Only the Curve Shape Matters: Training Foundation Models for Zero-Shot Multivariate Time Series Forecasting through Next Curve Shape Prediction

Cheng Feng [1] **Long Huang** [1 2] **Denis Krompass** [1]

Abstract

We present General Time Transformer (GTT), an encoder-only style foundation model for zero-shot multivariate time series forecasting. GTT is pretrained on a large dataset of 200M high-quality time series samples spanning diverse domains. In our proposed framework, the task of multivariate time series forecasting is formulated as a channel-wise next curve shape prediction problem, where each time series sample is represented as a sequence of non-overlapping curve shapes with a unified numerical magnitude. GTT is trained to predict the next curve shape based on a window of past curve shapes in a channel-wise manner. Experimental results demonstrate that GTT exhibits superior zero-shot multivariate forecasting capabilities on unseen time series datasets, even surpassing state-of-the-art supervised baselines. Additionally, we investigate the impact of varying GTT model parameters and training dataset scales, observing that the scaling law also holds in the context of zero-shot multivariate time series forecasting.

1. Introduction

Time series forecasting, the task of predicting future values of one or multiple variables based on their historical values and other potentially relevant information, holds significant importance across diverse domains including manufacturing, traffic, healthcare, finance, and environmental science. In response to its practical significance, a large variety of time series forecasting methods have been developed. Earlier work includes classic statistical approaches such as ARIMA (Box & Jenkins, 1968; Durbin & Koopman, 2012), Exponential Smoothing (Hyndman et al., 2008) and VAR (Zivot & Wang, 2006), as well as those leverage deep sequential models like recurrent neural networks (RNNs) (Salinas et al., 2020) and convolutional neural networks (CNNs) (Borovykh et al., 2018). In recent years, two distinct directions have emerged regarding the utilization of deep neural networks for time series forecasting. On one hand, building on the success of the Transformer architecture (Vaswani et al., 2017) in natural language processing (NLP) and computer vision (CV), there has been a surge in leveraging Transformer-like architecture for time series forecasting. Examples include Pyraformer (Liu et al., 2021), LogTrans (Li et al., 2019), Informer (Zhou et al., 2021), Autoformer (Wu et al., 2021), FEDformer (Zhou et al., 2022), Crossformer (Zhang & Yan, 2022) and PatchTST (Nie et al., 2022), to name a few. On the other hand, there is also a different voice that simple multilayer perceptrons (MLP)-like models can achieve similar or even better time-series forecasting performance than sophisticated Transformer-based models (Zeng et al., 2023; Ekambaram et al., 2023). This discrepancy may be attributed to the fact that Transformers tend to overfit small datasets, and that the largest publicly available time series dataset is less than 10 GB (Godahewa et al., 2021), which is significantly smaller compared to those in NLP and CV domains.

In this work, inspired by the Transformer scaling successes in NLP and CV domains, we experiment with training a Transformer-based foundation model, which we term General Time Transformer (GTT), for zero-shot multivariate time series forecasting on a large dataset containing 200M high-quality time series samples collected from diverse domains. To overcome the challenges of dataset/distribution shift, as well as varying channel/variable[1] dimensions of time series samples across different domains, the task of multivariate time series forecasting is formulated as a channel-wise next curve shape prediction problem within our framework. Specifically, we do not introduce any time-series-specific inductive biases, but instead treat each time series sample as a sequence of non-overlapping curve shapes with a unified numerical magnitude. Each curve shape comprises M consecutive time points of a single variable. GTT is trained to use N preceding curve shapes as the context to pre-

[1]Siemens Technology [2]Tsinghua University. Correspondence to: Cheng Feng <cheng.feng@siemens.com>, Long Huang <huangl22@mails.tsinghua.edu.cn>, Denis Krompass <denis.krompass@siemens.com>.

[1]We will use the terms "channel" and "variable" interchangeably throughout the rest of the paper.

dict the next curve shape on a channel-wise basis. We adopt an encoder-only architecture for GTT with the fewest possible modifications to the standard Transformer. The only major modification we introduced is a cross-channel attention stage after the temporal attention stage in each multi-head self-attention block to capture cross-variate dependency between channels. GTT employs an auto-regressive approach to handle long-term forecasting tasks extend beyond M time steps.

GTT exhibits excellent zero-shot multivariate time series forecasting performance on various benchmark datasets, even outperforming state-of-the-art supervised baselines in several cases. We also demonstrate that GTT can achieve noticeably improved performance with cost-effective fine-tuning on target datasets. Moreover, in zero-shot univariate forecasting, GTT achieves higher prediction accuracy compared to other pretrained foundation models specifically designed for univariate time series forecasting. Additionally, we have conducted an investigation into the influence of different scales of GTT model parameters and training datasets, which reveals that the scaling law also holds in our context. Our codebase is available at https://github.com/cfeng783/GTT.

We summarize the main contributions of this paper as follows:

- We introduce a framework that formulates the task of cross-domain multivariate time series forecasting into a problem of predicting the next curve shape given a context window of past curve shapes in a unified numerical magnitude on a channel-wise basis. This framework lays a solid foundation for the development of large-scale foundation models for multivariate time series forecasting.
- Our experimental results demonstrate that foundational models for time series forecasting, trained on datasets of comparable size to those used in CV and NLP domains, can also achieve outstanding zero-shot forecasting capabilities.
- To the best of our knowledge, GTT is the first foundation model for zero-shot *multivariate* time series forecasting.

2. Related Work

In recent years, deep neural networks such as RNNs (Salinas et al., 2020) and dilated CNNs (Borovykh et al., 2017) have gained significant prominence as formidable contenders in the field of time-series forecasting, particularly when confronted with large training datasets. Empirical evidence has demonstrated their superiority over conventional statistical methods, including ARIMA and exponential smoothing via various forecasting competitions (Makridakis et al., 2022; Kopp et al., 2021).

