IN-CONTEXT FINE-TUNING FOR TIME-SERIES FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Motivated by the recent success of time-series foundation models for zero-shot forecasting, we present a methodology for *in-context fine-tuning* of a time-series foundation model. In particular, we design a pretrained foundation model that can be prompted (at inference time) with multiple time-series examples, in order to forecast a target time-series into the future. Our foundation model is specifically trained to utilize examples from multiple related time-series in its context window (in addition to the history of the target time-series) to help it adapt to the specific distribution of the target domain at inference time. We show that such a foundation model that uses in-context examples at inference time can obtain much better performance on popular forecasting benchmarks compared to supervised deep learning methods, statistical models as well as other time-series foundation models. Interestingly, our in-context fine-tuning approach even rivals the performance of a foundation model that is explicitly fine-tuned on the target domain.

1 INTRODUCTION

Time-series data is ubiquitous in several domains such as retail, finance, manufacturing, healthcare, and natural sciences. In many of these domains, time-series forecasting, i.e. predicting time-series into the future, is a critical problem - for example, in applications like retail forecasting, climate and weather predictions, traffic forecasting. In the last decade deep learning approaches (Salinas et al., 2020; Oreshkin et al., 2019; Sen et al., 2019) have become popular in forecasting, often outperforming statistical approaches like ARIMA (Box & Jenkins, 1968). However, until recently, deep learning approaches for forecasting have adhered to the traditional supervised machine learning framework of having to train a forecasting model on task-specific training data, before being able to perform forecasting for that task. On the other hand, in Natural Language Processing (NLP), Large Language Models (LLMs) (Radford et al., 2019; Brown et al., 2020) have shown the promise of foundation models i.e. a single pretrained model can perform well and adapt to tasks like translation, code generation, text summarization during inference time in a zero-shot or few-shot manner.

Motivated by the success in NLP, there has been significant work in recent years on time-series foundation models for forecasting, ranging from re-purposing LLMs directly for forecasting (Gruver et al., 2023) to fine-tuning pretrained LLMs on time-series data (Zhou et al., 2023; Chang et al., 2023) to pretraining time-series foundation models from scratch (Das et al., 2024; Goswami et al., 2024; Woo et al., 2024; Ansari et al., 2024; Garza & Mergenthaler-Canseco, 2023). The last approach in particular has been shown to obtain strong zero-shot accuracy, rivaling the best supervised models trained specifically for the target datasets.

Several of these papers (Zhou et al., 2023; Ansari et al., 2024; Goswami et al., 2024) have shown an opportunity for further accuracy improvement by fine-tuning of their pretrained models on target datasets. However this breaks the zero-shot paradigm that precisely makes these time-series foundation models so appealing to practitioners who do not want to build training pipelines. This raises a natural question: *Can we recover the benefits of fine-tuning a time-series foundation-model, by providing examples from a target dataset at inference time?*

At the same time, the first iterations of these foundation models lack some of the desirable features of LLMs with respect to *in-context learning*: the zero-shot performance of an LLM can be greatly improved *at inference time* by using its context window for prompting techniques such as few-shot (Brown et al., 2020), chain-of-thought (Wei et al., 2022b) or instruction tuning (Wei et al.,

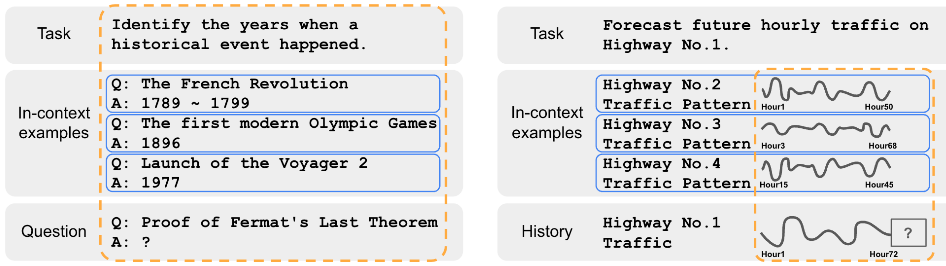


Figure 1: Analogous to few-shot prompting of a foundation LLM (left), we train a time-series foundation model to support few-shot prompting with an arbitrary number of related in-context time-series examples (right). The dashed box encloses the full context window/prompt.

2022a). These papers have shown emergent in-context learning abilities for LLMs i.e. if we prompt them with related examples, demonstrations and instructions, and then ask a specialized question, the model is able to reason similarly for the question at hand.

In this work, we study a methodology to enable similar in-context ability for a time-series foundation model in terms of being able to prompt the model with time-series examples from the target domain, and recover the benefits of domain-specific fine-tuning. We refer to this as *in-context fine-tuning* [1]

In particular, we train a foundation model that lets us forecast a time-series by providing in its context window not just the historical values of the time-series, but also examples from other related time-series that could help the model adapt, *at inference time*, to the distribution of the target time-series. For example, consider a highway traffic prediction system that stores hourly data from the last week, in order to predict the future hourly traffic for a particular highway. Consider a time-series foundation model that has not seen data in pretraining that captures the temporal patterns in this traffic data. Then, simply prompting the model with the previous week's traffic time-series for that highway might not be enough to obtain accurate zero-shot performance. However, adding to the prompt historical traffic data from other highways and weeks, might help the model better adapt to the traffic data distribution, and improve the target accuracy significantly.

To summarize, the main contributions of our paper are as follows:

(i) We introduce the study of in-context fine-tuning for time-series foundation models, and propose the use of prompts that not only include the usual history of the target time-series for forecasting, but also include related time-series examples in-context.

(ii) We pretrain a time-series foundation model to be able to effectively utilize these in-context time-series examples mentioned above. Our training is decoder-only (Liu et al., 2018) and can adapt not only to any context, horizon pair (up to a certain maximum context) but also to any number of supplementary time-series examples (again up to a certain maximum number of examples). Appropriately trained models can then learn to borrow patterns from these related examples to do better on the original forecasting task.

(iii) We empirically evaluate the benefits of in-context fine-tuning using our foundation model. Using evaluations on popular forecasting benchmarks, we show that in-context fine-tuning can lead to better zero-shot performance on popular forecasting benchmarks as compared to supervised deep learning methods, statistical models as well as other foundation models. In particular, it obtains up to 25% better performance than a state-of-the-art time-series foundation model and other supervised deep learning and statistical baselines. Surprisingly, it even slightly improves upon the performance of a time-series foundation model that is specifically fine-tuned to the target datasets.

[1]Terminology: In the LLM domain, this notion is also called "few-shot learning" (Brown et al., 2020), "few-shot prompting" (Ye & Durrett, 2022), or "in-context tuning" (Chen et al., 2022). Also, borrowing from LLM literature, we will refer to the generic ability of pretrained foundation models to learn from information in their context-window at inference time as "in-context learning". Additionally, we will refer to pretrained models that do not need gradient-updates via explicit training or tuning for an unseen target dataset as "zero-shot".

2 RELATED WORK

As mentioned previously, there has been a spurt of recent work on time-series foundation models for forecasting. These approaches can be broadly divided into three categories. (i) Prompting LLMs like GPT-4 to directly predict the future of a numerical series encoded as text. This was investigated in LLMTime (Gruver et al., 2023); despite the initial promise subsequent works have shown that such approaches can be lacking in accuracy (Woo et al., 2024; Das et al., 2024). (ii) fine-tuning pretrained LLMs like GPT2 on time-series data with adapter layers (Zhou et al., 2023; Chang et al., 2023). These approaches have mostly been shown to work well in the dataset-to-dataset transfer learning setting (rather than in the zero-shot setting) and they are also disadvantaged from having to use excessively large models due to their LLM backbones. (iii) Pretraining transformer based models from scratch on huge volumes of time-series data, which seems to be the most promising approach towards times-series foundation models (Das et al., 2024; Garza & Mergenthaler-Canseco, 2023; Ansari et al., 2024; Woo et al., 2024; Goswami et al., 2024). Indeed some of these models have shown superior zero-shot accuracy when compared to supervised deep forecasters and statistical methods even on datasets that are outside of their pretraining set.

