Forecasting Early with Meta Learning

Shayan Jawed*
Information Science and Machine Learning Lab
University of Hildesheim
Hildesheim, Germany
shayan@ismll.uni-hildesheim.de

Kiran Madhusudhanan*
Information Science and Machine Learning Lab
University of Hildesheim
Hildesheim, Germany
madhusudhanan@ismll.uni-hildesheim.de

Vijaya Krishna Yalavarthi
Information Science and Machine Learning Lab
University of Hildesheim
Hildesheim, Germany
yalavarthi@ismll.uni-hildesheim.de

Lars Schmidt-Thieme
Information Science and Machine Learning Lab
University of Hildesheim
Hildesheim, Germany
schmidt-thieme@ismll.uni-hildesheim.de

***Abstract*—In the early observation period of a time series, there might be only a few historic observations available to learn a model. However, in cases where an existing prior set of datasets is available, Meta learning methods can be applicable. In this paper, we devise a Meta learning method that exploits samples from additional datasets and learns to augment time series through adversarial learning as an auxiliary task for the target dataset. Our model (FEML), is equipped with a shared Convolutional backbone that learns features for varying length inputs from different datasets and has dataset specific heads to forecast for different output lengths. We show that FEML can meta learn across datasets and by additionally learning on adversarial generated samples as auxiliary samples for the target dataset, it can improve the forecasting performance compared to single task learning, and various solutions adapted from Joint learning, Multi-task learning and classic forecasting baselines.**

***Index Terms*—Early Time series forecasting, Meta Learning**

I. Introduction

Time series forecasting is an active area of research with wide applications in multiple domains such as Energy resource management [1], Financial modeling [2], and Environment planning [3] to name a few. For many of these applications to realize, a practical assumption is the availability of Big data [4], [5], given most modern research methods rely on Deep learning from vast amounts of data. This motivates the research question of learning accurate forecasting methods from only limited observations. In practice, limited observations could be a result of not observing the time series for a long duration, for example if the frequency of observation is yearly, or due to limited number of data generating processes, for example, availability of only a few sensors. Arguably one of the most impactful applications recently has been forecasting COVID-19 number of cases and deaths where only limited historic observations were available initially. Herein, Early Time Series Forecasting (eTSF), the task of forecasting the time series given only limited historic observations, can be modeled. We further motivate the problem setting with Fig. 1.

*Equal Contribution

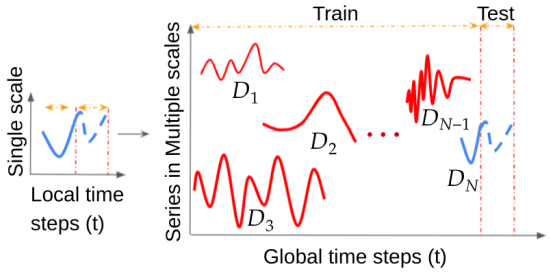


Fig. 1: Motivation for modeling eTSF as a meta-learning problem. The target eTSF dataset D_N (blue) and the auxiliary datasets $\{D_1, \ldots, D_{N-1}\}$ (red). A meta-learning model could utilize the general representation learned across the auxiliary datasets to improve the prediction accuracy on the target dataset where just a few time steps are observed. The dotted line indicates the forecast horizon h_N for target dataset D_N.

However, there is a dearth of research in the direction of general eTSF methods, unlike its classification counterpart of Early time series classification (eTSC) [6] [7]. the early hazard warning forecasting [8], or the early manufacturing data forecasting [9]. Secondly, in comparison to eTSC, eTSF is a more challenging task as it concerns forecasting the full trajectory of the time series from the very limited available data in contrast to solving for in-sample data in classification.

We alleviate the first complication by providing a benchmark for early time series forecasting with 32 datasets collected from the Monash Forecasting Repository [10] covering various domains. The benchmark could motivate the research community for developing domain-independent models for the task of early forecasting. The second complication translates into an interesting research question, and could be solved by considering eTSF as a direct realization of time series compatible methods from the research areas of Meta learning [4], [11] and Transfer learning [12], [13].

To the best of our knowledge, meta-learning for the task of eTSF as shown in Figure 1, is yet to be explored. However, the use of meta-learning methods for general time series forecasting have been explored previously in [4], [12]–[14]. The pioneering works [12], [13] showcased the applicability of well-established forecasting methods to the Zero-shot forecasting problem. The work [4] learned a joint forecasting model across observations from multiple datasets by employing permutation invariant Deep Set blocks [15] that allow learning with heterogeneous multivariate channels. Among the drawbacks concerning the prior approaches are learning from only a single source dataset [12], [13], learning on a fixed sized input and forecast horizon fixed across datasets [4], [12]. Another limiting factor is the lack of gradient based adaptation [4], [12], [13] in the scenario of having a larger number of samples (as related time series), a possibility unconstrained by an early time period.

A significant stream of work is dedicated to data augmentation with the similar goal of improving modeling performance in scarce data scenarios. Traditional approaches that inject Gaussian noise or slice windows can lead to a generation of redundant sampling and require extensive hyperparameter tuning for effective coupling with a downstream learning model [16]. On the other hand, deep generative learning method [17], can augment data in either imbalanced datasets or be applied for anonymization but require substantial data hindering their application for solving the eTSF task.