More recently, drawing inspiration from the triumph of the Transformer architecture in the realms of NLP and CV, there has been a notable surge in the utilization of Transformer-like architectures for time series forecasting. The first group of Transformer variants focuses on the development of novel attention modules to reduce computational complexity for long time series. Pyraformer (Liu et al., 2021), LogTrans (Li et al., 2019), and Informer (Zhou et al., 2021) are notable examples within this category. The second group of Transformer variants places specific emphasis on leveraging time and frequency domain features. For example, Autoformer (Wu et al., 2021) introduces a seasonal-trend decomposition architecture, incorporating an auto-correlation mechanism to serve as an attention module. Recognizing that the point-wise attention of the Transformer architecture fails to capture overall characteristics of time series, FEDformer (Zhou et al., 2022) proposes a frequency-enhanced seasonal-trend decomposition method to better capture global properties of time series. Three variants closely aligned with our research are Crossformer (Zhang & Yan, 2022), iTransformer (Liu et al., 2023) and PatchTST (Nie et al., 2022). Crossformer and iTransformer, in particular, explicitly exploit cross-channel dependencies that play a vital role in achieving precise multivariate forecasting outcomes. PatchTST, on the other hand, employs patching techniques to enhance the local semantic information of input tokens of time series data.

Conversely, an alternative perspective suggests that simple MLP-like models may yield comparable, if not superior, performance in time-series forecasting when compared to sophisticated Transformer-based models (Zeng et al., 2023; Ekambaram et al., 2023). Our conjecture is rooted in the notion that Transformers tend to overfit small datasets. For instance, the largest publicly available dataset for time series analysis is less than 10 GB (Godahewa et al., 2021), which pales in comparison to the vast datasets used in the NLP and CV domains to train Transformers. Thus, to better leverage the powerful modelling ability of Transformers while mitigating the risk of overfitting, reprogramming or fine-tuning pretrained acoustic and large language models (LLMs) for time series forecasting becomes another promising option (Yang et al., 2021; Zhou et al., 2023; Jin et al., 2023; Chang et al., 2023). Meanwhile, there are also works which directly use LLMs for time series forecasting (Gruver et al., 2023).

The exploration of foundation models pretrained on large datasets for zero-shot time series forecasting remains relatively limited in comparison to the advancements made in NLP and CV fields. However, there have been some notable efforts recently. Among the examples are

ForecastPFN (Dooley et al., 2023), TimeGPT (Garza & Mergenthaler-Canseco, 2023), Lag-Llama (Rasul et al., 2023) and PreDcT (Das et al., 2023). ForecastPFN is a Transformer-based prior-data fitted network trained purely on synthetic data designed to mimic common time series patterns. TimeGPT is a Transformer-based time series forecasting model trained over 100B data points, with other data and model details remain unrevealed. Lag-Llama, a probabilistic time series forecast model adapted from the LlaMA (Touvron et al., 2023) architecture, is trained on a large collection of time series from the Monash Time Series Repository (Godahewa et al., 2021). PreDcT is a patched-decoder style model trained on 1B time points from Google Trends. GTT distinguishes itself from existing models through several notable differences. Firstly, our training data is much more diverse compared with ForecastPFN, LlaMA and PreDcT. Secondly, we utilize an encoder-only architecture instead of a decoder-only architecture, wherein the task of time series forecasting is approached as a problem of predicting the next curve shape in a unified numerical magnitude. Lastly, GTT incorporates a channel attention mechanism, specifically designed for multivariate time series forecasting, rather than focusing solely on univariate forecasting.

3. Problem Definition

We consider building a general purpose zero-shot multivariate time series forecaster that takes in a look-back window of L time points of a time-series and optionally their corresponding time features as context, and predicts the future H time points. Let $\mathbf{x}_{1:L}$ and $\mathbf{d}_{1:L}$ be the context time series and corresponding time feature values, GTT is a function to predict $\hat{\mathbf{x}}_{L+1:L+H}$, such that

$$\hat{\mathbf{x}}_{L+1:L+H} = f(\mathbf{x}_{1:L}, \mathbf{d}_{1:L})$$

Note since we are building a general purpose multivariate forecaster, the only covariates we consider in the pretraining stage are three time features: second of the day, day of the week and month of the year. These three time features, if available, are converted to 6 features using sine and cosine transformations[2].

4. Method

4.1. Pretraining Data Preparation

We collected a large scale time series repository containing 2.4B univariate or multivariate time points from both internal and public sources. Our repository consists of about 180,000 univariate or multivariate time series spanning diverse domains including manufacturing, transportation, finance, environmental sensing, healthcare, to name some.

For each series, we take its first 90% time points to extract training samples and the remaining 10% time points to extract validation samples (we monitor validation loss for early stopping during training). Each extracted time series sample consists of 1088 consecutive time points without missing values. Our model is trained to predict the values of the last 64 time points using the preceding 1024 time points as context.

To ensure consistency, we restrict the max number of channels for a time series sample to 32, in which 6 channels are reserved for time features. In case the number of channels of a time series sample is less than 32, we complement its channel number to 32 by setting all the values in the added channels to zero. Conversely, if a time series sample has more than 32 channels, we divide it into samples with 32 or fewer channels and then supplement the samples with less than 32 channels to reach the total of 32 channels.

To achieve a unified numerical magnitude for time series samples across different datasets, we normalize each time series sample on a channel-wise basis. Specifically, the first 1024 time points are normalized to have zero mean and unit variance. Then, the last 64 time points are normalized using the calculated channel mean and standard deviation from the first 1024 time points. More precisely, let $x_{1:1024}$ and $x_{1025:1088}$ be the first 1024 and last 64 time points for a channel of time series sample, the normalization is conducted as follows:

$$\begin{aligned} x_{1025:1088} &= \frac{x_{1025:1088} - \text{mean}(x_{1:1024})}{\text{stdev}(x_{1:1024}) + \epsilon} \\ x_{1:1024} &= \frac{x_{1:1024} - \text{mean}(x_{1:1024})}{\text{stdev}(x_{1:1024}) + \epsilon} \end{aligned}$$

Normalized samples that have a data point with an absolute value greater than 9 are discarded to exclude samples containing extreme values. Furthermore, we mask 1 to 960 time points in the beginning of 10% randomly chosen samples to zero values. This manipulation allows us to generate samples with shorter context lengths, providing a variation in the length of context within the training data.

Lastly, to ensure a balance between the scale and domain diversity of our training data, we restrict the max number of training or validation samples that can be extracted from a single time series to 60,000. In the end, approximately 200M high quality training samples and 24M validation samples are generated from our repository.