Some of the above papers have additionally shown (Ansari et al., 2024; Goswami et al., 2024) that their pretrained models' performance can be further improved by fine-tuning the model on examples from the target dataset. While this supervised fine-tuning results in improved per-task accuracy, this practice breaks the zero-shot paradigm in terms of requiring extra training on the target dataset.

In the NLP domain, a defining property of a foundation LLM is its ability to be further adapted to domain-specific tasks through either fine-tuning or prompting. In particular, LLMs have been shown to perform *in-context learning* on a variety of downstream NLP tasks by prompting them with a natural language instruction (Radford et al., 2019) and a few demonstrations or examples of the task. This phenomenon is also referred to as *few-shot learning* (Brown et al., 2020). Subsequent works (Min et al., 2022a; Chen et al., 2022) have proposed fine-tuning a pretrained LLM to obtain better performance on few-shot learning prompts. Other papers (Min et al., 2022b; Wei et al., 2023) have empirically investigated how few-shot learning works in LLMs. More recently, Shi et al. (2023) explored a similar idea for in-context pretraining, where they pretrain an LLM on sequences of related documents. This in-context learning ability is widely recognized as being associated with the stacked transformers used in the LLMs, and their theoretical properties are studied in a more precise sense (Garg et al., 2022; Von Oswald et al., 2023; Ahn et al., 2024) for simpler function classes such as linear regression.

Despite the commonality between time-series foundation models and LLMs, it is not obvious how (or even if) the phenomenon of few-shot learning for NLP tasks carry over to the time-series setting. There is no clear definition of few-shot learning in terms of a time-series foundation model. In fact prior pretrained time-series foundation models like (Ansari et al., 2024; Das et al., 2024; Garza & Mergenthaler-Canseco, 2023) do not provide a clear opportunity to be prompted with anything apart from the past values of a time-series in the context window.

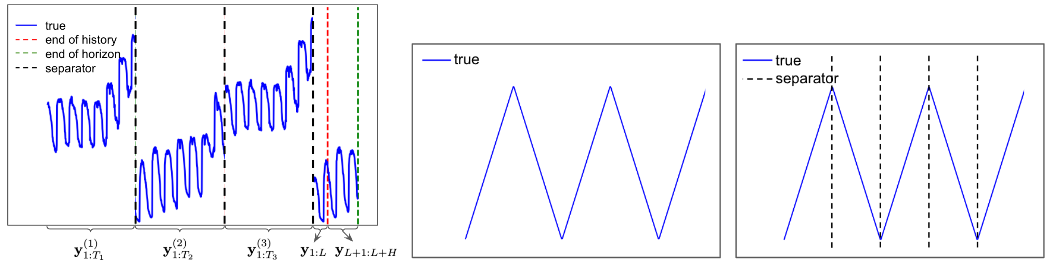
3 PROBLEM DEFINITION

Time-series foundation models aim to build a general purpose forecaster that can take in a past *history* of a target forecasting task, $\mathbf{y}_{1:L} = \{y_1, y_2, \cdots y_L\}$, where we look back L time-steps and map them to a forecast $\widehat{\mathbf{y}}_{L+1:L+H}$, for a horizon length of H. The aim is to have $\widehat{\mathbf{y}}_{L+1:L+H}$ as close as possible to the unseen future $\mathbf{y}_{L+1:L+H}$ according to some well defined error metric. Such a model can be thought of as a function,

$$g : \mathbf{y}_{1:L} \to \widehat{\mathbf{y}}_{L+1:L+H} \tag{1}$$

which is capable for handling different values of L and H.

In this work, we aim to further enhance the abilities of such models by enriching their context. In addition to the target task's history $\mathbf{y}_{1:L}$, we add up to $n-1$ *in-context examples* of the form $\{\mathbf{y}^{(1)}_{1:T_1}, \mathbf{y}^{(2)}_{1:T_2}, \cdots \mathbf{y}^{(n-1)}_{1:T_{n-1}}\}$ that can represent the past time-points of other related time-series (with possibly varying lengths $T_1, \cdots, T_{n-1}$). In the case of our motivating example of highway traffic prediction, $\mathbf{y}_{1:L}$ is a week of hourly traffic data on that highway, and $\{\mathbf{y}^{(1)}_{1:T_1}, \mathbf{y}^{(2)}_{1:T_2}, \cdots \mathbf{y}^{(n-1)}_{1:T_{n-1}}\}$ are traffic data on $n-1$ nearby highways.



(a) An example prediction task. (b) A triangular wave. (c) Multiple linear trends.

Figure 2: In (a), we show an example prediction task $(\{\mathbf{y}^{(1)}_{1:T_1}, \mathbf{y}^{(2)}_{1:T_2}, \mathbf{y}^{(3)}_{1:T_3}, \mathbf{y}_{1:L}\}, \mathbf{y}_{L+1:L+H})$. The three black dashed lines (separators) separate the four in-context examples, where the goal is to predict the horizon $\mathbf{y}_{L+1:L+H}$ of the example $\mathbf{y}_{1:L}$. In (b) and (c), we show that naïvely concatenating in-context examples together without separators can confuse the model: multiple linear trends look like a triangular wave if concatenated naïvely.

Therefore, the enhanced forecasting problem is aimed at training a model f,

$$f : \left(\mathbf{y}^{(1)}_{1:T_1}, \mathbf{y}^{(2)}_{1:T_2}, \cdots \mathbf{y}^{(n-1)}_{1:T_{n-1}}, \mathbf{y}_{1:L}\right) \to \widehat{\mathbf{y}}_{L+1:L+H}. \tag{2}$$

As before, our time-series foundation model should be able to handle different values of L and H. Additionally it should be able to support any number of in-context examples $(n-1)$ ranging from zero to a maximum value. With some abuse of notation, let us index the target task's forecasting history and horizon as the n-th example i.e. $\mathbf{y}^{(n)}_{1:T_n} := \mathbf{y}_{1:L+H}$, where $T_n = L + H$. Therefore, our decoder-only model will work with n examples of the form $\{\mathbf{y}^{(1)}_{1:T_1}, \mathbf{y}^{(2)}_{1:T_2}, \cdots, \mathbf{y}^{(n)}_{1:T_n}\}$ which are drawn from related time-series. Henceforth, we will refer to $\{\mathbf{y}^{(i)}_{1:T_i}\}_{i=1}^{n}$ as the *context* (synonymous with prompt) supplied to the model.

4 Model Architecture

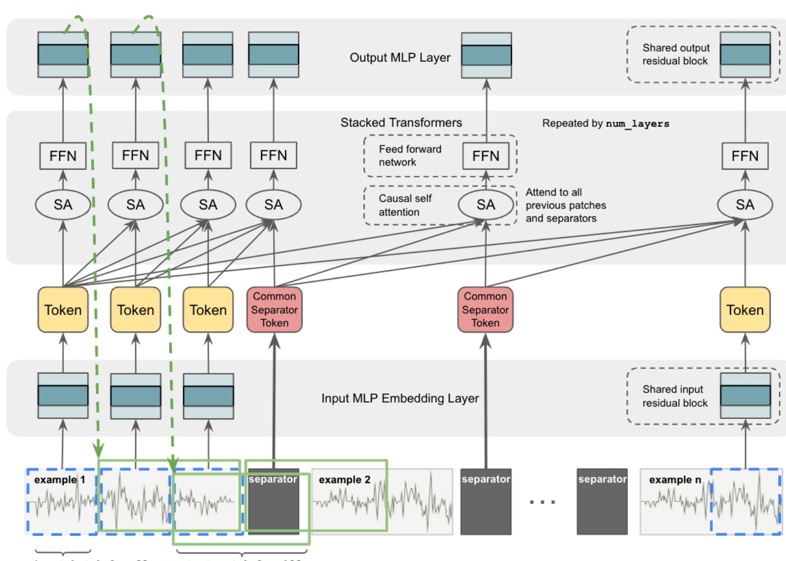


Figure 3: Our decoder-only architecture for time-series prediction with in-context examples.