In order to solve these challenges, we propose a novel model for Forecasting Early with Meta Learning or *FEML* (pronounced "fee·mayl") composed of a convolutional encoder and a stack of linear layer decoders. The convolutional layers learn a joint embedding of input time series across datasets of varying lengths, and the decoding layers enable dataset-specific direct forecasting of the forecast horizons. These Linear layers are further guided by findings in recent studies [18] that credit the linear layers extrapolating for multiple outputs as the main building block of deep forecasting architectures [19]–[23]. We further adapt a well-established meta-learning algorithm *Reptile* [24] to serialized learning across time series datasets with our multi-head forecasting model FEML. Moreover, we design a novel multi-task loss that enables additional learning on adversarially generated samples through the Fast Gradient Sign Method [25] for the target dataset. This further improves the forecasting performance and the applicability of this augmentation scheme is unhindered by lack of samples and generation is guided by tightly coupled learning on the multi-task loss.

We summarize the contributions as follows:

1) To the best of our knowledge, we are the first to address the problem of Early time series forecasting with a principled meta learning solution.
2) We design our method to learn across multiple time series datasets, with varying input and forecast lengths.
3) We provide the first useful benchmark for Early time series forecasting across 32 datasets from the Monash Forecasting Repository [10].
4) We investigate the effect of data augmentation through adversarial learning.
5) We benchmark our method against Statistical baselines, Stand-alone State-of-the-art forecasting models and other meta-learning methods like Joint learning and Multi-task learning.

II. Related Works

Statistical methods for time series forecasting positions themselves as strong baselines while forecasting within the low data regime. Methods like ARIMA [10] and ETS [10], [26] are carefully curated techniques for time series forecasting and have been the default method for explainable forecast from previous decades. ARIMA models are simple regression models where previous lagged observations are used to generate forecast. Simple exponential Smoothing (ETS) computes the forecasts as exponentially decaying weighted averages of past time series observations. However, until the introduction of TBATS [27], none of the statistical baselines were equipped to handle multiple seasonality, high-frequency seasonality, non-integer seasonality and dual calendar effects. TBATS fuses together Box-Cox transformations [28] and Fourier representations along with ARMA error corrections. Since the statistical models leverage only the last few data instances for forecasting the future time steps, they should excel in theory for the task of eTSF. Nevertheless, such a comparison is missing in prior works on meta-learning for forecasting [4], [5], [29].

Modern deep learning architectures based on CNN [30], RNN [31], [32] and Transformer [1], [20] have also been explored for forecasting. Recently, however, the work from [18] has showcased that for most time series forecasting tasks, even linear representations can suffice. For the task of long horizon forecasting, linear methods like DLinear and NLinear [18] were found to outperform counterpart non-linear deep architectures. In essence, the NLinear model is a simple feed-forward network that is meticulously designed to reduce the domain shift while working with long horizon forecasting. Although long horizon forecasting is an interesting topic, it is yet unclear if deep learning methods for time series forecasting would outperform the statistical counterparts for eTSF.

Transferring knowledge from one task to improve performance in a test dataset has been extensively researched within the deep learning community. In Computer vision, models are consistently trained on a large corpus of ImageNet [33] as a pre-training step before fine-tuning the model for the target task. Within the time series forecasting community, transfer learning was used in [34] to improve the electricity demand forecast by transferring knowledge from a source location to that of a target location. The authors achieve the objective by decomposing the time series and transferring knowledge from the decomposed components. In [12], the authors propose a zero-shot training strategy that learns on a source dataset and predicts on a target dataset without learning on the target dataset. The authors interpret the zero-shot results as an evidence supporting the use of meta learning for time series forecasting. Even though, the aforementioned

studies successfully transfer knowledge across datasets, the transfer of knowledge is limited from one dataset. The work from [5] uses a Recurrent neural network equipped with context based Attention for learning meta representations by learning to forecast time series across datasets. In [4] a meta learning model that is able to deal with the problem of having heterogeneous channels was proposed. The method encodes the time series channels with a recurrent network based encoder and then uses Deep Sets [15] for learning invariant representations across the channel space, effectively providing forecasts for multivariate time series with varying number of channels. Another recent approach that aims to transfer representations [29] shares attention representations between a source dataset with many samples and target dataset with fewer samples. Through sharing the attention representations and learning a discriminator that distinguishes samples between the two datasets, the forecasting performance was shown to be improved for the target dataset. In [35] the task of domain adaptation for time series forecasting was explored with augmenting the target dataset with similar samples from various other time series datasets based on computing the DTW based distances [36]. Among the gradient based approaches to meta-learning for time series, [37] focuses on learning multi-modal meta representations. The intuition being that a time series can cover multiple modes and thought of as collection of multiple intra-time related tasks to be modeled. Another recent approach in this direction is also noted in [38].

We delineate from prior works as firstly, we focus on Early time series forecasting where, to the best of our knowledge, there exists no work and secondly, our work eliminates the restriction set by previous meta-learning techniques to have a fixed forecast horizon and input length across datasets [4], [5], [12], [29], [35]. By treating these characteristics as a function of the dataset, we allow for a principled meta-learning solution across datasets. Prior work [12], [29], [35] is also limited to learning only from a single auxiliary dataset.