4.2. General Time Transformer

An overview of GTT is depicted in Figure 1. Specifically, we split an input multivariate time series sample into fixed-size non-overlapping patches channel-wise. Each patch

[2] https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/

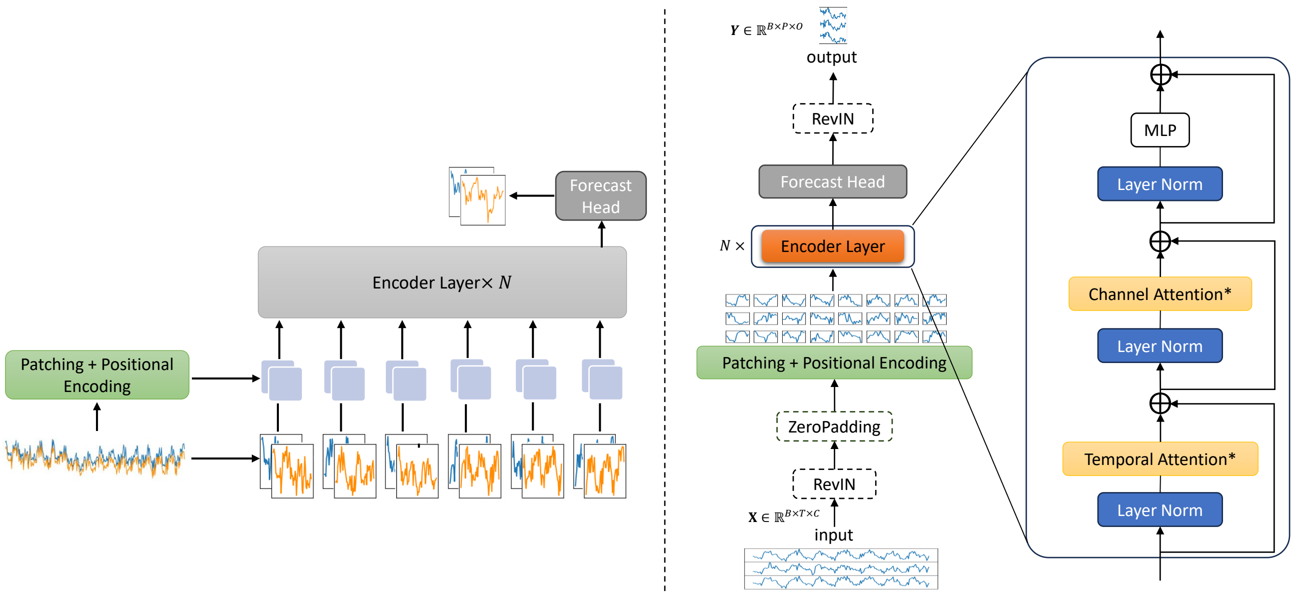


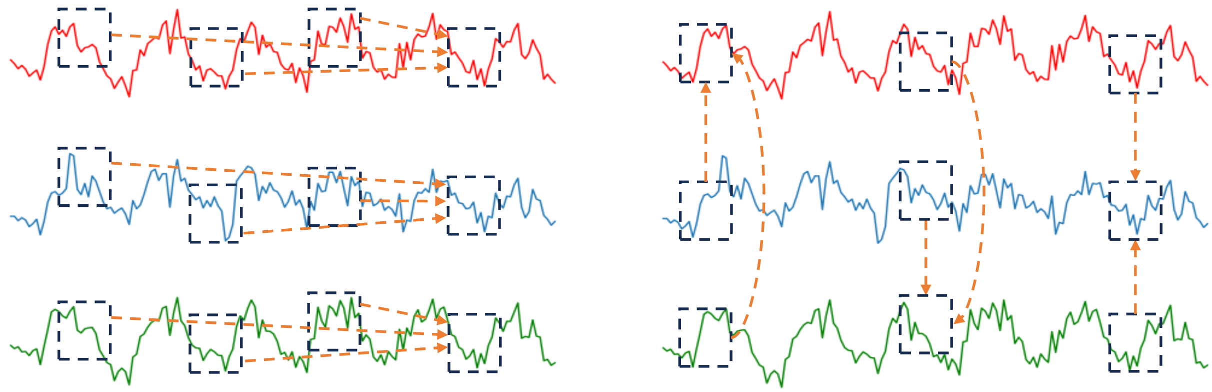
Figure 1: Model overview. We split an input multivariate time series into fixed-size non-overlapping patches (curve shapes) channel-wise, linearly embed each of them, add position encodings, and feed the resulting sequence of patches to the encoder. The encoder has an extra channel attention stage compared with the standard Transformer, the temporal and channel attention share the same weights. We add a linear head to the last token to perform forecasting. During inference, we add RevIN layers to normalize and denormalize time series channels and pad zeros in front of time series samples with less than 1024 time points.

represents a curve shape composed of 64 time points of a single variable. We linearly embed each of the patches, add position encodings, and then feed the resulting sequence of patches to the encoder. The encoder has an extra channel attention stage compared with the standard Transformer. For parameter efficiency, the temporal and channel attention share the same weights. Weight sharing between multi-head self attentions applied on different input dimensions has been proved effective (Yang et al., 2022). Lastly, we add a linear head to the last token to perform forecasting of the next patch (curve shape). During inference, we apply reversible instance normalization (RevIN) (Kim et al., 2021) to first normalize the input data into zero mean and unit variance and then padding zeros in the front if the length of input time series is shorter than 1024. The predicted output is denormalized into its original scale by using the pre-calculated mean and variance of the input data. In this way, no normalization is needed for input data before using GTT which significantly improves the convenience of model usage.

It is important to note that upon closer examination, the architectural similarities between GTT and Vision Transformer (ViT) (Dosovitskiy et al., 2020) become apparent if curve shapes are thought as special type of image patches. However, a significant distinction arises in their treatment of channels. While ViT combines RGB channels of an image within its patching process, GTT independently processes time series channels and incorporates an additional stage for channel attention, which facilitates the learning of cross-variate dependencies with varying channel numbers. We now describe the key components of GTT architecture. In the presentation, we use the following notations: B: batch size, T: input time series length, C: number of input channels (number of target variables, covariates, time features in total), O: number of output channels (number of target variables), M: number of patches, P: patch size, D: number of embedding dimensions, N: number of encoder layers.

