for time-series shares a lot of commonality with research on image data since both areas are just special cases of structural data. As an example, both benefit from using convolutional networks and transformer-based approaches. However, one main difference in the respective state-of-the-art models is that the best approaches in computer vision rely almost exclusively on a deep feature extractor [13, 47]. These models are pretrained on vast amounts of data pooled from various sources with a trend toward ever-larger collections (e.g. Imagenet [8], JFT-300M [16], JFT-3B [5]) to facilitate the ever-growing models, which nowadays consist of several billion parameters [6]. Meanwhile, this is not easily possible on time-series data due to heterogeneous covariates. Thus, virtually all models are just trained on limited single tasks of time-series data. Looking at the state-of-the-art in the area of computer vision, our aim is to enable the training of a single model across a larger pool of various time-series datasets. The M4 competition was held with that objective [28], by assembling a dataset consisting of 100.000 time-series from various datasets and domains, they analyzed the performance of forecasting approaches when applied to various tasks at once. The clear winner was a hybrid deep learning model by Smyl et al. [43] outperforming any statistical model. However, the competition was limited to univariate time-series data to avoid the problem of dealing with heterogeneous channels and did not introduce unseen datasets in the test evaluation. Most real-world applications involve multivariate time-series data since a set of covariates is almost always given, which can aid forecasting greatly. These covariate channels can be, for example, additional sensors or simple information about the respective day and month.

Learning a single model on a set of different tasks can be achieved through meta-learning. Meta-learning has been hugely successful in various areas of machine learning, with a special focus on computer vision and few-shot image recognition in particular. In contrast to classical machine learning, where a model is typically trained on a single dataset for one specific task, meta-learning aims at learning from a distribution of tasks which can vary in their target [12, 34] or even their predictors [19, 4]. Meta-learning techniques have been successfully applied to various areas of machine learning including few-shot classification [17], hyperparameter optimization [11, 20], reinforcement learning [15] and neural architecture search [25]. In particular, research in few-shot learning has seen an immense rise in popularity, with methods undergoing fundamental changes and benchmarks significantly improving over a very short period of time. Motivated by the fact that humans require only a few examples to correctly classify previously unseen objects based on their past experience, few-shot learning strives to learn models which can generalize to novel tasks based on task-agnostic information extracted from a large set of tasks. Most meta-learning approaches still

require a homogeneous representation across tasks, rendering them not feasible in the application to multivariate time-series tasks with heterogeneous channels. In recent works, various approaches were published to enable machine learning on sets by introducing permutation-invariant and equivariant layers [53, 30]. The work of Iwata et al. [19] incorporated these permutation-invariant layers in a few-shot learning approach to enable learning on vector data with heterogeneous attributes. Encouraged by these findings, we propose the first model for few-shot forecasting on time-series tasks with heterogeneous channels. Our main contributions are as follows:

1. We formalize the problem of few-shot forecasting of time-series with heterogeneous channels for the first time.
2. We develop a model for this new problem that extends prior work on vector data in a principled way.
3. We assemble the first meta-dataset of 40 multivariate time-series datasets and thus provide a public benchmark for future research.
4. We show that our model provides a good generalization, outperforming baselines carried over from simpler scenarios that either fail to learn across tasks or miss temporal information.

2   Related work

There exists a vast amount of research on time-series forecasting and few-shot learning in literature, but the intersection of the two is still very limited, with no existing approaches dealing with few-shot forecasting for time-series tasks with heterogeneous channels. In this section, we will discuss the research of these related areas in a concise way and point out the most important distinctions.

Time-series forecasting focuses on identifying temporal patterns in given data. Historically this was done with methods like ARIMA [3]. In the field of machine learning, CNN and RNN architectures [41, 39] were used to significantly outperform these methods. A further improvement came with the incorporation of attention layers [48] in time-series models [38]. While proving to be very effective, the quadratic complexity of attention comes with a high computational cost. Recent architectures like the Reformer [22], Yformer [27], and Informer [54] focused on reducing this cost by introducing restricted attention layers to effectively approximate the full attention mechanism. Currently, the best performing model architectures are SCINet [26] and N-BEATS [37] on all common datasets and we will compare against them as baselines.

In relation to the problem setting in this work, learning across time-series stemming from different datasets has also been the goal of the popular M4 time-series forecasting competition [28]. However, it was limited to univariate time-series and, more importantly, designed as a classical forecasting problem and not a meta-learning one, meaning that the test set only contained future windows of datasets seen during training. In contrast, our work aims at generalizing to a new time-series dataset during testing, which renders the winning approaches

of the M4 competition not applicable to our problem. **Few-Shot Learning** describes a subarea of meta-learning that deals with evaluating tasks of unseen classes or even datasets with very few labeled samples [51]. By learning from a large collection of related tasks, the model is trained to capture task-agnostic knowledge, which can then be used for a fast adaptation to a novel task that shares this similarity. Different approaches have been proposed with this goal in mind. Gradient-based methods rely on second-order gradient information that is passed across tasks to optimize meta-parameters [12, 33, 40]. Neighbor-based approaches learn a metric embedding space to compare novel tasks [49, 44] while memory-based approaches rely on recurrent components to memorize a representation of the previous tasks [42, 31].