III. Problem Formulation

A time series dataset D consists of tuple of time series $D := \{(X, Y)|(X, Y) \sim \rho\}$ with $X \in \mathbb{R}^{\delta}$, $Y \in \mathbb{R}^{h}$, are the predictors and forecasts with δ being observation time and h being forecast horizon, drawn from a random distribution ρ. We denote the domain of the dataset as Ω. We consider a meta dataset $\mathcal{D}$ consists of N many datasets with $D_i \in \mathcal{D}$ with domain Ω^i drawn from distribution ρ_i, with δ_i and h_i as the observation time range and forecast horizon. For notational convenience, we denote the number of samples per dataset as M_i. Further, we denote D_N as the support for the target dataset with $\delta_N \ll \delta_i, i < N$. We define the loss function $\ell : \mathbb{R}^{h_N} \times \mathbb{R}^{h_N} \to \mathbb{R}$. Our objective then is to find a meta-model $m : \Omega^1 \times ..., \times \Omega^{N-1} \times \mathbb{R}^{\delta_N} \to \mathbb{R}^{h_N}$, such that the expected loss on $(X, Y) \sim \rho_N$, observed after the occurrence of D_N in chronological order, is minimized:

$$\min \mathop{\mathbb{E}}_{(X,Y)\in\rho_N} \ell(Y, m(X, D_1, ..., D_N))$$

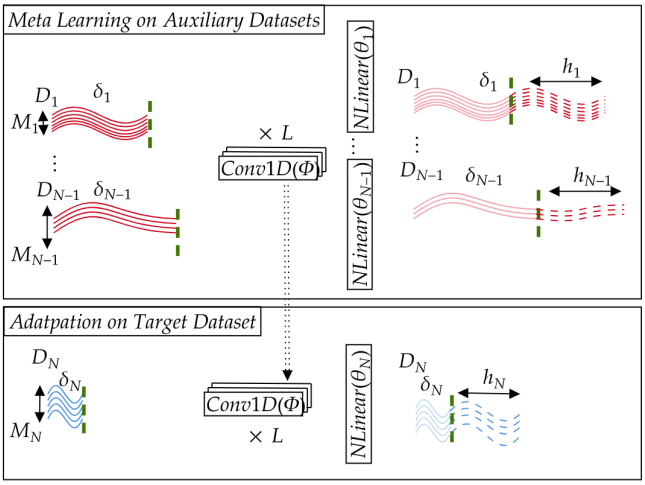


Fig. 2: FEML learns over the samples from the auxiliary datasets $\{D_1, \ldots, D_{N-1}\}$ (represented in red) during the inner iteration and adapts the parameters to the target dataset D_N (represented in blue) in the outer loop. The FEML model consists of shared $\text{Conv1D}(\Phi)$ parameters and dataset specific $\text{NLinear}(\theta_i)$ for $i \in \{1, \ldots, N\}$. Here, δ, h, M indicate the observed range, the forecast horizon and number of tuples in a time series dataset D respectively.

IV. Method

Our forecasting method, FEML learns input length invariant convolutional features for accurate forecasting of multiple datasets. The inputs to FEML are fed through a series of one dimensional convolution layers and non-linear activations. These convolutional features can learn rich locality information. Given that the convolutional features are learned by sliding the 1D kernels over the input window, the convolutional encoding can process input of different lengths from different datasets, expressed as:

$$Z_i = \text{Conv1D}(X_i; \Phi) \tag{1}$$

The Encoder $\text{Conv1D}(.; \Phi) : \mathbb{R}^{1\times\delta_i} \to \mathbb{R}^{d\times\delta_i}$ is composed of L many layers and its learnable parameters are jointly denoted as Φ. Passing the input time series from the i^{th} dataset, we get the embedding: $Z_i \in \mathbb{R}^{d\times\delta_i}$. Where d represents the dimensionality of the convolutional filters.

Importantly, these convolutional features also constitute the shared embedding parameters in FEML ensuring that multiple datasets are embedded in a joint feature space. This distinction is important, as later we will see that we perform meta-learning updates on the shared parameters of the network. Furthermore, the use of shared parameters is also guided by a key-insight in Multi-task learning literature which points to the fact that for optimal Multi-task learning a combination of shared, and task-specific parameters is desired. Therefore, we design multiple output heads composed of task-specific Linear layers that can forecast for each dataset respectively. These layers take the Convolutional encoding Z_i from the Encoder, and then learn

Algorithm 1 Learning for FEML

Require: Time series datasets $\{D_1, ..., D_N\}$, learning rates $\left(\mu^{in}, \mu^{out}, \mu^{ad}, \epsilon, w\right)$, Φ, $\{\theta_i \mid i = 1, ..., N\}$

Initialize Φ
Initialize $\{\theta_i \mid i = 1, ..., N-1\}$
while not converged **do**
  Draw $i \sim \text{Uniform}(\{1, ..., N-1\})$
  $\Phi^{in} \leftarrow \Phi$
  $\theta_i^{in} \leftarrow \theta_i$
  for $(X_i, Y_i) \in D_i$ **do**
    Compute $\hat{Y}_i$ ▷ using Eq. 1,2
    Update Φ^{in}, θ_i^{in} ▷ using Eq. 5
  end for
  Update meta parameters Φ, θ_i ▷ using Eq. 6
  $\Phi^{ad} \leftarrow \Phi$
  Initialize θ_N
  for $(X_N, Y_N) \in D_N$ **do**
    Generate X'_N ▷ using Eq. 3
    Compute $\hat{Y}_N, \hat{Y}'_N$ for X_N, X'_N ▷ using Eq. 1,2
    Compute ℓ_N ▷ using Eq. 4
    Update Φ^{ad}, θ_N ▷ using Eq. 7
  end for
end while
Output Φ^{ad}, θ_N

to extrapolate the forecast directly for all forecast outputs. Our choice for stacking task-specific layers besides having a combination of shared and non-shared parameters among tasks is also based on recent findings from [18] where it was posited that *direct* forecasting with a normalized Linear layer (NLinear), in recent forecasting architectures was key to their success. The output from the i^{th} head can be expressed as:

$$\hat{Y}_i = \text{NLinear}_i(Z_i; \theta_i) \tag{2}$$

The $\text{NLinear}_i(.; \theta_i) : \mathbb{R}^{d \times \delta_i} \rightarrow \mathbb{R}^{h_i}$ where layer parameters θ_i denote the i^{th} task specific parameters for forecasting a fixed horizon per dataset specified apriori. To further elaborate this, although the $\text{Conv1D}(.; \Phi)$ layers can process inputs of various lengths from different datasets and learn shared parameters for those, the $\text{NLinear}_i(.; \theta_i)$ layer takes a flattened fixed length vector and outputs for a fixed sized horizon. Therefore, the $\text{NLinear}_i(.; \theta_i)$ layer is defined for each dataset separately, to forecast for the corresponding forecast horizons. This construction resembles architectures with multiple heads to solve multi-task problems in computer vision tasks [39]–[41]. We can also contrast this construction to a single head decoding model as presented in the ablation.