Patching and Positional Encoding Let $\mathbf{X} \in \mathbb{R}^{B\times T\times C}$ be the input batch of time series samples, we first reshape $\mathbf{X}$ to $\hat{\mathbf{X}} \in \mathbb{R}^{BC\times T\times 1}$, then utilize a one-dimensional convolutional layer (Conv1D) with kernel size and strides equal to patch size P and number of filters equals to D, to segment input series into patches and then embed them into $M \times D$ dimensional patch embeddings channel-wise. We use the Positional Encoding in the original Transformer paper (Vaswani et al., 2017) for encoding position information and add position encodings to the patch embeddings to retain sequential information:

$$\hat{\mathbf{X}} = \mathrm{Reshape}(\mathbf{X}), \quad \mathbf{X} \in \mathbb{R}^{B\times T\times C}, \hat{\mathbf{X}} \in \mathbb{R}^{BC\times T\times 1}$$
$$\mathbf{Z}_0 = \mathrm{Conv1D}(\hat{\mathbf{X}}) + \mathbf{E}_{pos}, \quad \mathbf{E}_{pos}, \mathbf{Z}_0 \in \mathbb{R}^{BC\times M\times D}$$



(a) Temporal Attention (T-MSA)

(b) Channel Attention (C-MSA)

Figure 2: Illustration of Temporal and Channel Attention

Encoder Layers To utilize both temporal and cross channel dependencies for the forecasting task, we apply two multi-head self attentions (MSA), which we call temporal attention (T-MSA) and channel attention (C-MSA) in each encoder layer of GTT. An illustration of T-MSA and C-MSA is given in Figure 2. A MLP consisting of two layers with a GELU non-linearity (Hendrycks & Gimpel, 2016) is applied after T-MSA and C-MSA. Layernorm (LN) and residual connections are also applied:

$$
\begin{aligned}
&\mathbf{Z}'_l = \text{T-MSA}(\text{LN}(\mathbf{Z}_{l-1})) + \mathbf{Z}_{l-1}, \quad l = 1, \ldots, N \\
&\hat{\mathbf{Z}}'_l = \text{Reshape}(\mathbf{Z}'_l), \quad \mathbf{Z}'_l \in \mathbb{R}^{BC \times M \times D}, \hat{\mathbf{Z}}'_l \in \mathbb{R}^{BM \times C \times D} \\
&\hat{\mathbf{Z}}''_l = \text{C-MSA}(\text{LN}(\hat{\mathbf{Z}}'_l)) + \hat{\mathbf{Z}}'_l, \quad l = 1, \ldots, N \\
&\mathbf{Z}''_l = \text{Reshape}(\hat{\mathbf{Z}}''_l), \quad \hat{\mathbf{Z}}''_l \in \mathbb{R}^{BM \times C \times D}, \mathbf{Z}''_l \in \mathbb{R}^{BC \times M \times D} \\
&\mathbf{Z}_l = \text{MLP}(\text{LN}(\mathbf{Z}''_l)) + \mathbf{Z}''_l, \quad l = 1, \ldots, N
\end{aligned}
$$

Note that although the channel number in the pretraining stage is set to 32, since channel attention requires no positional information, the trained model can generalize to varying channel dimensions in the inference stage.

Forecast Head Following the encoder layers, we retrieve $\mathbf{Z}_N^M$, the last token of the last encoder layer, and then a linear forecast head is attached to $\mathbf{Z}_N^M$ for predicting the next patch of time points for all channels, i.e., the linear head is shared by all channels. Lastly, we retrieve the channels for the target variables as the output:

$$
\begin{aligned}
&\mathbf{Y}' = \mathbf{Z}_N^M W^{D \times P} + \mathbf{b}^P, \quad \mathbf{Z}_N^M \in \mathbb{R}^{BC \times D}, \mathbf{Y}' \in \mathbb{R}^{BC \times P} \\
&\mathbf{Y}'' = \text{Reshape}(\mathbf{Y}'), \quad \mathbf{Y}'' \in \mathbb{R}^{B \times P \times C} \\
&\mathbf{Y} = \text{Retrieve}(\mathbf{Y}''), \quad \mathbf{Y} \in \mathbb{R}^{B \times P \times O}
\end{aligned}
$$

Loss Function The model is trained to minimize the Mean Absolute Error (MAE) between ground-truth and predicted values. We choose the MAE loss because it is less sensitive to outliers. It is worth to mention that the MAE loss is only calculated on the originally exist data points, i.e., data points in the supplemented channels from the data preparation step are excluded from the loss computation.

RevIN and Zero-Padding During the inference phase, we employ reversible instance normalization (RevIN) (Kim et al., 2021) to normalize the input data to have zero mean and unit variance within the model. If the length of the input time series is shorter than 1024, we pad zeros at the beginning. The predicted output is then denormalized back to its original scale using the pre-calculated mean and variance of the input data. This approach eliminates the need for normalization of the input data prior to using GTT, thereby significantly improving the convenience of model usage during inference.

It is important to note that RevIN is not employed during the training stage, which distinguishes it from many recently proposed deep supervised models (Wu et al., 2022; Nie et al., 2022; Zhou et al., 2023; Chang et al., 2023). This decision is based on the fact that the use of RevIN cannot guarantee a unified magnitude of values for time series samples. For instance, two time series samples [0.0001,0.0002,...,0.1288] and [0.001,0.002,...,1.288], with the same curve shapes, would result in the exact same loss value in our framework. However, if RevIN were used, they would contribute different loss values if their original values were used to calculate the loss. This discrepancy would introduce bias towards time series samples with larger values during pretraining.

4.3. Why Encoder-only Architecture

Our encoder-only architecture ensures that the predicted values are normalized strictly using the mean and standard deviation of the entire context window. However, employing a decoder-only architecture is problematic in this regard. In

Table 1: Details of GTT model variants.

Model	Encoder layers	Embedding dimension	Number of heads	MLP size	Parameters
GTT-Tiny	4	384	6	1536	7M
GTT-Small	6	512	8	2048	19M
GTT-Large	8	768	12	3072	57M

Table 2: Statistics of benchmark datasets.

Datasets	Number of features	Time points (Train, Validation, Test)	Frequency
ETTm1, ETTm2	7	(34465, 11521, 11521)	15 mins
ETTh1, ETTh2	7	(8545, 2881, 2881)	Hourly
Electricity	321	(18317, 2633, 5261)	Hourly
Traffic	862	(12185, 1757, 3509)	Hourly
Weather	21	(36792, 5271, 10540)	10 mins
ILI	7	(617, 74, 170)	Weekly

the decoder-only architecture, where the first patch predicts the second patch, and subsequently, the first two patches predicts the third patch, the normalization process faces a conflict. Specifically, during the pretraining data preparation step, normalization is based on the mean and standard deviation of the complete context window. As a result, the normalization of predicted values for earlier patches is influenced by the mean and standard deviation of subsequent patches which are not observable during the inference phase if the decoder-only architecture were used. This conflict compromises the normalization process, leading to inconsistencies in the magnitude of values across different time series samples.