All these methods, however, require a homogeneous predictor and target space in order to learn a joint distribution. One of the first methods to attempt few-shot learning on homogeneous predictors was CHAMELEON [4], which used a convolutional encoder to align tasks from similar domains to a common attribute space before utilizing gradient-based few-shot methods. Similarly, other works tried to learn across tasks with varied label spaces [9, 34]. Finally, Iwata et al. [19] proposed a model that uses deep sets [53] based blocks to compute a task-embedding over predictor and targets of training samples (support data), which then can be combined with new unlabeled samples (query data) to perform classification or regression without the need of retraining or fine-tuning, similar to neighbor-based approaches (we will refer to this method as HETNET throughout the rest of the paper). The main advantage of HETNET is that, since it uses the deep sets formalization, it is invariant to the order of attributes and samples in both query and support set. So far, these approaches are limited to simple vector data and not applicable to structural data.

Few-shot learning for Time-series data. Few works have been published that apply few-shot learning or even meta-learning to time-series data. We argue that this is due to the fact that, in contrast to image data, it is not readily possible to learn a single feature extractor across tasks stemming from different datasets when dealing with multivariate time-series data. Thus, published approaches can be divided into two groups. The first group of approaches utilizes meta-learning techniques to train across slices of the same time-series with homogeneous channels. This includes approaches that combine classical time-series regression with gradient-based meta-learning [1] and approaches that utilize metric-based meta-learning in combination with shapelet learning [46]. These methods are not applicable to our problem setting as they are not equipped to learn across tasks with heterogeneous channels.

Second, there are meta-learning approaches for time-series data that limit their problem setting to univariate time-series tasks in order to learn a single feature extractor without having to deal with heterogeneous channels. Iwata et al. [18] proposed a method to embed tasks through BiLSTM and regular LSTM layers. Narwariya et al. [32] utilized Resnet to embed each time-series to a vector, and then trained across tasks with REPTILE [33] on a meta-dataset of 41

univariate UCR datasets. Lastly, Oreshkin et al. [36] showed how N-BEATS can be used for zero-shot time-series forecasting by rephrasing it in a meta-learning formalization. None of these approaches are capable of dealing with multivariate time-series tasks with heterogeneous channels, which is the focus of this work. Nevertheless, we compare our approach against N-BEATS for zero-shot time-series forecasting by using only the target channel to show that incorporating covariate channels is absolutely necessary for this problem setup. Our method called TimeHetNet serves as the first few-shot time-series forecasting model for multivariate time-series data that can learn across different tasks with heterogeneous channels.

3 Methodology

In this work, we want to propose the first work in the intersection of few-shot learning and multivariate time-series forecasting. We will first formalize the problem of time-series forecasting on a single task before extending it to a few-shot learning setting across a meta-dataset of tasks with heterogeneous channels.

3.1 Problem setting

In the **(vanilla) time-series forecasting problem**, a time-series x with C channels is a finite sequence of vectors in $\mathbb{R}^C$. Their space is denoted by $\mathbb{R}^{*\times C} := \bigcup_{T\in\mathbb{N}} \mathbb{R}^{T\times C}$ with time-series length $|x| := T$. Time-series forecasting data with a single univariate target time-series and C predictor channels is then given by:

$$\mathcal{D} := \{(x_1, y_1, x_1', y_1'), ..., (x_N, y_N, x_N', y_N')\} \in \mathcal{X} \times \mathcal{Y} \tag{1}$$

with $x, x' \in \mathbb{R}^{*\times C}$ and $y, y' \in \mathbb{R}^*$, sampled from an unknown distribution p, where x, y are predictors and targets up to a reference time point $t_0 - 1$ and x', y' denote the corresponding future predictors and targets starting from time point t_0 up to T. The predictors can also be described as future covariate information for the target. Given a loss function $l : \mathbb{R}^* \times \mathbb{R}^* \to \mathbb{R}$, we want to learn a function $\hat{y} : \mathbb{R}^{*\times C} \times \mathbb{R}^* \times \mathbb{R}^{*\times C} \to \mathbb{R}^*$ called model with minimal expected loss over the data:

$$\mathbb{E}_{(x,y,x',y')\sim p} \quad l(y', \hat{y}(x, y, x')) \tag{2}$$

Extending this formalization, the problem of **few-shot time-series forecasting across tasks with heterogeneous channels** is then given by a sample $D := \{(\mathcal{D}_1^s, \mathcal{D}_1^q), ..., (\mathcal{D}_m^s, \mathcal{D}_m^q)\}$ called meta-dataset of pairs $\mathcal{D}^s, \mathcal{D}^q \in (\mathcal{X} \times \mathcal{Y})$ from an unknown distribution p_m of dataset pairs, and a function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Each pair is called a task and consists of support data D^s for which the full instance (x, y, x', y') is known during prediction time and the query data D^q for which the future target y' is not known during prediction time. The number of predictor channels C varies across tasks between $C_{\min}$ and $C_{\max}$. In few-shot learning, the number of samples N^s in the support data D^s is typically low. We