Motivated by the strong zero-shot performance achieved by stacked transformer models in decoder-only mode for time-series forecasting, we propose to adapt a base TimesFM model (Das et al., 2024) to leverage the additional information available via in-context examples. In particular, we pretrain TimesFM in its original fashion to obtain a base checkpoint. We then modify the model architecture and continue pretraining from the base checkpoint using training data with in-context examples (we call this phase *continued pretraining*) to obtain a new pretrained foundation model *TimesFM-ICF*. The base TimesFM checkpoint that we start from will be referred to as *TimesFM (base)*.

Adapting their model architecture to make use of the in-context examples is somewhat delicate, and requires modifications to the original model. A depiction of our proposed model architecture is given in Figure 3. As in their model, our model partitions each example into non-overlapping input *patches*, and uses a shared input residual block (a one-hidden layer perceptron with skip connection, see Das et al. (2023)), to embed each patch as a token before feeding the tokens into the stacked transformers in a decoder-only fashion. The output embeddings are mapped to the next output patches via another shared output residual block.

To teach the model to use the new in-context examples, we adapt the original TimesFM architecture to better handle (1) the in-context example separators, (2) the cross-example attention, and (3) the positional encoding. Despite these changes, we are still able to leverage the TimesFM (base) checkpoint, which was pretrained for forecasting given just the history of the target time-series. We describe the key details of our architecture design below.

4.1 Separators for In-context examples

Our context window contains in-context examples from different time-series. Hence the model needs to be able to separate these, since naïve concatenation can confuse the model. Consider the example in Figure 2. If we naïvely concatenate multiple in-context examples (e.g., linear trends, Figure 2c) together, then the combination of these trends may appear to the model as an entirely different time-series (e.g., a triangle wave, Figure 2b). Therefore, we choose to insert a common learnable separator token after each in-context example. We visually depict these separators as the dashed lines in Figure 2c. When feeding examples to the decoder, we sequentially pass each tokenized patch of each time-series example to the model, followed by the separator token at the end of an example. This process is depicted in Figure 3.

4.2 Cross-example Attention

In order to allow our model to distinguish between different in-context examples, we allow the transformer to attend (causally) to all previous patches including the separator tokens. Note that, if the model did not attend to the separator tokens, then we could never hope to distinguish between the two scenarios from Figure 2b and Figure 2c. By attending to the previous separator tokens, the model can potentially distinguish how many in-context examples have been processed so far.

Although at the input to the stacked transformer we use a common separator token to separate the examples, the output tokens corresponding to the positions of these separator tokens can play a much more nuanced role as we proceed through the subsequent transformer layers. As the output tokens corresponding to these separator tokens causally attend to all previous tokens, after several transformer layers these tokens can, for instance, potentially summarize the information in all the patches corresponding to their example and/or convey the separation boundaries of the different in-context examples to the model.

4.3 Positional Encoding

Based on the findings in Haviv et al. (2022), we create the pretrained TimesFM (base) checkpoint with No Positional Encodings (NoPE), in contrast to the absolute positional encodings (Vaswani et al., 2017) used in the original TimesFM model. We note that we can achieve the same accuracy reported in the original TimesFM paper without using any positional encodings. Indeed it has been hypothesized in Haviv et al. (2022) that the presence of causal attention itself can encode positional information when there are more than one stacked transformer layers.

The advantages of NoPE for our continued pretraining are two fold: (i) NoPE models usually have better length generalization, which is particularly important here since we increase the prompt length by adding in-context examples to the context (ii) If we use the original absolute positional encodings used in (Das et al., 2024), the meaning of these positional encodings in the base model would be different from their meaning during the continued pretraining with in-context examples. This could cause problems for the continued pretraining phase.

4.4 Overall Model

Since our model builds upon the TimesFM architecture (Das et al., 2024), we introduce a similar notation style for ease of exposition. The model processes in-context examples in the following fashion. Starting with an example input $\{\mathbf{y}^{(1)}_{1:T_1}, \ldots, \mathbf{y}^{(n)}_{1:T_n}\}$, each example $\mathbf{y}^{(i)}_{1:T_i}$ is partitioned into input patches of length p:

$$\tilde{\mathbf{y}}_j^{(i)} = \mathbf{y}^{(i)}_{(p-1)j+1:pj} \quad \forall j \in [\lceil T_i/p \rceil] \text{ and } i \in [n].$$

As in (Das et al., 2024), our model takes an additional padding mask $\mathbf{m}^{(i)}_{1:T_i}$ to ensure that it makes good predictions on time-series which are not a multiple of the patch length p. Given these patches and masks, we feed each patch $\tilde{\mathbf{y}}_j^{(i)}$ through a common MLP embedding layer to obtain tokens:

$$\mathbf{t}_j^{(i)} = \mathsf{InputResidualLayer}(\tilde{\mathbf{y}}_j^{(i)} \odot (1 - \tilde{\mathbf{m}}_j^{(i)})) \quad \forall j \in [\lceil T_i/p \rceil] \text{ and } i \in [n].$$

We will slightly abuse notation by denoting the separator token $\boldsymbol{\sigma}$ as $\mathbf{t}^{(i)}_{\lceil T_i/p \rceil + 1} = \boldsymbol{\sigma}$, and let the mask for the separator token $\tilde{\mathbf{m}}^{(i)}_{\lceil T_i/p \rceil + 1} = \mathbf{0}$ (i.e., the separator tokens are never masked). After tokenizing the input patches, we feed the tokens, together with a learnable separator token $\boldsymbol{\sigma}$, autoregressively to the stacked transformer layers in decoder-only mode. We take $\dot{m}_j^{(i)}$ to be the last entry of $\tilde{\mathbf{m}}_j^{(i)}$[2], and denote the sequence of token/mask pairs corresponding to example i as

$$\tilde{\mathbf{t}}^{(i)}_{1:j} = ((\mathbf{t}_1^{(i)}, \dot{m}_1^{(i)}), \ldots, (\mathbf{t}_j^{(i)}, \dot{m}_j^{(i)})) \quad \forall j \in [\lceil T_i/p \rceil] \text{ and } i \in [n].$$

Then, the output of the stacked transformer layer for token $\mathbf{t}_j^{(i)}$ can be described as:

$$\mathbf{o}_j^{(i)} = \mathsf{StackedTransformer}(\tilde{\mathbf{t}}^{(1)}_{1:\lceil T_1/p \rceil}, \tilde{\boldsymbol{\sigma}}, \ldots, \tilde{\mathbf{t}}^{(i-1)}_{1:\lceil T_i/p \rceil}, \tilde{\boldsymbol{\sigma}}, \tilde{\mathbf{t}}^{(i)}_{1:j}) \qquad \forall j \in [\lceil T_i/p \rceil] \text{ and } i \in [n].$$

Finally, we feed the outputs $\mathbf{o}_j^{(i)}$ from the stacked transformer through a residual block to obtain the predicted time-series:

$$\widehat{\mathbf{y}}^{(i)}_{pj+1:pj+h} = \mathsf{OutputResidualLayer}(\mathbf{o}_j^{(i)}).$$

This corresponds to the model's prediction of the next h steps (output patch length) of $\mathbf{y}^{(i)}_{pj+1:pj+h}$.