A. Adversarial learning with FEML

Adversarial samples are samples generated from a similar input data distribution that could confuse the model to output wrong predictions. We follow a well-established approach, Fast Gradient Sign Method (FGSM) [25] to generate adversarial samples. Given input $(X_N, Y_N) \in D_N$ the FGSM method computes an adversarial perturbation by taking a gradient step in the loss maximization direction with respect to the input. This can be expressed simply as:

$$X'_N = X_N + \epsilon \cdot \text{sign}\left(\frac{\partial}{\partial X_N} l\left(Y_N, \hat{Y}_N\right)\right) \tag{3}$$

Where ϵ is the learning rate, or the amount of perturbation based on the sign of the gradient that is added to the input to maximize the loss. Note that we only generate the adversarial samples for the target dataset samples, (X_N, Y_N). Our task then is to incorporate these adversarial samples as additional training samples for the meta-model $m(.)$, effectively doing data augmentation for the target dataset. Consequently, we can generate $\hat{Y}'_N$ as the model forecasts for the adversarial samples X'_N following similar Eq. 1,2. This leads to a multi-task loss formulation combining learning on the augmented and the original samples using a weighting parameter w to increase model robustness:

$$\ell_N = \ell(Y_N, \hat{Y}_N) + w \cdot \ell(Y_N, \hat{Y}'_N) \tag{4}$$

B. Meta learning with FEML

We aim to learn an initialization Φ from a collection of tasks $i \in \{1, ..., N-1\}$ such that, in only a few optimization steps, the parameters Φ can be adapted to solving a new task N. We adapt a serialized version of Reptile [24] to learn across datasets with FEML. We initialize, the two sets of parameters, $\Phi^{in} \leftarrow \Phi$, $\theta_i^{in} \leftarrow \theta_i$ and then for a select number of inner epochs, for the i^{th} sampled task, the shared parameters, Φ and the task specific parameters, θ_i are updated as follows:

$$\begin{aligned} \Phi^{in} &\leftarrow \Phi^{in} - \mu^{in} \frac{\partial}{\partial \Phi^{in}} \ell\left(Y_i, \hat{Y}_i\right) \\ \theta_i^{in} &\leftarrow \theta_i^{in} - \mu^{in} \frac{\partial}{\partial \theta_i^{in}} \ell\left(Y_i, \hat{Y}_i\right) \end{aligned} \tag{5}$$

The meta-update can be expressed with respect to the meta-gradient, being the difference of the previous and the updated parameters with learning on i^{th} task :

$$\begin{aligned} \Phi &\leftarrow \Phi - \mu^{out}(\Phi - \Phi^{in}) \\ \theta_i &\leftarrow \theta_i - \mu^{out}(\theta_i - \theta_i^{in}) \end{aligned} \tag{6}$$

where μ^{in}, μ^{out} denote the inner optimization and outer optimization learning rates respectively. In the serialized optimization, after the meta-gradient updates, the meta-parameters are adapted to the target dataset. This involves learning on few samples from the target dataset, $(X_N, Y_N) \in D_N$.

For the target dataset, a new head is initialized, θ_N. We start the adaptation by copying the shared parameters Φ into another set Φ^{ad}. The parameters updates can be expressed similar to before, however, with the adversarial multi-task loss, ℓ_N:

$$\begin{aligned} \Phi^{ad} &\leftarrow \Phi^{ad} - \mu^{ad} \frac{\partial}{\partial \Phi^{ad}} \ell_N\left(Y_N, \hat{Y}_N, \hat{Y}'_N\right) \\ \theta_N &\leftarrow \theta_N - \mu^{ad} \frac{\partial}{\partial \theta_N} \ell_N\left(Y_N, \hat{Y}_N, \hat{Y}'_N\right) \end{aligned} \tag{7}$$

The optimization routine then involves sampling another task $i \in [1, ..., N-1]$ and updating the parameters by reinitializing $\Phi^{in} \leftarrow \Phi$, $\theta_i^{in} \leftarrow \theta_i$ with the meta-gradient updates before

the adaptation. The serial adaptation thus ensures that the model parameters are not overfit on the target dataset and only adapted through one epoch over the target dataset samples after learning on the i^{th} randomly sampled task.

We can note that the above optimization steps involve updating the shared parameters and only the parameters from the Linear layer corresponding to the task sampled. This is in contrast to the standard update in `Reptile` where no such differentiation is made with regard to task specific and shared parameters, and all parameters are updated jointly. We summarize the method in Algorithm 1.

V. Experiments

A. Datasets

We base our experiments on the first comprehensive time series forecasting repository composed of 25 publicly available time series datasets published by [10]. The datasets can be differentiated by distinctive forecast horizons, series lengths, seasonalities and missing values. Further aggregation across multiple frequencies like monthly, yearly etc., leads to 43 available time series datasets in total.