want to find a function $\hat{y} : \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{Y}$ called meta-model with minimal expected loss:

$$\mathbb{E}_{(\mathcal{D}^s, \mathcal{D}^q) \sim p_m} \frac{1}{|\mathcal{D}^q|} \sum_{(x,y,x',y') \in \mathcal{D}^q} \mathcal{L}(y', \hat{y}(x,y,x',\mathcal{D}^s)) \tag{3}$$

In this work, the loss $\mathcal{L}$ is chosen to be the mean squared error (MSE) averaged over the query datasets of our meta-dataset. By weighting only a single time step of the target y', we can learn a point-forecasting model with a specific focus on that one point.

3.2   Model formulation

In this work, we extend HETNET [19] which is a permutation-invariant model for few-shot classification on vector data with heterogeneous attributes. Their model relies on nested deep set-blocks [53] which were first published as a means to enable machine learning models to process sets by introducing a family of permutation-invariant functions. Each deep set-block consists of an inner function f that is applied on each element of the set and an outer function g which is applied after aggregating the output of f such that the block is permutation-invariant to the order of elements. The architecture consists of an inference network that extracts latent task-dependent features of the support data D^s, which are then used by the prediction network in conjunction with the query data $(x, y, x') \in D^q$ to generate the forecast for the future targets y' of a given task.

Inference Network. We adapted the formalization of HETNET [19] to our proposed problem setting by extending it from simple vector data to a forecasting task on structural data. A conceptual depiction of our model is shown in Figure 3.2. First, using the support data D^s we compute the target embeddings $\bar{w}$ and $\bar{c}$ for y and y' respectively, and a predictor embedding $\bar{v}_i$ for each predictor channel $i \in C$ by aggregating across the instances such that the block is permutation-invariant to their order:

$$\bar{v}_i = g_{\bar{v}} \left(\frac{1}{N} \sum_{n=1}^{N} f_{\bar{v}}([x_{ni}, x'_{ni}]) \right) \quad \forall i \in C \tag{4}$$

$$\bar{w} = g_{\bar{w}} \left(\frac{1}{N} \sum_{n=1}^{N} f_{\bar{w}}(y_n) \right), \quad \bar{c} = g_{\bar{c}} \left(\frac{1}{N} \sum_{n=1}^{N} f_{\bar{c}}(y'_n) \right)$$

Here, $\bar{v} \in \mathbb{R}^{C \times T \times K}$ with K being the latent output dimension of $g_{\bar{v}}$, T the length of the time-series task at hand, N the number of samples in the support data D^s and $[\cdot, \cdot]$ a concatenation. Through concatenation of the embeddings with the respective support data followed by another deep-set block, we can generate an
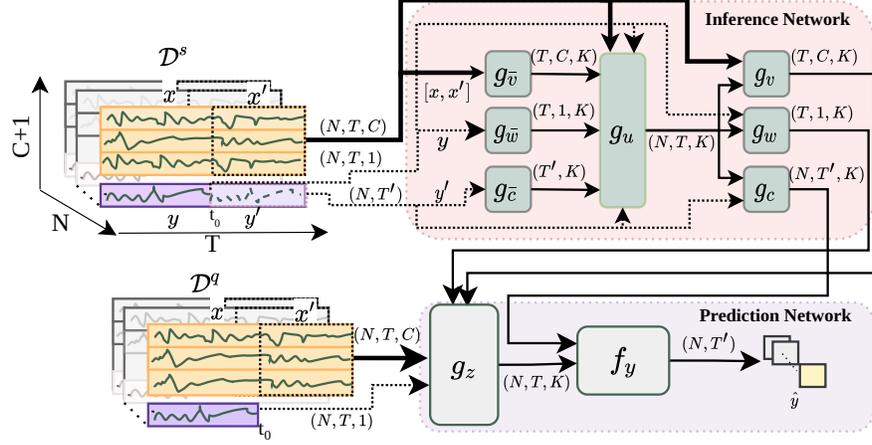
Fig. 1. TIMEHETNET Architecture. Values in parenthesis represent the output shape of each layer. x and y represent the predictor and target channels respectively, while x' and y' represent future predictor and targets. N represents the number of samples in the set, T represents the maximum time length, C describes the number of channels/attributes in each sample. K represents the latent space embedding. T' represents the number of future points of y' we want to predict. For readability: Raw predictors are bold, raw targets are dotted, and latent tensors are regular arrows.