4.5 Loss Function

Similar to (Das et al., 2024), we use Mean Squared Error (MSE) as our point forecast loss.

$$\mathsf{TrainLossPerContext} = \frac{1}{\sum_{i=1}^{n} \lceil T_i/p \rceil} \sum_{i=1}^{n} \sum_{j=1}^{\lceil T_i/p \rceil} \|\widehat{\mathbf{y}}^{(i)}_{pj+1:pj+h} - \mathbf{y}^{(i)}_{pj+1:pj+h}\|^2.$$

5 Pretraining Data

As mentioned before, we start with TimesFM (base) which was pretrained on a diverse corpus of about 400B time-points. Please see Table 1 in Appendix A.1 and Das et al. (2024) for more details on the datasets. We then continue pretraining it on training data containing in-context examples.

Context Generation. We convert individual datasets to generate *contexts* with in-context examples that the model sees during the continued pretraining. Recall that the original TimesFM model is

[2] Intuitively, $\dot{m}_j^{(i)}$ indicates whether or not patch $\tilde{\mathbf{y}}_j^{(i)}$ is masked from the right. We attend only to patches which are not padded from the right, and have at least one unpadded values (see Appendix A.1)

trained up to a maximum history length of $L_{max} = 512$. During the training of TimesFM (base) a time-series of length $T = L_{max} + h$ is loaded for back propagation where $h = 128$ is the output patch length. Therefore, we choose T as the maximum length of our n in-context examples. For any time-series in a particular dataset, we use windowing with a shift of 1 to generate examples of length T i.e. for a time-series $\mathbf{y}_{1:M}$ the possibles examples are $\{\mathbf{y}_{1:T}, \mathbf{y}_{2:T+1}, \cdots \mathbf{y}_{M-T+1:M}\}$. For time-series that are less than T in length, we generate padded examples as detailed in Appendix A.1. Now these examples are packed in groups of n to form the context. We consider two kinds of grouping:

1. *Times-series level:* For a long time-series, we can split the original time-series into shorter time-series examples, each of length T, then select n of those shorter examples to form the context$\{\mathbf{y}_{1:T}^{(i)}\}_{i=1}^{n}$ for the original time-series.
2. *Dataset level:* For each dataset, we can group any n segments of length T from any of the time-series in that dataset, to form a context. For instance, a set of n segments from any of the time-series from the Electricity dataset could be grouped to form a context $\{\mathbf{y}_{1:T}^{(i)}\}_{i=1}^{n}$.

Both time-series level and dataset level groupings guarantee that the grouped examples have similar patterns to borrow from each other.

Dataset Mixture. We choose all datasets in Table 1 other than the four Wiki datasets to generate in-context examples for continued training. The Wiki datasets contain millions of time-series that correspond to a wide variety of articles, which need not be related to each other. In fact the Wiki dataset can be potentially clustered into groups of related articles, and the time-series for each group could be deemed to form a separate dataset. But we leave such preprocessing of the Wiki dataset for future work and leave these datasets out of our continued pretraining.

For the remaining datasets, we set the number of examples in each context as $n = 50$ and generate contexts from both time-series level and dataset level grouping. Note that if all the time-series in a dataset have a total of N examples, then generating all $\binom{N}{n}$ such contexts is intractable. Therefore, we randomly generate $20N$ such groups of n examples as our training contexts.

Following the original TimesFM paper, the training data loader samples 90% real data and 10% synthetic, with the real data mixture providing equal weights to the groups: hourly + sub-hourly, daily, weekly, and monthly datasets. Moreover, we provide equal weights to the two kinds of examples i.e., time-series level and dataset level.

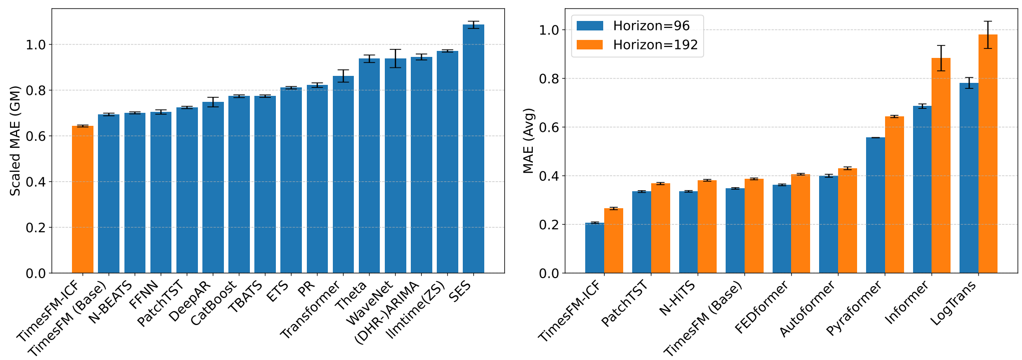
6 Experimental Results

Following prior time-series foundation model papers like (Das et al., 2024; Gruver et al., 2023), we compare the zero-shot performance of our proposal with that of supervised models, statistical models trained per dataset as well as other zero-shot models. Similar to prior works, we report our results on a subset of Monash datasets (Godahewa et al., 2021) and the ETT datasets (Zhou et al., 2021) that have not been seen by our model or the TimesFM (base) model.

6.1 Out-of-domain Forecasting on Monash

Monash archive (Godahewa et al., 2021) is a collection of 30 datasets of different training and prediction lengths that covers granularities ranging from minutes to years and domains including finance, demand forecasting, weather and traffic. The archive reports four official metrics for several statistical baselines such as Exponential Smoothing(ETS) and ARIMA, as well as supervised ML baselines like CatBoost (Prokhorenkova et al., 2018), DeepAR (Salinas et al., 2020) and WaveNet (Oord et al., 2016). We report our results on the 18 datasets that were also considered for zero-shot forecasting in Das et al. (2024). We provide more details in Appendix A.2.1.

The datasets contain time-series with vastly different scales and therefore we cannot aggregate the raw metrics. Therefore, following prior works (Gruver et al., 2023; Das et al., 2024) we calculate the MAE for all methods and normalize them by the MAE achieved by a naive baseline that just repeats the last time-point's value in the history for the whole horizon. Then we report the Geometric Mean of these scaled MAE values across all datasets. Note that when dealing with normalized metrics it is better to report the Geometric Mean (Fleming & Wallace, 1986). We borrow the official numbers for



(a) Monash

(b) ETT

Figure 4: In (a), we report the geometric mean of scaled MAE for Monash datasets. We include all official Monash baselines as well as TimesFM-ICF, TimesFM (base). TimesFM (base) yields a 7% improvement over the next best baseline. We also report one standard error similar to (Das et al., 2024). In (b), we report the average MAE numbers for 4 datasets ETTh1, ETTh2, ETTm1 and ETTm2. Similar to prior work like (Nie et al., 2022), the numbers are reported for rolling validation over the test split which makes up the last 1/5th of time-points in each dataset. We also report one standard error. TimesFM-ICF yields a marked improvement of at least 25% over other baselines.

all baselines from (Godahewa et al., 2021) except for TimesFM (base)(we evaluate our base model) and LLMTime (we use the precomputed output from the original paper).