5. Experiments

5.1. Experimental Settings

Model Variants Our largest trained model has 57M parameters, which is significantly smaller than those foundation models in NLP and CV domains. Nevertheless, we already observe excellent zero-shot forecasting performance. Details on the GTT model variants are provided in Table 1. All models are trained using the 200M training samples and 24M validation samples as described in Section 4.1. We train all models using the AdamW optimizer (Loshchilov & Hutter, 2017), training is stopped when the validation loss increases in three consecutive epochs. More training details are given in the appendix.

Benchmark Datasets To evaluate the forecasting performance of GTT, we follow the benchmarks used in PatchTST (Nie et al., 2022). Specifically, 8 popular datasets, including 4 ETT datasets, Electricity, Traffic, Weather and ILI are used. The statistics of benchmark datasets are summarized in Table 2. It is also worthy to mention that all the benchmark datasets are not included in our pretraining data.

5.2. Comparison to Supervised Models

We first compare the zero-shot multivariate forecasting performance of our largest model, GTT-Large, to state-of-the-art supervised forecasting models, including GPT4TS (Zhou et al., 2023), PatchTST (Nie et al., 2022), Crossformer (Zhang & Yan, 2022), Fedformer (Zhou et al., 2022), TimesNet (Wu et al., 2022), DLinear (Zeng et al., 2023), TSMixer (Ekambaram et al., 2023) and iTransformer (Liu et al., 2023). All the above supervised baselines are trained on the train split of each benchmark dataset. Additionally, we also report the performance of GTT (we refer GTT to GTT-Large if not specified hereafter) after fine-tuning on the train split of each benchmark dataset. Note that we only tune the Forecast Head and keep other parameters of GTT fixed during fine-tuning. The results, in terms of Mean Square Error (MSE) and Mean Absolute Error (MAE) on the test split of each benchmark dataset, are given in Table 3. We report the results of the baselines directly from their original papers if available.

We find that GTT without fine-tuning achieves the best MAE on 1 dataset, second best MAE on 5 datasets, and second best MSE on 2 datasets. Notably, GTT performs remarkably well even in a zero-shot scenario, where it faces a disadvantage as other methods have the opportunity to train on the benchmark datasets. Furthermore, we find that after fine-tuning on the train split of benchmark datasets, the performance of GTT can be further significantly improved. It achieves the best MAE on 6 datasets, best MSE on 4 datasets. These results clearly demonstrate the superiority of GTT as a foundation model for multivariate time series forecasting.

Table 3: Multivariate time series forecasting. The results are obtained by averaging predictions for four different lengths: 24, 36, 48, and 60 for the ILI dataset, 96, 192, 336, and 720 for other datasets. The best value for each metric is highlighted in red, while the second-best value is highlighted in blue. ZS is short for Zero-Shot and FT is short for Fine-Tune.

Model	GTT (ZS)		GTT (FT)		GPT4TS		PatchTST		Crossformer		Fedformer		TimesNet		DLinear		TSMixer		iTransformer	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	0.398	0.392	0.370	0.383	0.352	0.383	0.353	0.382	0.405	0.424	0.448	0.452	0.400	0.406	0.357	0.379	0.351	0.378	-	-
ETTm2	0.279	0.324	0.253	0.309	0.266	0.326	0.256	0.317	-	-	0.305	0.349	0.291	0.333	0.267	0.332	0.254	0.314	-	-
ETTh1	0.418	0.415	0.420	0.411	0.427	0.426	0.413	0.434	0.457	0.454	0.440	0.460	0.458	0.450	0.423	0.437	0.408	0.430	-	-
ETTh2	0.314	0.359	0.298	0.353	0.346	0.394	0.331	0.381	-	-	0.434	0.447	0.414	0.427	0.431	0.447	0.340	0.387	-	-
Electricity	0.157	0.249	0.155	0.246	0.167	0.263	0.159	0.253	0.305	0.358	0.214	0.327	0.192	0.295	0.166	0.264	0.155	0.251	0.178	0.270
Traffic	0.404	0.260	0.390	0.257	0.414	0.294	0.391	0.264	0.506	0.285	0.610	0.376	0.620	0.336	0.434	0.295	0.386	0.263	0.428	0.282
Weather	0.227	0.247	0.218	0.249	0.237	0.270	0.226	0.264	0.409	0.447	0.309	0.360	0.259	0.287	0.246	0.300	0.224	0.262	0.258	0.279
ILI	1.536	0.732	1.668	0.724	1.925	0.903	1.480	0.807	3.387	1.236	2.847	1.144	2.139	0.931	2.169	1.041	-	-	-	-

Table 4: Zero-shot univariate time series forecasting for the "OT" feature of benchmark datasets. The results are obtained by averaging predictions for four different lengths: 24, 36, 48, and 60 for the ILI dataset, 96, 192, 336, and 720 for other datasets. The best number for each metric is colored red.

Model	GTT		ForecastPFN	
	MSE	MAE	MSE	MAE
ETTm1	0.049	0.160	0.175	0.322
ETTm2	0.118	0.254	0.568	0.598
ETTh1	0.084	0.222	0.190	0.350
ETTh2	0.212	0.359	0.604	0.618
Electricity	0.267	0.355	2.257	1.233
Traffic	0.126	0.219	4.121	1.650
Weather	0.002	0.028	0.003	0.040
ILI	0.582	0.551	2.903	9.825

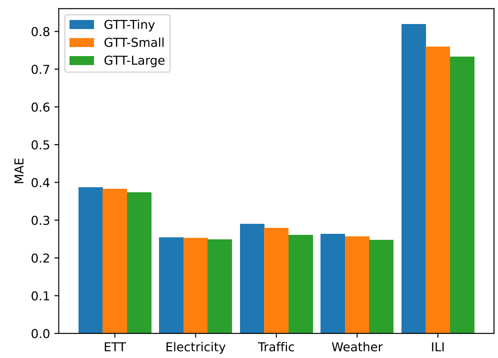


Figure 3: Zero-shot multivariate forecasting performance on benchmark datasets of GTT with different model parameter scales. The results for ETT are averaged from the four ETT datasets.