embedding for each instance in the support data D^s through aggregation over the predictor channels:

$$u_n = g_u \left(\frac{1}{C} \sum_{i=1}^{C} \Big(f_u([x_{ni}, x'_{ni}, \bar{v}_i]) \Big) + f_o([y_n, \bar{w}]) + f_p([y'_n, \bar{c}]) \right) \quad \forall n \in N \quad (5)$$

where C is the number of predictor channels of the task and $u \in \mathbb{R}^{N \times T \times K}$. By concatenating these instance-wise embeddings with the respective support data before repeating the block structure of Equation 4, we can again generate the predictor and target embeddings, only this time they are computed with regard to the entire support set. The embeddings for the future target y' are not aggregated over the number of instances, as they are fed directly into the final network f_y generating the forecast on an instance-level:

$$v_i = g_v \left(\frac{1}{N} \sum_{n=1}^{N} f_v([x_{ni}, x'_{ni}, u_n]) \right) \quad \forall i \in C \tag{6}$$

$$w = g_w \left(\frac{1}{N} \sum_{n=1}^{N} f_w([y_n, u_n]) \right), \quad c = f_c([y'_n, u_n])$$

Prediction Network. The embeddings for the predictors v and the past targets w are concatenated with the predictors x, x', and past target y of the
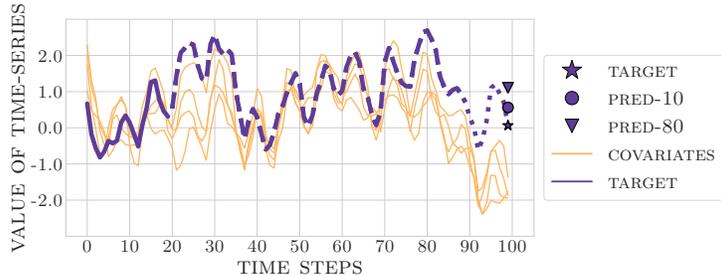
Fig. 2. Task visualization with forecast: The figure shows the four covariate channels (orange) and the target channel (purple) of a query instance for a task sampled from *HandMovementDirection*, and the predictions for the two models trained to predict $t_0 + 10$ (star) and $t_0 + 80$ (triangle) which predict 10 and 80 time steps ahead respectively. The solid target line represents the 20 target time steps y which serve as input to the model trained to predict $t_0 + 80$, while the model trained to predict $t_0 + 10$ receives the target data including the dashed line mark. The dotted line shows the target over the last 10 steps.

query data D^q. This concatenation is passed to a deep set-block consisting of networks g_z and f_z which compute the per-instance features:

$$z_n = g_z \left(\frac{1}{C} \sum_{i=1}^{I} f_z([x_{ni}, y_n, x'_{ni}, v_i, w]) \right) \quad \forall n \in N^q \tag{7}$$

where N^q is the number of samples in D^q. Finally, the prediction for a query sample n is made by passing the embedding z of D^q and the embedding c of the future target of D^s to a final network f_y:

$$\hat{y}_n = f_y([z_n, c]) \tag{8}$$

The full TIMEHETNET then contains the following neural networks $g_{\bar{v}}$, $f_{\bar{v}}$, $g_{\bar{w}}$, $f_{\bar{w}}$, $g_{\bar{c}}$, $f_{\bar{c}}$, g_u, f_u, f_o, f_p, g_v, f_v, g_w, f_w, f_c, g_z, f_z. We share the parameters between the network pairs $(f_{\hat{v}}, f_{\hat{w}}), (g_{\hat{v}}, g_{\hat{w}}), (f_v, f_w), (g_v, g_w)$.

4 Experimental setup

In order to evaluate TIMEHETNET, we have performed an extensive evaluation by creating a meta-dataset to learn few-shot forecasting on multivariate time-series data and comparing the performance of our method to related baseline methods. In this section, we will describe our meta-dataset construction, our baseline methods, the experimental procedures with observed results, and finally, a discussion of our findings.

4.1 Meta-dataset

In order to evaluate our approach, we assembled the first meta-dataset for few-shot forecasting on multivariate time-series data. For that purpose, we collected

40 multivariate time-series datasets consisting of popular forecasting datasets like the *ETT* and *ECL* [54], as well as datasets from the *Monash* time series forecasting archive, [14], 3 *Kaggle* datasets [45, 35, 50], the *UCR* and *UEA* time series archive [2, 7], and also the sparse motion capture dataset *PeekDB* [10]. The details on all datasets are summarized in Table 3 (supplementary material). A single task is sampled from a dataset by randomly selecting between $C_{\min}$ and $C_{\max}$ of the channels, a random slice of the temporal dimension of size T and $N_Q + N_S$ samples for the query and support split of the task. We normalized the channel of each task to mean zero and standard deviation one. In our experiments, we set the number of query and support instances to 20 each, the time length to 100, and varied the number of channels between 5 and 10. In case a dataset has only a single instance, we sample multiple temporal slices for a single task. Furthermore, one channel is selected to be the target channel, with the remaining channels serving as covariate information (during sampling, we make sure that the target channel is not included in the covariates). Finally, the last time step $t = 100$ of the target channel for a given instance is chosen as the future target y', while the last p time steps of the target channel y are removed, thus creating a multivariate time-series forecasting task with covariate information with the aim of forecasting p steps ahead. We evaluated our approach for $p = 1, 10, 80$ and 100, where $p = 1$ corresponds to forecasting the next time step $t + 1$ with the target channel y including the first 99 steps, and $p = 80$ to forecasting the time step $t + 80$ with the target channel including the first 20 time steps. The experiments with $p = 100$ demonstrate an uncontrolled scenario where only covariate channels are given in addition to the future target y' for the support instances. A visualization of a single task is given in Figure 4 including the forecast of our approach.