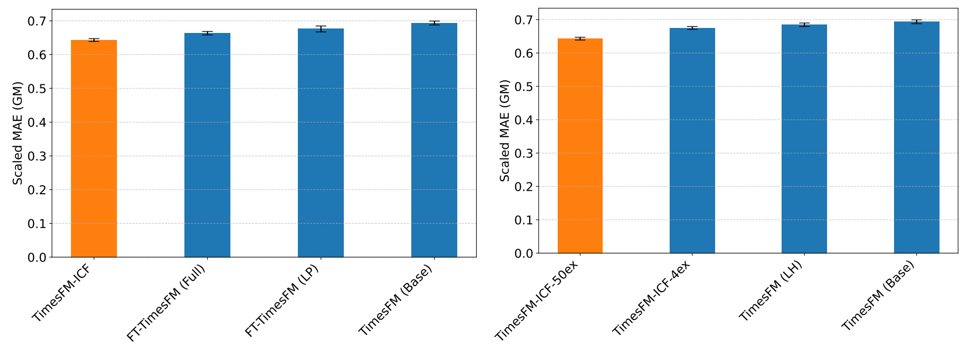
The results are summarized in Figure 4a. We can see that TimesFM-ICF performs the best followed by TimesFM (base) and N-BEATS. It can be seen that TimesFM-ICF yields a 7% improvement over the closest supervised baseline (N-BEATS), which has been trained per dataset. More importantly, we obtain a 7% improvement over TimesFM (base), thus showing the value of in-context fine-tuning for time-series foundation models. Note that TimesFM-ICF, TimesFM (base) and LLMTime are the only zero-shot methods in this benchmark.

6.2 Out-of-domain Forecasting on ETT

A group of long horizon datasets have been commonly used for benchmarking (mainly) transformer based deep learning algorithms starting from (Zhou et al., 2021). Some of the datasets in these benchmarks are in our pretraining datasets (like Electricity and Traffic). Therefore, for the interest of zero-shot evaluation we use the 4 Electricity Transformer Temperature (ETT) datasets, specifically ETTh1, ETTh2 (hourly) and ETTm1, ETTm2 (15 min).

In terms of baselines, following (Das et al., 2024), we compare against Informer (Zhou et al., 2021) and subsequent works like Pyraformer (Liu et al., 2021), FEDFormer (Zhou et al., 2022), PatchTST (Nie et al., 2022). We also compare with N-HiTS (Challu et al., 2023) which yields an improvement over N-BEATS (Oreshkin et al., 2019) for these datasets. Similar to Das et al. (2024), we focus on the task of predicting horizon lengths 96, 192 given a history of 512 time-steps. We provide rolling validation numbers for the test time-period which consists the last 1/5-th of the time-points. This is standard for these benchmarks (Nie et al., 2022), where the datasets are split into train:validation:test in the ratio 7:1:2.

We present the MAE obtained for horizon lengths 96 and 192 averaged over the 4 datasets in Figure 4b. Note that since the MAE is computed on scaled datasets in this benchmark (Zhou et al., 2021), we can directly report the arithmetic mean across datasets. We see that TimesFM-ICF yields a marked improvement of more than 25% on mean MAE over the nearest baseline. PatchTST, N-HiTS and TimesFM (base) perform similarly and are much better than the other baselines. In this case, all the datasets have in-context examples with enough time-points to cover T time-steps, unlike in Monash where 9 out of 18 datasets have time-series of length less than 512 time-steps. Therefore, we can see more value from in-context fine-tuning. We provide a more fine-grained analysis with the number of in-context examples on ETTh datasets in Sections 6.4.1 and 6.4.2.



(a) Fine-tuning per Dataset

(b) Longer History

Figure 5: In (a), we report the geometric mean of scaled MAE across the Monash datasets. FT-TimesFM corresponds to fine-tuning the original TimesFM (base) model per dataset either (1) Full fine-tune or (2) Linear Probed (see Section 6.3). We can see that TimesFM-ICF is clearly better than FT-TimesFM models even though it is zero-shot. In (b), we compare TimesFM-ICF with a base TimesFM model trained with a longer maximum supported history of 2048 time-points. We can see that TimesFM-ICF performs better than TimesFM (LH) in terms of the scaled MAE (GM) metric on Monash. This is further discussed in Section 6.4.2.

6.3 Comparison with Fine-tuning per dataset

One of the main motivations of this work was to see whether we can recover the gains from fine-tuning foundation models on the target domain without doing any gradient updates. Therefore, in this section, we compare against a very strong baseline: for every dataset in our Monash benchmark from Section 6.1 we fine-tune the TimesFM (base) model on the training set and evaluate it on the test set. We do two kinds of fine-tuning (1) we update all the model weights which we will refer to as FT-TimesFM (Full) (2) we hold all the transformer layer fixed while only the input and output residual blocks are fine-tuned, which we will refer to as FT-TimesFM (LP) [3].

The aggregated scaled MAE numbers are presented in Figure 5a. TimesFM-ICF actually yields close to 3% improvement over FT-TimesFM (Full) which is already a 4% improvement over TimesFM (base). This shows that in-context fine-tuning can sometimes be better than per-dataset fine-tuning, even though we do not perform any gradient updates! The advantages of our method are further highlighted by the fact the total time required for fine-tuning on all datasets is *115 minutes* (not including job scheduling times) for the cheaper FT-TimesFM (LP) method while the total inference time for TimesFM-ICF is merely 4 minutes [4].

While this is surprising, we believe that one reason could be that in many of the smaller datasets in Monash, fine-tuning the weights of a foundation model can actually lead to catastrophic forgetting of the learnt patterns which is also observed in LLMs (Luo et al., 2023). Indeed on the smaller datasets like tourism yearly, bitcoin and us births, TimesFM-ICF is better than FT-TimesFM and vice versa on larger datasets like Australian electricity demand. We provide per dataset metrics and more details about the fine-tuning in Section A.5.

6.4 Ablation

We now present two important ablation studies that justify the benefits of in-context examples, as well as the advantages of our technique versus others like training longer-history models.

6.4.1 Number of examples

The number of in-context examples is an important consideration that dictates the performance of our model. We perform an ablation where we vary the number of in-context examples from 1 to the

[3]LP is meant to stand for Linear Probing even though here we are tuning the MLP layers.

[4]The inference numbers are reported on TPUv5e with 8 tensor cores.

maximum during our training i.e. $n = 50$. The corresponding results are reported on the ETTh test set in Figure 6. We can see a monotonic increase in performance with more in-context examples. We chose to perform this ablation on the ETT datasets since, unlike the Monash datasets, all time-series are big enough to provide complete in-context examples of length T, which makes it easier to perform this experiment.

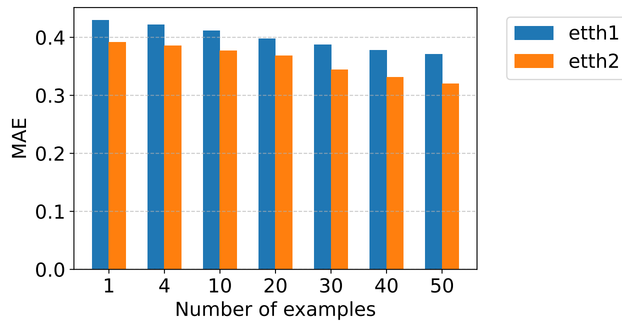


Figure 6: The performance of the model gets better with increasing number of in-context examples on ETTh1 and ETTh2.

6.4.2 Longer History

In this section, we compare the performance of TimesFM-ICF with a version of TimesFM (base) trained with a longer history $L = 2048$ which we will refer to as TimesFM (LH). We provide the aggregate scaled MAE on Monash datasets in Figure 5b where we include two versions of TimesFM-ICF, one with 4 in-context examples (TimesFM-ICF-4ex) and one with 50 in-context examples (TimesFM-ICF-50ex). We can see that TimesFM (LH) yields a modest 1% improvement over TimesFM (base) (which has a maximum history of 512) while TimesFM-ICF-50ex yields a 7% improvement. Even TimesFM-ICF-4ex which uses the same total context length for all in-context examples as TimesFM (LH) is 3% better than the baseline.