Table 5: Zero-shot univariate time series forecasting for the "OT" feature of benchmark datasets. The prediction length is 96 for ETT datasets, and 24 for ILI. The best number for each metric is colored red. The results for PreDcT is cited from (Das et al., 2023).

Model	GTT		PreDcT	
	NRMSE	WAPE	NRMSE	WAPE
ETTm1	0.129	0.096	0.591	0.310
ETTm2	0.240	0.174	0.199	0.123
ETTh1	0.203	0.153	0.672	0.378
ETTh2	0.397	0.306	0.238	0.151
ILI	0.313	0.215	0.477	0.152

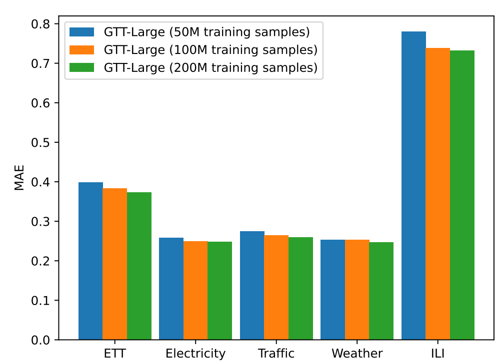


Figure 4: Zero-shot multivariate forecasting performance on benchmark datasets of GTT-Large with different training data scales. The results for ETT are averaged from the four ETT datasets.

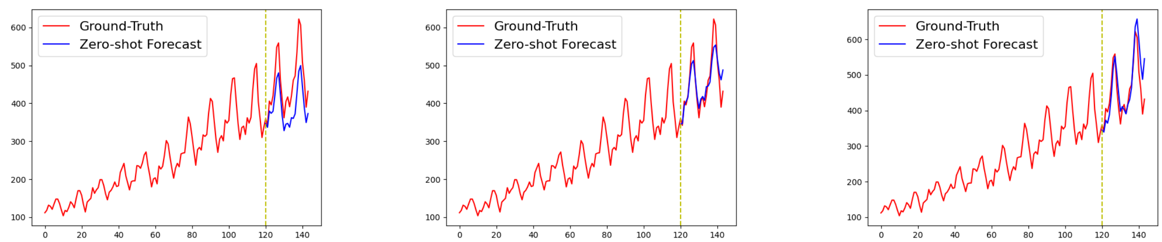


Figure 5: Zero-shot forecast of last 24 months' values in Air Passenger dataset produced by GTT-Tiny (left), GTT-Small (mid), and GTT-Large (right). We observe that with a larger model, the accelerated increasing trend can be better captured.

5.3. Comparison to Pretrained Models

We then compare GTT against recently proposed pretrained models for zero-shot univariate time series forecasting, specifically ForecastPFN (Dooley et al., 2023) and PreDcT (Das et al., 2023). It should be noted that we excluded TimeGPT (Garza & Mergenthaler-Canseco, 2023) and Lag-Llama (Rasul et al., 2023) from the comparison as their models have not been released yet and no comparable results on the benchmark datasets are presented in their papers. Table 4 presents the zero-shot performance of GTT and ForecastPFN in terms of MAE and MSE on all benchmark datasets for univariate time series forecasting. GTT demonstrates a substantial margin of improvement over ForecastPFN, outperforming it on all datasets. Table 5 compares GTT's zero-shot performance for univariate time series forecasting with PreDcT in terms of Normalized Root MSE (NRMSE) and Weighted Average Percentage Error (WAPE) on five benchmark datasets. It is important to note that since a pretrained model for PreDcT has not been released, we directly cite the results for PreDcT from the original paper (Das et al., 2023). GTT outperforms PreDcT in half of the scenarios. It is worth mentioning that the comparison is limited to univariate forecasting as GTT is the only pretrained model capable of supporting both univariate and multivariate forecasting.

5.4. Scaling Study

We first study how the parameter scale impacts the zero-shot multivariate forecast performance of GTT models. Figure 3 gives the forecasting performance in terms of MAE on the benchmark datasets for GTT-Tiny, GTT-Small and GTT-Large. It can be observed that when the training data size does not bottleneck, the zero-shot forecasting accuracy increases with a larger model. Figure 5 gives an interesting example of how model parameter scale impact zero-shot forecasting performance using the Air Passenger dataset [3]: we find that with a larger model, the accelerated increasing trend in the Air Passenger dataset can be better captured.

We then study how crucial is the training dataset size. Specifically, we also pre-train GTT-Large models on smaller datasets of size: 50M and 100M training samples. Figure 4 gives the zero-shot forecasting performance in terms of MAE on the benchmark datasets for GTT-Large pretrained on 50M, 100M and 200M samples respectively. It can be seen that when the model size does not bottleneck, the zero-shot forecasting accuracy also increases with a larger training dataset.

Importantly, the above results indicates that GTT does not appear to reach saturation within the range of model parameter and dataset sizes explored. This motivates future scaling efforts for our proposed method.

6. Conclusions

We have explored training a Transformer-based foundation model called GTT for multivariate time series forecasting. We do not introduce any time-series-specific inductive biases into the GTT architecture. Instead, we interpret an arbitrary multivariate time series as a sequence of curve shapes (patches) and process it on a channel-wise basis by a Transformer encoder with an extra channel attention stage. This straightforward yet scalable approach yields impressive results, particularly when combined with pretraining on large datasets. GTT demonstrates comparable or superior performance to many advanced supervised models in multivariate time series forecasting across various widely used benchmark datasets.

While these initial findings are promising, several challenges still need to be addressed. Firstly, it is crucial to incorporate uncertainty calibration, encompassing both aleatoric and epistemic uncertainties, into the forecasting outcomes to enhance the reliability of GTT predictions. Another challenge is to extend the context length of GTT models to further improve their forecasting capabilities. Lastly, further scaling of GTT is expected to yield enhanced performance.

[3]https://www.kaggle.com/datasets/rakannimer/air-passengers/data

7. Potential Broader Impact

This paper presents work whose goal is to advance the field of time series forecasting. One of the societal consequences of our work could be the promotion of large-scale foundation models specifically designed for time series analysis tasks. Like in CV and NLP domains, the reliability of such foundation models must be carefully checked before massive usage in public.