4.2   Experimental details

We conducted a 5-fold cross-validation with each fold having 8 datasets in meta-test, 8 in meta-validation, and 24 in meta-training. During each epoch of meta-training, we sample 10 meta-batches, where each meta-batch includes one task per dataset: 24 tasks in meta-training and 8 in meta-validation. For the sake of comparability, we generated 11.000 tasks for each dataset in meta-testing beforehand (1.000 tasks per channel size 5 to 10). The final meta-test performance can then be computed by evaluating the model on each of the 440.000 tasks in the fixed test set while guaranteeing comparability between different models. Instead of evaluating our model for the average loss over all time steps, we evaluate the model for individual time steps. By doing this, we want to emphasize the concrete performance differences for close and far events without overlap. For our approach TIMEHETNET, all networks f and g in the deep set-blocks consist of three layers. In all configurations, f_y is a feed-forward network. After optimizing the concrete architecture on the validation tasks, we selected GRU layers for the deep-set blocks formalized in equations 4 and 7 and convolutional layers for 5 and 6. All hyperparameters for all approaches were optimized via grid search. Our model was trained with Adam [21] for a maximum number

Table 1. Experimental Results across all folds. All scores represent Mean Squared Error. Standard deviation is computed over 5 repeated experimental runs. Oracle Channel gives the best possible performance if the best control channel is known. Bold-faced results represent the best scores.

Category	Method	$t_0 + 1$	$t_0 + 10$	$t_0 + 80$	$t_0 + 100$
Proposed	TimeHetNet	**0.148**	**0.389**	**0.509**	**0.579**
	(ours)	**$\pm$ 0.003**	**$\pm$ 0.007**	**$\pm$ 0.006**	**$\pm$ 0.004**
Meta-Learning	HetNet[19]	0.178	0.413	0.524	0.582
		$\pm$ 0.002	$\pm$ 0.003	$\pm$ 0.003	$\pm$ 0.006
	zero prediction	1.006	1.006	1.006	1.006
Heuristic	last time step	0.215	0.899	1.404	$\times$
	avg time step	0.867	0.867	0.867	0.867
	GRU	0.531	0.692	0.699	0.712
Single Task	FCN [52]	0.631	0.791	0.806	0.871
	1D-FF	0.484	0.726	0.845	0.947
No covariates	N-BEATS [36]	0.193	0.677	0.924	$\times$
		$\pm$ 0.002	$\pm$ 0.005	$\pm$ 0.006	$\times$
	SCINet [26]	0.192	0.594	0.718	$\times$
		$\pm$ 0.003	$\pm$ 0.006	$\pm$ 0.006	$\times$
Oracle	Best Channel*	0.353	0.353	0.353	0.353

of 15.000 epochs with early stopping over the validation tasks. More details on our experimental setup and the concrete hyperparameters can be found in our supplementary material.

4.3   Baseline methods

We evaluated our approach against baselines from different related problem settings since there is no approach that can learn across multivariate time-series tasks with heterogeneous channels to the best of our knowledge: a set of heuristics consisting of predicting the constant zero (zero prediction) as our data is normalized to mean zero, predicting the last observed time step of the target channel (last time step), and predicting the average of the last time steps over the covariate channels (avg time step). Moreover, we evaluate a set of models on each individual task in meta-test, namely a stacked GRU network (gru), a fully-convolutional model (fcn) [52] and a feed-forward neural network which relies only on the last observed time step (1d-ff). Note that these models have a lower model complexity than the other approaches, as they are only trained on the support data of a single task without incorporating any other tasks. Additionally, we show the performance of a hypothetical oracle which gives the mean squared error between the target and the last time step of the closest covariate channel (oracle). This model can be used as a point of orientation for the upper limit of the information within the covariate channels and is not a feasible model.