This shows that our technique of in-context fine-tuning can be more effective than training a longer history model, especially when there is a mix of short-history and long-history time-series. This is because, for in-context fine-tuning, many short time-series can be packed as in-context examples inside the context, while for the case of usual long history training such time-series will just be padded and most of the context is wasted. As shown in the detailed results in Appendix A.2, the long history model performs better on longer datasets like australian electricity demand, but degrades on shorter datasets like cif and tourism yearly.

7 Conclusion

In this paper, we introduce and study a methodology for in-context fine-tuning of a time-series foundation model for forecasting. In particular, we start with a base foundation model and adapt it to be able to effectively utilize, at inference time, not just the history of the target time-series for forecasting, but also in-context examples from related time-series. Our results show that in-context fine-tuning can lead to significantly better zero-shot performance on popular forecasting benchmarks compared to the base foundation model and state-of-the-art supervised models. Furthermore, it even outperforms a version of the base foundation model that is explicitly fine-tuned on the target domain.

While we have chosen a specific base time-series foundation model (TimesFM) for our in-context fine-tuning approach, it would be an interesting direction of future work to study these adaptations for other base foundation models. It would also be interesting to study better forms of relative positional encodings specifically designed for handling in-context examples and length generalization.

REFERENCES

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

George EP Box and Gwilym M Jenkins. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968.

Tom B Brown et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020.

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. NHITS: Neural Hierarchical Interpolation for Time Series forecasting. In *The Association for the Advancement of Artificial Intelligence Conference 2023 (AAAI 2023)*, 2023. URL `https://arxiv.org/abs/2201.12886`.

Ching Chang, Wen-Chih Peng, and Tien-Fu Chen. Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms. *arXiv preprint arXiv:2308.08469*, 2023.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. Meta-learning via language model in-context tuning. *Association for Computational Linguistics*, 2022.

Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan K Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with TiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL `https://openreview.net/forum?id=pCbC3aQB5W`.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *International conference on machine learning*, 2024.

Philip J Fleming and John J Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *Communications of the ACM*, 29(3):218–221, 1986.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Azul Garza and Max Mergenthaler-Canseco. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I Webb, Rob J Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*, 2021.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. *arXiv preprint arXiv:2310.07820*, 2023.

Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. *arXiv preprint arXiv:2203.16634*, 2022.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*, 2021.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. *Association for Computational Linguistics*, 2022a.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*, 2022b.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *International conference on learning representations*, 2022.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2019.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Victoria Lin, Noah A Smith, Luke Zettlemoyer, Scott Yih, and Mike Lewis. In-context pretraining: Language modeling beyond document boundaries. *arXiv preprint arXiv:2310.10638*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.

Jingyuan Wang, Jiawei Jiang, Wenjun Jiang, Chengkai Han, and Wayne Xin Zhao. Towards efficient and comprehensive urban spatial-temporal prediction: A unified library and performance benchmark. *arXiv preprint arXiv:2304.14343*, 2023.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *ICLR*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022b.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. *International conference on machine learning*, 2024.

Xi Ye and Greg Durrett. The unreliability of explanations in few-shot prompting for textual reasoning. *Advances in neural information processing systems*, 35:30378–30392, 2022.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, 2021.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pp. 27268–27286. PMLR, 2022.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained lm. *arXiv preprint arXiv:2302.11939*, 2023.

A Appendix

A.1 More Details about our Model and Baselines

Monash Baselines. For the results on Monash datasets, we borrow the official numbers from (Godahewa et al., 2021). For LLMTime (Gruver et al., 2023) we use the pre-computed outputs supplied by the original authors.

We also add the PatchTST (Nie et al., 2022) as a baseline for this benchmark because it is the best performing baseline (only worse than our models) in the ETT datasets. For this model we use the hyperparameters used by original paper for the ETTh datasets [5].

ETT Baselines. On the ETT datasets, the baseline numbers (except TimesFM (base)) are borrowed from the official numbers reported in Table 2 of (Das et al., 2023). We evaluate the base model, TimesFM (base) as well as our method in a rolling validation manner on the test splits to obtain the corresponding metrics.

TimesFM (base). Following Das et al. (2024), we train a 200M model with 16 attention heads, 20 layers, a input patch length of 32 and output patch length of 128. The model dimension is set to 1280. We use the learning rate schedule in (Vaswani et al., 2017) with peak learning rate of $5e-4$. The hidden dims of both the residual block and the FFN in the transformer layers are set as the same as model dimensions. We keep layer norm in transformer layers but not in the residual blocks. The only difference between the model in Das et al. (2024) and our base model is that we use NoPE instead of teh absolute positional encoding (Vaswani et al., 2017). As we have mentioned before, this leads to no loss in accuracy while being easier to extend to our in-context fine-tuning setting.

Fine-tuning Per Dataset. On the Monash benchmark, we also compare with TimesFM (base) fine-tuned on the train set for every dataset and the forecasting on the corresponding test set. For all our fine-tuning runs, we use a batch size of 16 and a maximum of 10k iterations. Note that this means that the fine-tuned model will see many more training examples than the in-context examples given to our model. For the fine-tuning runs, we use the same decoder only loss function that was used in the original pretraining of TimesFM (base), the only difference is that the training is not restricted to the training set of one dataset. We do two kinds of fine tuning:

- *Full:* All weights in the model are updated during fine-tuning.
- *Linear Probing (LP):* We hold the transformer weights fixed and only update the parameters in the input and output residual blocks.

TimesFM-ICF. We continue to train TimesFM-ICF model from TimesFM (base). Therefore, most of the parameters in the model remain the same. Here, are the key training details that are unique to TimesFM-ICF:

- *Separator Token:* We have a trainable separator token that is also updated during the continued pretraining. The token is nothing but a learnt embedding whose dimension is equal to the model dimension i.e. 1280 in our case.
- *Number of Examples:* We use a maximum of $n = 50$ in-context examples for each context during training.
- *Padding:* In short datasets like M4 yearly and quarterly, each time-series might have number of time-points much less than $T = 640$. Sometimes the number of time-points are even less than our input patch length $p = 32$. For such cases, a whole time-series can fit into one of the n examples and they are preprocessed in the following manner:
  - If the length of the time-series l is less than p, we left pad with k padding time-points such that $p < k + l < 2p$. This is because we want the decoder only model to predict something meaningful for the second patch after seeing the first patch and if not, is penalized by the loss on the second patch. If the $l > p$, we do not need to perform this left padding.
  - Lastly, we right pad such that the length of the total padded example is $T = 640$.

[5] https://github.com/yuqinie98/PatchTST/blob/main/PatchTST_supervised/scripts/PatchTST/etth1.sh

– Note that the last patch in such examples would be padded from the right i.e., they will have real time-series values for the first few points and padding for the rest. We make sure that such incomplete from the right patches are not attended by subsequent tokens belonging to examples coming after.

The pretraining datasets are detailed in Table 1.

Table 1: List of datasets included in pretraining. All datasets except the Wiki datasets are also repurposed for continued pretraining with in-context examples.

Dataset	Granularity	# Time series	# Time points
Synthetic		3,000,000	6,144,000,000
Electricity	Hourly	321	8,443,584
Traffic	Hourly	862	15,122,928
Weather (Zhou et al., 2021)	10 Min	42	2,213,232
Favorita Sales	Daily	111,840	139,179,538
LibCity (Wang et al., 2023)	15 Min	6,159	34,253,622
M4 hourly	Hourly	414	353,500
M4 daily	Daily	4,227	9,964,658
M4 monthly	Monthly	48,000	10,382,411
M4 quarterly	Quarterly	24,000	2,214,108
M4 yearly	Yearly	22,739	840,644
Wiki hourly	Hourly	5,608,693	239,110,787,496
Wiki daily	Daily	68,448,204	115,143,501,240
Wiki weekly	Weekly	66,579,850	16,414,251,948
Wiki monthly	Monthly	63,151,306	3,789,760,907
Trends hourly	Hourly	22,435	393,043,680
Trends daily	Daily	22,435	122,921,365
Trends weekly	Weekly	22,435	16,585,438
Trends monthly	Monthly	22,435	3,821,760

A.2 DETAILED METRICS ON MONASH AND ETT

A.2.1 MONASH

Table 2 presents the per-dataset MAE numbers of TimesFM-ICF against other supervised and zero-shot methods on Monash.