In our experiments, we limit ourselves to work with 32 datasets from the available 43, by ignoring datasets without date-time information and the datasets with a single univariate series. Additionally, the Monash archive provides the forecast horizon h_i and the lag (observation range) δ_i defined per dataset D_i based on the human expert evaluation for these datasets. As a heuristic, the lag δ_i is computed as the seasonality multiplied with a factor of 1.25. By using this large collection of datasets, we can study the performance of the benchmarked baselines and proposed model comprehensively. We use Intel(R) Xeon(R) CPU E5645 for running all statistical baseline experiments and NVIDIA GeForce GTX 1080 Ti for the deep learning experiments. We mark baselines that required extensive run time (more than 5 days) using "-" symbol within the results table. All reported results are the mean of 3 runs. The code[1] is open sourced for reproducibility.

B. Evaluation Protocol

1) Validation: A trivial choice for validation is to use out-of-sample data [18], [20]. However, within the low data regime, in-sample strategy becomes a more suitable option [32]. For each dataset D_i, we randomly choose 10% percent of the available M_i many univariate time series for in-sample validation and train on the rest 90% of the data. In case of datasets with fewer than 10 time series samples, we randomly assign 1 of these time series as validation, restricting us to work with datasets with more than 1 series.

2) Test: During testing, the model forecasts for the next forecast horizon that was unseen during training.

C. Baselines

Naive Baselines: Our battery of experiments includes benchmarking the following Naive baselines.

[1]https://github.com/super-shayan/FEML

1) *Mean Forecast*: Repeats the mean of the input as the forecast for the entire horizon.
2) *Naive Forecast*: Copies the last season as the forecast.

Statistical Baselines : In order to judge the performance of our method, we compare against well established statistical baselines:

3) *ARIMA* [28]: We fit each univariate time series by varying lagging and differencing hyperparameters.
4) *ETS* [26]: ETS fits the best exponential smoothing parameters automatically for a given time series.
5) *TBATS* [27]: TBATS handles multiple seasonal periods within a time series. TBATS automatically fits the time series by selecting the best seasonality parameters, transformation parameters etc.

Single task Learning Baselines: Another interesting set of baselines are the boosting models and current state-of-the-art time series forecasting models.

6) XGBoost [42]: Gradient boosted methods are ensemble models of decision trees. We tune the number of estimators on a grid of $[100, 200, 400, 800]$ per dataset [43].
7) NLinear [18]: NLinear is a normalized single feed-forward network and requires no hyperparameter tuning.

D. Evaluation

All models are trained using Mean Absolute Error (MAE) [10]. Table I reports results across multiple datasets, using the MASE metric. For M_N samples, the MASE is computed by normalizing the MAE of the evaluated model forecast $\hat{Y}$ with the MAE corresponding to the Naive forecast $\hat{Y}^{\text{naive}}$:

$$\text{MASE} = \frac{1}{M_N} \sum_{j=0}^{M_N} \frac{|Y_j - \hat{Y}_j|}{|Y_j - \hat{Y}_j^{\text{Naive}}|} \tag{8}$$

It avoids symmetry issues, scaling issues and division by 0 issues and is a generally applicable scale free measure to compare across datasets [10]. We use *leave-one-out* strategy during evaluation, by considering one dataset as target dataset D_N for eTSF and assuming that the rest of the datasets $D_i \in \{D_1, \ldots, D_{N-1}\}$ are observed before D_N.

VI. Results

Table I, summarizes the experimental results comparing the proposed `FEML` model with single task baseline on 32 datasets with varying (1) number of time series samples per dataset M_N, (2) forecast horizon h_N and (3) lag δ_N. We sort the table in the increasing order of samples for comparative purposes. Here, we try to answer the following research questions:

RQ1: How does the proposed `FEML` compare against classical and single-task methods?

RQ2: Which statistical baselines are better suited for the task of eTSF?

RQ3: How do the statistical baselines fair against their counterpart deep learning baselines?

Table I: Comparison of the proposed FEML with single task baseline methods based on mean MASE metric. M_N, δ_N, h_N indicate the number of univariate series, input sequence length and the forecast horizon for the target dataset respectively.