Moreover, we evaluate a set of time-series forecasting models by training them across all datasets without using the covariate channels. Here, we select only the
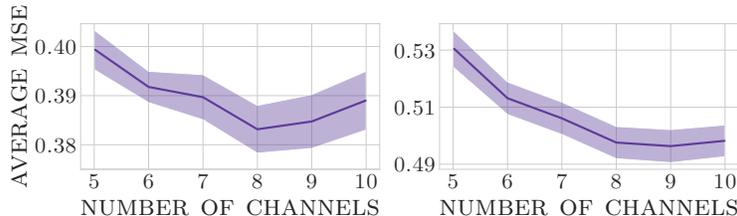
Fig. 3. Varying number of channels: Mean-squared error over the test tasks with fixed number of channels. Results are averaged over all five folds and five repetitions with the corresponding standard deviation (shade) for the experiment $t_0 + 10$ (left) and $t_0 + 80$ (right).

target channel y and the future target y' per task, similar to the setup of Oreshkin et al. [36]. In this setup we evaluate the current best model for time-series forecasting on *PapersWithCode*, SCINET, and the popular time-forecasting model N-BEATS [37] as it is shown as a successful model for for zero-shot time-series forecasting on univariate data [36]. Evaluating the state-of-the-art forecasting models on the single-task level has proven to be infeasible, as a single task is too small for these architectures to train on, as well as too computationally expensive for all tasks in the test set. Lastly, we evaluate HETNET by feeding the last time-steps of x' and y of the time-series as it is the only few-shot learning model which can learn with heterogeneous attributes. The details on all baseline approaches and the training setup can be found in the supplementary material.

4.4   Results

The results for our main experiments can be seen in Table 1 stating the mean-squared error averaged over the test tasks of each fold. It is not useful to compute the standard deviation over cross-validation folds as each fold includes different datasets for which the expected losses naturally vary. Instead, we repeated the full 5-fold cross-validation experiment five times to compute a standard deviation. However, this was too computationally expensive for the models trained on each task from scratch. Thus it is only given for the approaches that train across tasks. Note that heuristical approaches have no standard deviation since they are completely deterministic, and the test tasks are pre-generated. This is the same reason why ZERO PREDICTION, AVG TIME STEP and ORACLE have the same performance for all experiments, as the test tasks only vary in what part of the target channel is given.

   Our approach is shown to outperform all of our baselines in all 4 scenarios. As expected, LAST TIME STEP yields competitive results for the $t_0 + 1$ experiments since oftentimes the next time step does not deviate too much. The closest baseline is HETNET which learns across tasks and utilizes the heterogeneous channels but does not incorporate any past temporal information. The approach is still significantly worse than our proposed model for $t_0 + 1$, $t_0 + 10$ and $t_0 +$
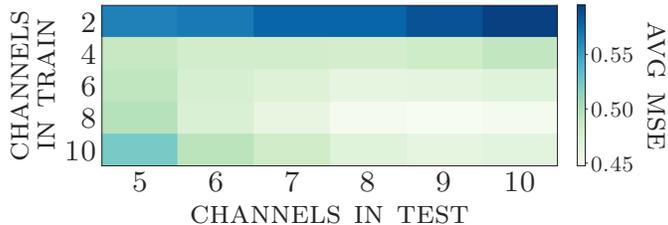
Fig. 4. Heatmap over channels: Performance of our approach for $t_0 + 80$ when trained and evaluated on a fixed task size. Results are averaged over five repetitions of the 5-fold cross validation. Train channels include [2,4,6,8,10].

80 when looking at the standard deviation, suggesting that especially existing temporal information for the target channel aids the prediction. The models trained on a single task from scratch show an expected subpar performance as they can only learn from a single limited few-shot task. Moreover, the state-of-the-art time-series forecasting methods show a good performance on $t_0 + 1$ and $t_0 + 10$, but degrade a lot for $t_0 + 80$, which is the consequence of only relying on the target channel, while especially for $t_0 + 80$ the covariates are shown to be crucial. This fact is emphasized when evaluating the performance of ORACLE as it gives an upper limit on the achievable score when only considering the predictor channels. Note that there are no scores listed for approaches that rely solely on the target channel for $t_0 + 100$ as there is no past target y given in that setting.

4.5   Ablations

We conducted several ablation studies to analyze the robustness of our model with respect to the number of channels across tasks and to show the performance of different architectural choices. First, we analyzed the performance of our approach for $t_0 + 10$ and $t_0 + 80$ with respect to the number of predictor channels for a given novel test task. We show the aggregated results for this experiment in Figure 4.3. We can see that the performance of our approach increases with the number of predictor channels up to 8 channels. Note that c channels here refer to one target channel and $c - 1$ predictor channels for each task. This suggests that our model successfully learned to process tasks independently of their number of channels. It can benefit from the fact that tasks with a higher number of randomly sampled channels are more likely to feature covariates that correlate to the target channel up to a certain point. In the case of predicting 10 steps ahead, the performance slightly degrades for tasks with more than 8 channels.

To investigate the robustness of our model to the number of predictor channels of the tasks in training, we repeated our experiments while limiting the tasks sampled in training to a fixed number of channels. In Figure 4.5 we show the performance of our model for $t_0 + 80$ trained only on tasks with 2, 4, 6, 8 and 10 channels respectively, while evaluating it again for a fixed number of channels

Table 2. Architectures ablation: Comparison of four different architecture setups.