References

Borovykh, A., Bohte, S., and Oosterlee, C. W. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

Borovykh, A., Bohte, S., and Oosterlee, C. W. Dilated convolutional neural networks for time series forecasting. *Journal of Computational Finance, Forthcoming*, 2018.

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

Chang, C., Peng, W.-C., and Chen, T.-F. Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms. *arXiv preprint arXiv:2308.08469*, 2023.

Das, A., Kong, W., Sen, R., and Zhou, Y. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

Dooley, S., Khurana, G. S., Mohapatra, C., Naidu, S. V., and White, C. Forecastpfn: Synthetically-trained zero-shot forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Durbin, J. and Koopman, S. J. *Time series analysis by state space methods*, volume 38. OUP Oxford, 2012.

Ekambaram, V., Jati, A., Nguyen, N., Sinthong, P., and Kalagnanam, J. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. *arXiv preprint arXiv:2306.09364*, 2023.

Garza, A. and Mergenthaler-Canseco, M. Timegpt-1. *arXiv preprint arXiv:2310.03589*, 2023.

Godahewa, R., Bergmeir, C., Webb, G. I., Hyndman, R. J., and Montero-Manso, P. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*, 2021.

Gruver, N., Finzi, M. A., Qiu, S., and Wilson, A. G. Large language models are zero-shot time series forecasters. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

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

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kopp, M., Kreil, D., Neun, M., Jonietz, D., Martin, H., Herruzo, P., Gruca, A., Soleymani, A., Wu, F., Liu, Y., et al. Traffic4cast at neurips 2020-yet more on the unreasonable effectiveness of gridded geo-spatial processes. In *NeurIPS 2020 Competition and Demonstration Track*, pp. 325–343. PMLR, 2021.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*, 2021.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., Ma, L., and Long, M. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

Rasul, K., Ashok, A., Williams, A. R., Khorasani, A., Adamopoulos, G., Bhagwatkar, R., Biloš, M., Ghonia, H., Hassen, N. V., Schneider, A., et al. Lag-llama: Towards foundation models for time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

Yang, C.-H. H., Tsai, Y.-Y., and Chen, P.-Y. Voice2series: Reprogramming acoustic models for time series classification. In *International conference on machine learning*, pp. 11808–11819. PMLR, 2021.

Yang, T., Zhu, Y., Xie, Y., Zhang, A., Chen, C., and Li, M. Aim: Adapting image models for efficient video action recognition. In *The Eleventh International Conference on Learning Representations*, 2022.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Zhang, Y. and Yan, J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, pp. 27268–27286. PMLR, 2022.

Zhou, T., Niu, P., Wang, X., Sun, L., and Jin, R. One fits all: Power general time series analysis by pretrained lm. *arXiv preprint arXiv:2302.11939*, 2023.

Zivot, E. and Wang, J. Vector autoregressive models for multivariate time series. *Modeling financial time series with S-PLUS®*, pp. 385–429, 2006.

Models	Optimizer	Initial LR	LR Decay	Weight Decay	Gradient Clip Norm	Warmup Steps	Batch Size
GTT-Large	AdamW	3×10^{-4}	cosine	0.004	1.0	2048	1024
GTT-Small	AdamW	6×10^{-4}	cosine	0.004	1.0	2048	2048
GTT-Tiny	AdamW	1×10^{-3}	cosine	0.004	1.0	2048	4096

Table 6: Pretraining Setups for GTT models.

A. Experiment Details

Table 6 summarizes our pretraining setups for GTT variants. All GTT variants are trained on a cluster of Nvidia A800 GPUs. Pretraining is stopped when the validation loss increases in three consecutive epochs.

For fine-tuning, we use the Adam optimizer (Kingma & Ba, 2014) with 10^{-3} learning rate (LR) and no LR decay. Fine-tuning is also stopped when the validation loss increases in three consecutive epochs. The model with the best validation loss is used for further experiments.

B. Detailed Experiment Results

We give the detailed results for multivariate time series forecasting on the benchmark datasets in Table 7. The detailed results for univariate time series forecasting on benchmark datasets is presented in Table 8.

C. Visualization of Zero-shot Forecasting Results

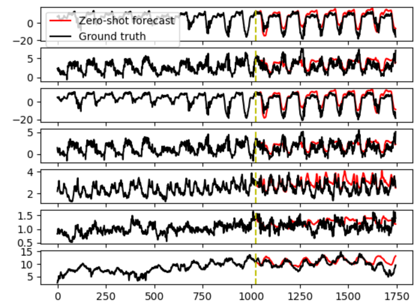
To provide a clear view of GTT's zero-shot forecast performance, we give an example of GTT's zero-shot forecast results on each benchmark datasets in Figure 6. For ETT datasets, we show results on all variables. For other datasets, we only show results on the "OT" variable.

Table 7: Detailed results for multivariate time series forecasting. We fix the input context length of GTT to 128 for ILI, and 1024 for other datasets.