Table 2: MAE of TimesFM-ICF against other supervised and zero-shot methods on Monash.

	(DHR-)ARIMA	CatBoost	DeepAR	ETS	FFNN	N-BEATS	Naive	PR	PatchTST	SES	TBATS	Theta	TimesFM (Base)	TimesFM-ICF	Transformer	WaveNet	llmtime(ZS)
australian electricity demand	1045.92	241.77	302.41	1282.99	258.76	213.83	659.60	247.18	248.35	659.60	370.74	665.04	426.12	338.98	231.45	227.50	459.96
bitcoin	3.62e+18	1.93e+18	1.95e+18	1.10e+18	1.45e+18	1.06e+18	7.78e+17	6.66e+17	1.84e+18	5.33e+18	9.90e+17	5.33e+18	1.90e+18	9.58e+17	2.61e+18	2.46e+18	1.75e+18
fred md	2957.11	2475.68	4264.36	2041.42	2339.57	2557.80	2825.67	8921.94	2005.86	2798.22	1989.97	3492.84	2514.63	2021.52	4666.04	2508.40	2013.49
nn5 daily	4.41	4.22	3.94	3.72	4.06	4.92	8.26	5.47	5.56	6.63	3.70	3.80	3.57	3.74	4.16	3.97	9.39
pedestrian counts	635.16	43.41	44.78	216.50	46.41	66.84	170.88	44.18	45.90	170.87	222.38	170.94	42.55	43.71	47.29	46.46	70.20
saugeenday	22.38	21.28	23.51	30.69	22.98	27.92	21.50	25.24	21.52	21.50	22.26	21.49	30.54	24.91	28.06	22.17	28.63
traffic hourly	0.04	0.02	0.01	0.03	0.01	0.02	0.03	0.02	0.01	0.03	0.04	0.03	0.01	0.01	0.01	0.02	0.03
us births	526.33	441.70	424.93	419.73	557.87	422.00	1152.67	574.93	556.23	1192.20	399.00	586.93	446.49	399.74	452.87	504.40	459.43
weather	2.45	2.51	2.02	2.35	2.09	2.34	2.36	8.17	2.12	2.24	2.30	2.51	1.98	2.10	2.03	2.29	2.32
cif 2016	469059.49	603551.30	3200418.00	642421.42	1495923.44	679034.80	386526.37	563205.57	271198.00	581875.97	855578.40	714818.58	438028.90	647255.33	4057973.04	5998224.62	715086.33
covid deaths	85.77	475.15	201.98	85.59	144.14	158.81	353.71	347.98	246.55	353.71	96.29	321.32	124.86	113.78	408.66	1049.48	304.68
hospital	19.60	19.17	18.25	17.97	22.86	20.18	24.07	19.24	18.52	21.76	17.43	18.54	17.95	17.26	36.19	19.35	24.62
nn5 weekly	15.38	15.29	14.69	15.70	15.02	14.19	16.71	14.94	15.38	15.66	14.98	15.30	14.15	15.38	20.34	19.34	15.91
solar weekly	839.88	1513.49	721.59	1131.01	1050.84	1172.64	1729.41	1044.98	1525.59	1202.39	908.65	1210.83	1380.09	1424.71	576.35	1996.89	2049.09
tourism monthly	2536.77	2537.04	1871.69	2004.51	2022.21	2003.02	5636.83	2187.28	2587.16	5302.10	2940.08	2069.96	3406.55	2018.07	2146.98	2095.13	4724.94
tourism quarterly	10475.47	10267.97	9511.37	8925.52	8981.04	8640.56	15845.10	9092.58	13271.98	15014.19	9972.42	7656.49	9535.86	8202.19	9521.67	9137.12	14121.09
tourism yearly	95033.24	79567.22	71471.29	94818.89	79593.22	70951.80	99456.05	82682.97	99574.68	95579.23	94121.08	90653.60	75955.39	80365.15	74316.52	69905.47	140081.78
traffic weekly	1.22	1.17	1.18	1.14	1.15	1.11	1.19	1.13	1.15	1.12	1.17	1.13	1.06	1.09	1.42	1.20	1.17
Scaled MAE (GM)	0.945	0.773	0.748	0.810	0.704	0.700	1.000	0.822	0.724	1.086	0.774	0.937	0.694	0.643	0.862	0.938	0.971

A.2.2 ETT

Table 3 presents the MAE numbers of TimesFM-ICF against other methods on ETTh1, ETTh2, ETTm1 and ETTm2 respectively, with forecasting horizons of 96 and 192 respectively.

A.3 VARYING THE NUMBER OF IN-CONTEXT EXAMPLES

Table 4 and 5 shows the accuracy metric numbers of TimesFM-ICF on ETT and Monash respectively when different numbers of in-context examples are used.

Table 3: MAE of TimesFM-ICF against other baselines on ETT

		Autoformer	FEDformer	Informer	LogTrans	N-HiTS	PatchTST	Pyraformer	TimesFM (Base)	TimesFM-ICF
avg	96	0.400	0.362	0.686	0.781	0.336	0.335	0.556	0.348	0.207
	192	0.430	0.406	0.883	0.979	0.381	0.368	0.643	0.387	0.265
etth1	96	0.446	0.415	0.769	0.740	0.393	0.401	0.612	0.398	0.263
	192	0.457	0.446	0.786	0.824	0.436	0.429	0.681	0.427	0.330
etth2	96	0.368	0.374	0.952	1.197	0.345	0.337	0.597	0.350	0.206
	192	0.434	0.446	1.542	1.635	0.401	0.376	0.683	0.392	0.265
ettm1	96	0.492	0.390	0.560	0.546	0.350	0.346	0.510	0.369	0.207
	192	0.495	0.415	0.619	0.700	0.383	0.370	0.537	0.405	0.265
ettm2	96	0.293	0.271	0.462	0.642	0.255	0.256	0.507	0.274	0.152
	192	0.336	0.318	0.586	0.757	0.305	0.296	0.673	0.323	0.201

Table 4: MAE of TimesFM-ICF on ETT with different numbers of in-context examples.

Number of in-context examples	1	4	10	20	30	40	50
etth1	0.430	0.421	0.411	0.398	0.387	0.378	0.371
etth2	0.392	0.386	0.377	0.368	0.344	0.331	0.320
Average MAE	0.411	0.404	0.394	0.383	0.366	0.354	0.345

A.4 Long History

Table 6 and 7 show respectively the aggregated (geometric mean of scaled MAE) and the raw MAE numbers on Monash of different TimesFM models, with the focus on the comparison between TimesFM-ICF and TimesFM (LH) which is a long-2048-history TimesFM model. We compare TimesFM-ICFin two different modes: (i) 50ex, in which the model has access to 50 in-context examples, and (ii) 4ex, in which the model has access to only 4 in-context examples. In mode (ii), the aggregate length of all in-context examples is the same as the length of the history used by TimesFM (LH).

A.5 Fine-tuning per Dataset

Table 8, 9 and 10 present the detailed accuracy and timing metrics to compare TimesFM-ICF and FT-TimesFM on Monash. While TimesFM-ICF is more accurate, it is also significantly faster than straighforward fine-tuning on the target dataset. Both are results of the TimesFM-ICF's in-context learning capability.