Experiment	GRU-Corner	Conv-Corner	All GRU	All Conv
$t_0 + 80$	**0.509**	0.538	0.512	0.522
$t_0 + 10$	**0.389**	0.397	**0.389**	0.395

in test. One can see that the model is generally robust to the number of channels in training, with the exception of training on tasks with only 2 channels, meaning only one predictor and one target channel. This is most likely due to the number of sampled training tasks that will not have a predictor channel that is sufficiently correlated to the target task. In accord with the previous ablation, there seems to be a slight performance optimum around tasks with 8 channels, while training on tasks with 10 channels and evaluating on tasks with 2 channels also degrades the performance.

Finally, we compare the performance of our model when changing the network design within the deep set blocks. Namely, we evaluate different combinations of convolutional and GRU layers to the one chosen in this work after optimizing the hyperparameters on the validation data. The results are shown in Table 2. The best architecture which is used throughout all our experiments utilizes stacked GRU layers in first the deep-set blocks formalized in Equation 4 which receive the raw data input as well as the one before the final output layer in equation 7 which output the final embedding, while using 1D-convolutional layers in the intermediate blocks (*GRU-Corner*). Using only GRU layers (*All GRU*) degrades the performance by a slight margin while using convolutional layers at the beginning and end of the network is shown to be significantly worse (*Conv-Corner*). This indicates that the GRU blocks are adapting more easily to the very heterogeneous time-series tasks. For comparison reasons, we make our code available at `https://github.com/radrumond/timehetnet`.

5   Conclusion

In this work, we presented the first multivariate time-series forecasting model that works across tasks with heterogeneous channels. Currently, to the best of our knowledge, this is the first work to build a multivariate time-series meta-dataset for this type of meta task. Our model significantly outperforms all related baselines, which either fail to incorporate covariate information or cannot learn across tasks. This approach serves as a benchmark for future research in this area. In future work, we would like to explore the effects of different deep-set blocks and how the model behaves with different types of models and problems.

References

1. Arango, S.P., Heinrich, F., Madhusudhanan, K., Schmidt-Thieme, L.: Multimodal meta-learning for time series regression. In: International Workshop on Advanced Analytics and Learning on Temporal Data. pp. 123–138. Springer (2021)
2. Bagnall, A., Dau, H.A., Lines, J., Flynn, M., Large, J., Bostrom, A., Southam, P., Keogh, E.: The uea multivariate time series classification archive, 2018. arXiv preprint arXiv:1811.00075 (2018)
3. Box, G.E.P., Jenkins, G.M.: Some recent advances in forecasting and control. Journal of the Royal Statistical Society. Series C (Applied Statistics) **17**(2), 91–109 (1968), `http://www.jstor.org/stable/2985674`
4. Brinkmeyer, L., Drumond, R.R., Scholz, R., Grabocka, J., Schmidt-Thieme, L.: Chameleon: learning model initializations across tasks with different schemas. arXiv preprint arXiv:1909.13576 (2019)
5. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1251–1258 (2017)
6. Dai, Z., Liu, H., Le, Q., Tan, M.: Coatnet: Marrying convolution and attention for all data sizes. Advances in Neural IHou, R., Chang, H., Ma, B., Shan, S., and Chen, X. (2019). Cross attention network for few-shot classification. Advances in Neural Information Processing Systems, 32. Information Processing Systems **34** (2021)
7. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The ucr time series archive. IEEE/CAA Journal of Automatica Sinica **6**(6), 1293–1305 (2019)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
9. Drumond, R.R., Brinkmeyer, L., Grabocka, J., Schmidt-Thieme, L.: Hidra: Head initialization across dynamic targets for robust architectures. In: Proceedings of the 2020 SIAM International Conference on Data Mining. pp. 397–405. SIAM (2020), `https://epubs.siam.org/doi/abs/10.1137/1.9781611976236.45`
10. Drumond, R.R., Marques, B.A., Vasconcelos, C.N., Clua, E.: Peek-an lstm recurrent network for motion classification from sparse data. In: VISIGRAPP (1: GRAPP). pp. 215–222 (2018)
11. Feurer, M., Springenberg, J., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 29 (2015)
12. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International conference on machine learning. pp. 1126–1135. PMLR (2017)
13. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. arXiv preprint arXiv:2010.01412 (2020)
14. Godahewa, R., Bergmeir, C., Webb, G.I., Hyndman, R.J., Montero-Manso, P.: Monash time series forecasting archive. arXiv preprint arXiv:2105.06643 (2021)
15. Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., Levine, S.: Meta-reinforcement learning of structured exploration strategies. Advances in neural information processing systems **31** (2018)
16. Hinton, G., Vinyals, O., Dean, J., et al.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 **2**(7) (2015)