Dataset	M_N	δ_N	h_N	Mean	Arima	ETS	TBATS	XGBoost	NLinear	FEML
Aus. Elecdemand	5	420	336	2.009	2.045	2.164	1.195	1.574	1.188	**0.969**
Bitcoin	18	9	30	0.989	**0.684**	0.846	1.026	1.391	0.889	1.025
Pedestrians	66	210	24	1.514	1.501	2.466	0.897	0.493	0.546	**0.475**
FRED-MD	107	15	12	0.574	**0.415**	1.161	2.318	1.021	1.698	0.812
NN5 Weekly	111	65	8	0.852	**0.691**	0.720	0.709	0.699	1.179	0.734
NN5 Daily	111	9	56	0.900	0.874	0.883	0.941	0.645	**0.587**	0.596
Solar Weekly	137	6	5	1.496	**0.554**	1.277	1.189	1.368	1.681	1.366
M1 Yearly	179	2	6	1.020	0.920	1.006	1.253	1.337	0.994	**0.639**
M1 Quarterly	203	5	8	0.933	0.944	0.992	**0.930**	0.940	1.152	1.059
COVID	266	9	30	1.007	**0.948**	0.976	0.952	1.213	0.950	0.974
KDD	270	210	168	**0.672**	0.947	1.037	0.997	0.714	0.863	0.791
Electricity Weekly	321	65	8	1.409	1.361	1.767	**0.884**	2.882	0.902	1.267
Electricity Hourly	321	30	168	2.472	2.360	3.648	**1.037**	2.471	1.325	1.313
Vehicle Trips	329	9	30	0.954	1.020	1.015	1.069	1.133	**0.792**	0.882
M4 Weekly	359	65	13	0.999	0.845	0.824	**0.728**	0.837	0.891	0.776
Tourism Monthly	366	15	24	1.106	1.075	1.122	0.988	0.817	**0.624**	0.625
M4 Hourly	414	210	48	2.224	1.393	2.353	0.805	1.029	0.992	**0.730**
Tourism Quarterly	427	5	8	0.819	0.869	0.754	0.884	0.750	**0.537**	0.621
Tourism Yearly	518	2	4	0.973	0.949	0.928	0.842	1.017	0.951	**0.633**
M1 Monthly	617	15	18	0.917	0.916	0.874	1.003	1.174	**0.843**	0.880
M3 Yearly	645	2	6	1.011	**0.948**	0.998	1.110	1.012	1.001	1.304
M3 Quarterly	756	5	8	0.890	0.882	0.843	0.864	0.795	0.809	**0.712**
Hospital	767	15	12	0.848	0.857	0.910	0.910	1.343	**0.752**	0.841
Traffic Weekly	862	65	8	0.797	0.560	**0.544**	0.655	0.716	0.620	0.664
Traffic Hourly	862	30	168	1.420	1.405	1.575	1.051	1.046	1.248	**1.045**
M3 Monthly	1428	15	18	0.896	0.910	0.873	**0.804**	0.892	0.804	0.896
Carparts	2674	15	12	0.910	0.982	0.948	1.034	0.908	0.899	**0.648**
M4 Daily	4225	9	14	0.948	0.954	0.801	0.892	0.917	0.800	**0.796**
M4 Quarterly	23792	5	8	0.899	0.886	0.821	-	0.784	0.787	**0.767**
Temp. Rain	32072	9	30	**0.861**	0.904	0.945	1.521	1.189	0.904	0.941
M4 Monthly	47776	15	18	0.928	0.783	0.704	-	0.727	0.707	**0.683**
Kaggle Weekly	145063	10	8	0.977	0.963	0.991	-	1.113	0.941	**0.879**
			Wins	2	6	1	5	0	6	12
			% Wins	6.25	18.75	3.125	15.625	0	18.75	37.5

***RQ1**: FEML vs classical and single-task methods*

From Table I, it is clear that the proposed FEML model outperforms the baselines by winning in 12 out of the 32 or 37.5% of all the datasets. The ability of the proposed FEML model to distil useful information across datasets along with adversarial augmentation technique evidently provides the model an advantage for the task of eTSF. For context, the reported results in Monash forecasting archive [10] show that the best performing method is able to win in 19% of datasets for the task of general time series forecasting. Transforming the general time series forecasting dataset to eTSF adds another complexity that only a few observations from target datasets are available. Moreover, the proposed FEML method wins in twice as many datasets versus the second-best baseline.

***RQ2**: Comparison of statistical baselines for eTSF*

As described before, in the eTSF setting, statistical methods should be viewed as strong baselines. Among the statistical baselines, ARIMA model outperforms other statistical baselines, especially for short forecast horizons. In general, TBATS, being a more sophisticated technique and containing ARMA errors, should outperform the simpler ARIMA model, however, tuning the large hyperparameter space in TBATS might restrict the performance of the model in low data regime.

***RQ3**: Statistical baselines vs Learned baselines*

Among the learned baselines, NLinear performs at-par with the ARIMA model by winning in 6 out of the 32 datasets, whereas, the XGBoost model is unable to win in any of 32 datasets, even though the complexity for the XGBoost model was tuned by varying the number of estimators.

VII. Ablation

In this section, we try to analyze each component of the proposed FEML model for eTSF.

RQ4: Does the proposed multi-head architecture perform better than a single shared head architecture?

RQ5: Does the proposed adversarial augmentation strategy improve performance of a base model, for eTSF?

RQ6: How does the proposed FEML compare against other meta-learning methods for eTSF?

The experiments were conducted on the same 32 benchmark datasets and using a similar experimental setup as described in Section V-B. We present the results of the experiments as the number of wins in a head-to-head comparison Fig. 3, where the x-axis shows the datasets arranged in ascending order of the number of samples per dataset (M_N), and the y-axis shows the sum of wins across the number of datasets.

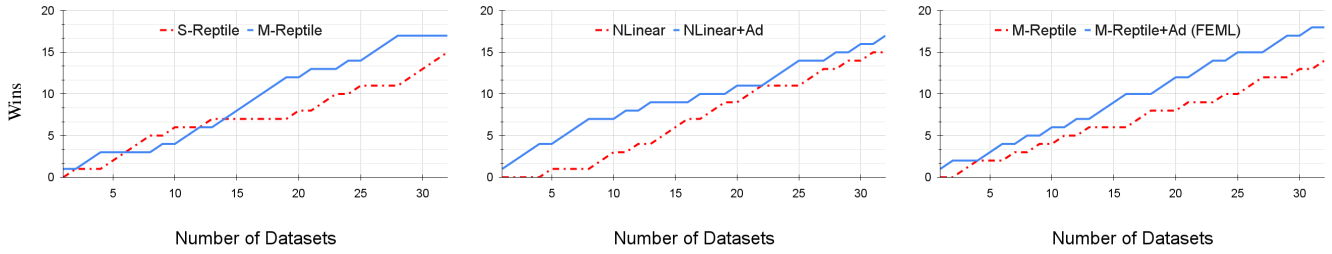
***RQ4** : Multi-head Forecasting Model*

Useful forecast horizon for a particular time series dataset depends on the use-case for that dataset. We build our model on top of this premise and compare FEML without adversarial learning (M-Reptile) with that of a single-head model (S-Reptile) to handle the flexible horizons across datasets. The S-reptile model is a single head model with a shared last linear layer having as many output neurons as the largest forecast horizon. For datasets requiring fewer forecast horizon, we ignore the outputs produced from the extra output nodes.