A.6 Illustrative Examples

We illustrate visually in Figure 7 how in-context examples can help disambiguate the prediction tasks, by plotting the actual forecasts from TimesFM-ICF with and without the in-context examples. In the left two figures, the history is not sufficiently informative for the model to make an accurate prediction. By providing in-context examples together with this short history (see the right two figures), however, the model is able to make a more accurate forecast.

Table 5: Scaled MAE (GM) of TimesFM-ICF on Monash with different numbers of in-context examples.

Number of in-context examples	1	4	5	10	20	30	40	50
Scaled MAE (GM)	0.667	0.675	0.667	0.658	0.651	0.657	0.653	0.643

Table 6: Scaled MAE (GM) on Monash for long history length

	Scaled MAE (GM)
TimesFM-ICF-50ex	0.643
TimesFM-ICF-4ex	0.675
TimesFM (LH)	0.685
TimesFM (Base)	0.694

Table 7: Detailed breakdown of MAE on Monash for long history length

	TimesFM (LH)	TimesFM-ICF-4ex	TimesFM-ICF-50ex	TimesFM (Base)	naive
australian electricity demand	468.81	492.56	338.98	426.12	659.60
bitcoin	1.50e+18	1.32e+18	9.58e+17	1.90e+18	7.78e+17
cif 2016	709069.14	477038.11	647255.33	438028.90	386526.37
covid deaths	151.64	131.75	113.78	124.86	353.71
fred md	1519.00	1795.34	2021.52	2514.63	2825.67
hospital	17.64	17.23	17.26	17.95	24.07
nn5 daily	3.52	3.74	3.74	3.57	8.26
nn5 weekly	15.05	14.80	15.38	14.15	16.71
pedestrian counts	43.96	46.30	43.71	42.55	170.88
saugeenday	25.87	29.40	24.91	30.54	21.50
solar weekly	1211.10	1324.05	1424.71	1380.09	1729.41
tourism monthly	2629.16	2155.61	2018.07	3406.55	5636.83
tourism quarterly	8595.55	8952.65	8202.19	9535.86	15845.10
tourism yearly	89423.79	85239.54	80365.15	75955.39	99456.05
traffic hourly	0.01	0.01	0.01	0.01	0.03
traffic weekly	1.08	1.09	1.09	1.06	1.19
us births	473.87	447.00	399.74	446.49	1152.67
weather	1.87	2.12	2.10	1.98	2.36
Scaled MAE (GM)	0.685	0.675	0.643	0.694	1.000

Table 8: Monash Per-Dataset Fine-tune (scaled MAE)

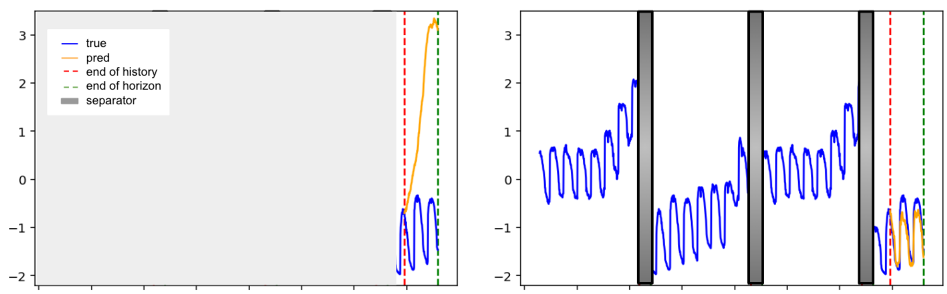
	scaled MAE (GM)
FT-TimesFM (Full)	0.663
FT-TimesFM (LP)	0.676
TimesFM-ICF	0.643
TimesFM (Base)	0.694

Table 9: MAE on Monash of TimesFM-ICF compared to models fine-tuned and evaluated on (the training and test set, respectively, within) each individual dataset within Monash

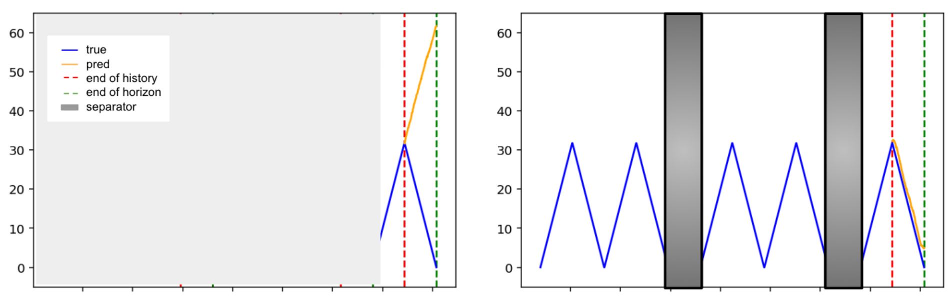
	FT-TimesFM (Full)	FT-TimesFM (LP)	TimesFM-ICF	TimesFM (Base)	naive
australian electricity demand	178.07	262.83	338.98	426.12	659.60
bitcoin	1.33e+18	1.43e+18	9.58e+17	1.90e+18	7.78e+17
cif 2016	724237.52	1344910.30	647255.33	438028.90	386526.37
covid deaths	181.89	85.12	113.78	124.86	353.71
fred md	2296.35	2330.96	2021.52	2514.63	2825.67
hospital	19.53	18.86	17.26	17.95	24.07
nn5 daily	3.42	3.37	3.74	3.57	8.26
nn5 weekly	15.24	15.02	15.38	14.15	16.71
pedestrian counts	41.80	40.88	43.71	42.55	170.88
saugeenday	22.07	25.22	24.91	30.54	21.50
solar weekly	882.09	1610.53	1424.71	1380.09	1729.41
tourism monthly	2469.08	2069.82	2018.07	3406.55	5636.83
tourism quarterly	10140.35	10725.62	8202.19	9535.86	15845.10
tourism yearly	88210.94	85915.69	80365.15	75955.39	99456.05
traffic hourly	0.02	0.01	0.01	0.01	0.03
traffic weekly	1.19	1.12	1.09	1.06	1.19
us births	405.81	397.24	399.74	446.49	1152.67
weather	1.81	1.84	2.10	1.98	2.36
Scaled MAE (GM)	0.663	0.676	0.643	0.694	1.000

Table 10: Timing breakdown (in minutes) of forecasting TimesFM-ICF compared to individually fine-tuning then evaluating models on a per-dataset basis in Monash

	FT-TimesFM (Full)	FT-TimesFM (LP)	TimesFM-ICF
australian electricity demand	6.350	2.370	0.048
bitcoin	9.600	4.620	0.053
cif 2016	8.610	4.230	0.069
covid deaths	26.470	9.520	0.178
fred md	10.310	6.020	0.077
hospital	15.720	3.610	0.347
nn5 daily	11.120	5.360	0.076
nn5 weekly	9.220	3.950	0.081
pedestrian counts	17.120	12.050	0.063
saugeenday	9.440	4.090	0.048
solar weekly	9.040	5.030	0.085
tourism monthly	6.780	4.120	0.209
tourism quarterly	11.200	6.140	0.226
tourism yearly	10.350	5.160	0.288
traffic hourly	20.250	16.920	0.413
traffic weekly	7.700	11.790	0.428
us births	10.190	5.580	0.047
weather	7.540	4.910	1.394
Total	207.010	115.470	4.130



(a) In-context examples help the history disambiguate between an increasing trend and an oscillating seasonality.



(b) In-context examples help the history disambiguate between an increasing linear trend and a triangular wave.

Figure 7: Two illustrative examples on how in-context examples can help disambiguate the prediction tasks, that likely patterns based solely on the history can get proved or disproved by the patterns from the in-context examples.