17. Hou, R., Chang, H., Ma, B., Shan, S., Chen, X.: Cross attention network for few-shot classification. Advances in Neural Information Processing Systems **32** (2019)
18. Iwata, T., Kumagai, A.: Few-shot learning for time-series forecasting. arXiv preprint arXiv:2009.14379 (2020)
19. Iwata, T., Kumagai, A.: Meta-learning from tasks with heterogeneous attribute spaces. Advances in Neural Information Processing Systems **33**, 6053–6063 (2020)
20. Jawed, S., Jomaa, H., Schmidt-Thieme, L., Grabocka, J.: Multi-task learning curve forecasting across hyperparameter configurations and datasets. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 485–501. Springer (2021)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
22. Kitaev, N., Kaiser, Ł., Levskaya, A.: Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451 (2020)
23. Krollner, B., Vanstone, B.J., Finnie, G.R., et al.: Financial time series forecasting with machine learning techniques: a survey. In: ESANN (2010)
24. Lim, B., Zohren, S.: Time-series forecasting with deep learning: a survey. Philosophical Transactions of the Royal Society A **379**(2194), 20200209 (2021)
25. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
26. Liu, M., Zeng, A., Xu, Z., Lai, Q., Xu, Q.: Time series is a special sequence: Forecasting with sample convolution and interaction. arXiv preprint arXiv:2106.09305 (2021)
27. Madhusudhanan, K., Burchert, J., Duong-Trung, N., Born, S., Schmidt-Thieme, L.: Yformer: U-net inspired transformer architecture for far horizon time series forecasting. arXiv preprint arXiv:2110.08255 (2021)
28. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: The m4 competition: Results, findings, conclusion and way forward. International Journal of Forecasting **34**(4), 802–808 (2018)
29. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: Statistical and machine learning forecasting methods: Concerns and ways forward. PloS one **13**(3), e0194889 (2018)
30. Maron, H., Ben-Hamu, H., Shamir, N., Lipman, Y.: Invariant and equivariant graph networks. arXiv preprint arXiv:1812.09902 (2018)
31. Munkhdalai, T., Yu, H.: Meta networks. In: International Conference on Machine Learning. pp. 2554–2563. PMLR (2017)
32. Narwariya, J., Malhotra, P., Vig, L., Shroff, G., Vishnu, T.: Meta-learning for few-shot time series classification. In: Proceedings of the 7th ACM IKDD CoDS and 25th COMAD. pp. 28–36 (2020)
33. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 (2018)
34. Oh, J., Yoo, H., Kim, C., Yun, S.Y.: {BOIL}: Towards representation change for few-shot learning. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=umIdUL8rMH
35. Oliveira, E.M.: Quality prediction in a mining process, https://www.kaggle.com/datasets/shasun/tool-wear-detection-in-cnc-mill?select=README.txt
36. Oreshkin, B.N., Carpov, D., Chapados, N., Bengio, Y.: Meta-learning framework with applications to zero-shot time-series forecasting. arXiv preprint arXiv:2002.02887 (2020)
37. Oreshkin, B.N., Carpov, D., Chapados, N., Bengio, Y.: N-beats: Neural basis expansion analysis for interpretable time series forecasting. In: International Con-

ference on Learning Representations (2020), `https://openreview.net/forum?id=r1ecqn4YwB`

38. Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., Cottrell, G.: A dual-stage attention-based recurrent neural network for time series prediction. arXiv preprint arXiv:1704.02971 (2017)
39. Rangapuram, S.S., Seeger, M.W., Gasthaus, J., Stella, L., Wang, Y., Januschowski, T.: Deep state space models for time series forecasting. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), `https://proceedings.neurips.cc/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf`
40. Rusu, A.A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., Hadsell, R.: Meta-learning with latent embedding optimization. arXiv preprint arXiv:1807.05960 (2018)
41. Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: Deepar: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting **36**(3), 1181–1191 (2020)
42. Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., Lillicrap, T.: Meta-learning with memory-augmented neural networks. In: International conference on machine learning. pp. 1842–1850. PMLR (2016)
43. Smyl, S.: A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. International Journal of Forecasting **36**(1), 75–85 (2020)
44. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. Advances in neural information processing systems **30** (2017)
45. Sun, S.: Cnc mill tool wear, `https://www.kaggle.com/datasets/shasun/tool-wear-detection-in-cnc-mill?select=README.txt`
46. Tang, W., Liu, L., Long, G.: Interpretable time-series classification on few-shot samples. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)
47. Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al.: Mlp-mixer: An all-mlp architecture for vision. Advances in Neural Information Processing Systems (2021)
48. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
49. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. Advances in neural information processing systems **29** (2016)
50. Von Birgelen, A., Buratti, D., Mager, J., Niggemann, O.: Self-organizing maps for anomaly localization and predictive maintenance in cyber-physical production systems. Procedia cirp **72**, 480–485 (2018)
51. Wang, Y., Yao, Q., Kwok, J.T., Ni, L.M.: Generalizing from a few examples: A survey on few-shot learning. ACM computing surveys (csur) **53**(3), 1–34 (2020)
52. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International joint conference on neural networks (IJCNN). pp. 1578–1585. IEEE (2017)
53. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. Advances in neural information processing systems **30** (2017)
54. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of AAAI (2021)