Figure 3a, compares the number of wins (y-axis) across the number of datasets (x-axis) for the M-Reptile and S-Reptile model. The results show that M-Reptile model wins on more number of datasets in comparison to the S-Reptile model, indicating that a shared output layer across datasets has an adverse effect on the expressiveness of the model. The M-Reptile model with dataset specific multitask heads helps the model to learn dataset specific parameters, which aids the model in outperforming its shared single head counterpart.

***RQ5**: Advantage of Adversarial Augmentation*

In order to understand the advantage offered by the adversarial augmentation, we perform experiments on two baseline models (NLinear and M-Reptile) and show that the adversarial augmentation improves the number of wins in comparison to the base model. Figure 3b highlights the improvement for a single task baseline NLinear with the addition of adversarial augmented examples. For the single task baseline, adversarial examples prove advantageous, especially when the number of samples within the dataset are few. The adversarial samples improve robustness of the baseline models by providing additional samples of the same distribution as the input data, that mitigates overfitting. Furthermore, the improvement is not limited to single task base models. We performed a similar experiment with the M-Reptile model to interpret the improvement brought about by the adversarial samples on top



(a) RQ4: Multi Horizon Comparison (b) RQ5: Adversarial NLinear (c) RQ6: Adversarial M-Reptile

Fig. 3: X-axis represents the number of datasets and the Y-axis represents the wins gathered across the datasets. (a) Comparison of wins across datasets for Multi-Head Reptile model (M-reptile) and shared Single-Head Reptile model (S-Reptile). (b) Plot showing the increase in wins while using adversarial augmentation along with base NLinear base model. (c) Plot showing the increase in wins while using adversarial augmentation along with M-Reptile base model (also proposed as FEML).

of a meta-learning baseline. Interestingly, Figure 3c shows that the number of wins increases even further while comparing the M-Reptile model with and without adversarial samples. The FEML M-Reptile+Ad model has 4 more wins over the 32 datasets (as can be viewed on the right corner of the image) in a head-to-head comparison with the M-Reptile model. The results show that the adversarial augmentation strategy could be beneficial even in a meta-learning setting.

***RQ6**: Comparison to Other Meta-Learning Methods*

We experiment multiple meta-learning methods to determine a useful strategy for eTSF. We introduce three additional baselines for these experiments. All the baselines have similar architecture as the proposed multi-head architecture from FEML and differ only in the learning technique employed.

1) *Joint Learning*: The model is trained on a concatenation of all the available source datasets $\{D_1, ..., D_{N-1}\}$ and the support samples from the target dataset represented as D_N. The model does not learn any task specific embeddings.
2) *Multitask Learning (MTL)*: In MTL the tasks are segregated as target task and auxiliary tasks. For eTSF, the target task is set as learning on the support samples from the target data represented as D_N and the auxiliary targets are the information accumulated from the auxiliary data $\{D_1, ..., D_{N-1}\}$. Mostly, the target task is heavily weighted in comparison to the auxiliary tasks, and we follow the same approach. We set the target task to half of the total weight and share the rest of the weights equally among the auxiliary tasks.
3) *Reptile*: We employ the Reptile training strategy in FEML and is described in Section IV-B.

The results of the experiment are presented in Figure 4. The proposed FEML achieves the highest number of wins when compared to base meta-learning baselines. This could be attributed to the benefit FEML receives from the additional adversarial samples. In comparison to the base MTL and Joint training, Reptile learning algorithm is able to adapt better to

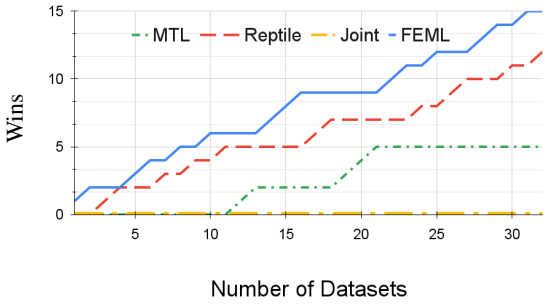


Fig. 4: Comparison of wins across datasets for various meta-learning baselines, namely Joint Learning (Joint), Multi-task Learning (MTL), Reptile, and the proposed FEML.

the fewer samples available for eTSF. Joint training is unable to win in any of the datasets, and this could be explained away as joint training does not prioritize the target dataset D_N any more than the auxiliary datasets, leading to poor performance overall. Multitask Learning (MTL) is able to win in 5 datasets, however is not as effective as the quick adaptation offered by the reptile learning algorithm.

VIII. Conclusion

In this paper, we formally define the task of Early Time Series Forecasting (eTSF), which deals with forecasting when very few observations are available from the target time series dataset. We propose the FEML model, a meta-learning based method to assimilate information from other related or unrelated time series datasets to improve performance on the target dataset. In addition, FEML is equipped with the proposed adversarial augmentation strategy that allows FEML to learn from additional augmented examples of the same distribution as the target dataset. Experimental results on 32 real world time series datasets indicate that the proposed FEML model outperforms statistical, single task and other meta-learning

baselines. We believe this paper provides a foundation for future work in this direction.

In future works, we plan to further explore useful augmentation strategies especially for eTSF as learning useful augmented samples from very few observations could prove beneficial. Research in the direction of a hybrid model that carefully integrates statistical models and deep learning or meta-learning models is also an interesting research direction.

IX. Acknowledgment

This work was supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK), Germany, within the framework of the IIP-Ecosphere project (project number: 01MK20006D) and the industry partner "VWFS DARC: Volkswagen Financial Services Data Analytics Research Center".

References

[1] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *NeurIPS*, vol. 32, 2019.

[2] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *arXiv*, 2017.

[3] S. Miau and W.-H. Hung, "River flooding forecasting and anomaly detection based on deep learning," *IEEE Access*, vol. 8, pp. 198 384–198 402, 2020.

[4] L. Brinkmeyer, R. R. Drumond, J. Burchert, and L. Schmidt-Thieme, "Few-shot forecasting of time-series with heterogeneous channels," in *ECML PKDD*, 2022, pp. 19–23.

[5] T. Iwata and A. Kumagai, "Few-shot learning for time-series forecasting," *arXiv*, 2020.

[6] P. Schäfer and U. Leser, "Teaser: early and accurate time series classification," *Data mining and knowledge discovery*, vol. 34, no. 5, pp. 1336–1362, 2020.

[7] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta, "Approaches and applications of early classification of time series: A review," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 1, pp. 47–61, 2020.

[8] J. Selva, S. Lorito, M. Volpe, F. Romano, R. Tonini, P. Perfetti, F. Bernardi, M. Taroni, A. Scala, A. Babeyko *et al.*, "Probabilistic tsunami forecasting for early warning," *Nature communications*, vol. 12, no. 1, pp. 1–14, 2021.

[9] D.-C. Li, C.-W. Yeh, and C.-J. Chang, "An improved grey-based approach for early manufacturing data forecasting," *Computers & Industrial Engineering*, vol. 57, no. 4, pp. 1161–1167, 2009.

[10] R. Godahewa, C. Bergmeir, G. I. Webb, R. J. Hyndman, and P. Montero-Manso, "Monash time series forecasting archive," *arXiv*, 2021.

[11] E. Vaiciukynas, P. Danenas, V. Kontrimas, and R. Butleris, "Two-step meta-learning for time-series forecasting ensemble," *IEEE Access*, vol. 9, pp. 62 687–62 696, 2021.

[12] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "Meta-learning framework with applications to zero-shot time-series forecasting," in *AAAI*, vol. 35, no. 10, 2021, pp. 9242–9250.

[13] R. Grazzi, V. Flunkert, D. Salinas, T. Januschowski, M. Seeger, and C. Archambeau, "Meta-forecasting by combining global deep representations with local adaptation," *arXiv*, 2021.

[14] S. Jawed, H. Jomaa, L. Schmidt-Thieme, and J. Grabocka, "Multi-task learning curve forecasting across hyperparameter configurations and datasets," in *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21*. Springer, 2021, pp. 485–501.

[15] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *NeurIPS*, vol. 30, 2017.

[16] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," *arXiv*, 2020.

[17] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *NeurIPS*, vol. 32, 2019.

[18] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" 2023.

[19] S. Jawed, A. Rashed, and L. Schmidt-Thieme, "Multi-step forecasting via multi-task learning," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 790–799.

[20] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *AAAI*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[21] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *NeurIPS*, vol. 34, pp. 22 419–22 430, 2021.

[22] S. Jawed and L. Schmidt-Thieme, "Gqformer: A multi-quantile generative transformer for time series forecasting," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 992–1001.

[23] K. Madhusudhanan, J. Burchert, N. Duong-Trung, S. Born, and L. Schmidt-Thieme, "U-net inspired transformer architecture for far horizon time series forecasting," in *ECML PKDD*, 2022, pp. 36–52.

[24] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv*, 2018.

[25] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *ICLR*, 2015.

[26] R. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder, *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[27] A. M. De Livera, R. J. Hyndman, and R. D. Snyder, "Forecasting time series with complex seasonal patterns using exponential smoothing," *Journal of the American statistical association*, vol. 106, no. 496, pp. 1513–1527, 2011.

[28] G. E. Box, G. M. Jenkins, and G. Reinsel, "Time series analysis: forecasting and control holden-day san francisco," *BoxTime Series Analysis: Forecasting and Control Holden Day*, 1970.

[29] X. Jin, Y. Park, D. Maddix, H. Wang, and Y. Wang, "Domain adaptation for time series forecasting via attention sharing," in *ICML*, 2022, pp. 10 280–10 297.

[30] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv*, 2017.

[31] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," *NeurIPS*, vol. 31, 2018.

[32] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. Ieee, 2009, pp. 248–255.

[34] X. Xu and Z. Meng, "A hybrid transfer learning model for short-term electric load forecasting," *Electrical Engineering*, vol. 102, pp. 1371–1381, 2020.

[35] H. Hu, M. Tang, and C. Bai, "Datsing: Data augmented time series forecasting with adversarial domain adaptation," in *CIKM*, 2020, pp. 2061–2064.

[36] C. A. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints," in *ICDM*. SIAM, 2004, pp. 11–22.

[37] S. P. Arango, F. Heinrich, K. Madhusudhanan, and L. Schmidt-Thieme, "Multimodal meta-learning for time series regression," in *ECML PKDD Workshop, AALTD 2021*. Springer, 2021, pp. 123–138.

[38] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, "Deeptime: Deep time-index meta-learning for non-stationary time-series forecasting," *arXiv*, 2022.

[39] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *CVPR*, 2016, pp. 3994–4003.

[40] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018, pp. 7482–7491.

[41] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *CVPR*, 2019, pp. 1871–1880.

[42] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*, 2016, pp. 785–794.

[43] S. Elsayed, D. Thyssens, A. Rashed, H. S. Jomaa, and L. Schmidt-Thieme, "Do we really need deep learning models for time series forecasting?" *arXiv*, 2021.