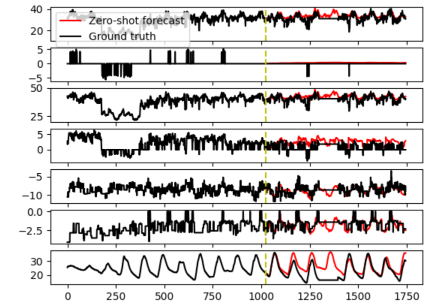
Model		GTT-Tiny		GTT-Small		GTT-Large		GTT-Large (100M traing samples)		GTT-Large (50M traing samples)		GTT-Large (Fine-tune)	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.348	0.362	0.329	0.353	0.324	0.352	0.342	0.366	0.387	0.394	0.322	0.355
	192	0.395	0.389	0.377	0.384	0.374	0.381	0.393	0.398	0.441	0.428	0.350	0.373
	336	0.442	0.413	0.424	0.412	0.418	0.403	0.437	0.424	0.456	0.486	0.382	0.391
	720	0.506	0.443	0.503	0.456	0.478	0.432	0.500	0.459	0.554	0.496	0.429	0.417
	mean	0.423	0.402	0.408	0.401	0.398	0.392	0.418	0.412	0.467	0.444	0.370	0.383
ETTm2	96	0.203	0.270	0.200	0.269	0.178	0.256	0.200	0.268	0.196	0.271	0.176	0.256
	192	0.273	0.316	0.270	0.315	0.247	0.304	0.261	0.309	0.267	0.319	0.225	0.290
	336	0.331	0.353	0.328	0.352	0.307	0.344	0.316	0.344	0.332	0.362	0.272	0.323
	720	0.400	0.398	0.412	0.406	0.383	0.395	0.395	0.394	0.421	0.419	0.340	0.370
	mean	0.302	0.334	0.303	0.336	0.279	0.324	0.293	0.329	0.304	0.343	0.253	0.309
ETTh1	96	0.396	0.393	0.391	0.390	0.375	0.384	0.380	0.380	0.391	0.391	0.366	0.377
	192	0.444	0.420	0.437	0.413	0.414	0.407	0.436	0.409	0.444	0.420	0.410	0.401
	336	0.466	0.436	0.459	0.427	0.424	0.419	0.468	0.432	0.475	0.444	0.433	0.418
	720	0.522	0.476	0.506	0.461	0.460	0.450	0.524	0.475	0.548	0.496	0.474	0.451
	mean	0.457	0.431	0.448	0.423	0.418	0.415	0.452	0.424	0.465	0.438	0.420	0.411
ETTh2	96	0.274	0.328	0.268	0.317	0.251	0.310	0.262	0.312	0.275	0.319	0.236	0.308
	192	0.321	0.362	0.316	0.352	0.295	0.342	0.315	0.349	0.320	0.354	0.275	0.335
	336	0.356	0.390	0.343	0.378	0.318	0.366	0.350	0.379	0.383	0.355	0.309	0.362
	720	0.433	0.445	0.426	0.435	0.393	0.421	0.419	0.433	0.419	0.432	0.374	0.411
	mean	0.346	0.381	0.338	0.371	0.314	0.359	0.336	0.368	0.342	0.372	0.298	0.353
Electricity	96	0.119	0.217	0.123	0.217	0.117	0.212	0.119	0.214	0.124	0.219	0.116	0.212
	192	0.140	0.236	0.147	0.240	0.138	0.232	0.140	0.233	0.148	0.239	0.136	0.230
	336	0.164	0.258	0.170	0.260	0.162	0.253	0.162	0.253	0.174	0.263	0.159	0.251
	720	0.219	0.303	0.211	0.292	0.215	0.298	0.213	0.295	0.238	0.312	0.212	0.293
	mean	0.161	0.254	0.163	0.252	0.157	0.249	0.159	0.249	0.171	0.258	0.155	0.246
Traffic	96	0.388	0.257	0.366	0.247	0.358	0.235	0.366	0.241	0.369	0.246	0.346	0.232
	192	0.413	0.271	0.389	0.260	0.381	0.246	0.385	0.251	0.391	0.260	0.368	0.244
	336	0.450	0.292	0.420	0.281	0.407	0.262	0.409	0.266	0.416	0.277	0.393	0.259
	720	0.534	0.338	0.502	0.329	0.472	0.298	0.470	0.299	0.482	0.318	0.455	0.295
	mean	0.446	0.289	0.419	0.279	0.404	0.260	0.407	0.264	0.415	0.275	0.390	0.257
Weather	96	0.147	0.183	0.149	0.182	0.144	0.179	0.150	0.183	0.150	0.183	0.143	0.182
	192	0.199	0.235	0.198	0.231	0.190	0.224	0.199	0.231	0.200	0.231	0.187	0.228
	336	0.265	0.284	0.257	0.277	0.247	0.267	0.259	0.274	0.258	0.272	0.236	0.269
	720	0.376	0.352	0.344	0.334	0.326	0.319	0.337	0.323	0.350	0.330	0.308	0.319
	mean	0.246	0.264	0.237	0.256	0.227	0.247	0.236	0.253	0.240	0.254	0.218	0.249
ILI	24	1.596	0.733	1.583	0.718	1.377	0.676	1.585	0.686	1.541	0.701	1.580	0.677
	36	1.908	0.810	1.607	0.740	1.469	0.714	1.643	0.720	1.752	0.763	1.617	0.707
	48	1.950	0.835	1.630	0.760	1.543	0.740	1.715	0.753	1.854	0.798	1.671	0.735
	60	2.154	0.900	1.841	0.818	1.758	0.800	1.850	0.799	2.082	0.862	1.804	0.780
	mean	1.902	0.819	1.666	0.759	1.536	0.732	1.698	0.739	1.807	0.781	1.668	0.724

Table 8: Detailed results for univariate time series forecasting. We fix the input context length of GTT to 128 for ILI, and 1024 for other datasets.

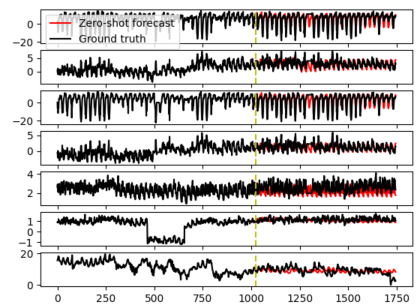
Model		GTT-Large				ForecastPFN	
		MSE	MAE	NRMSE	WAPE	MSE	MAE
ETTm1	96	0.029	0.127	0.129	0.096	0.171	0.317
	192	0.041	0.149	0.153	0.112	0.174	0.321
	336	0.053	0.167	0.173	0.126	0.177	0.325
	720	0.071	0.196	0.198	0.146	0.176	0.324
	mean	0.049	0.160	0.163	0.120	0.175	0.322
ETTm2	96	0.069	0.190	0.240	0.174	0.565	0.599
	192	0.098	0.232	0.285	0.211	0.568	0.599
	336	0.128	0.269	0.325	0.244	0.570	0.598
	720	0.177	0.326	0.378	0.293	0.568	0.596
	mean	0.118	0.254	0.307	0.231	0.568	0.598
ETTh1	96	0.063	0.189	0.203	0.153	0.198	0.358
	192	0.075	0.211	0.219	0.169	0.184	0.345
	336	0.087	0.229	0.230	0.179	0.186	0.347
	720	0.114	0.262	0.249	0.194	0.191	0.350
	mean	0.084	0.222	0.225	0.174	0.190	0.350
ETTh2	96	0.137	0.285	0.397	0.306	0.608	0.619
	192	0.175	0.327	0.440	0.345	0.566	0.598
	336	0.219	0.374	0.471	0.376	0.588	0.612
	720	0.318	0.450	0.512	0.408	0.655	0.642
	mean	0.212	0.359	0.455	0.359	0.604	0.618
Electricity	96	0.217	0.314	0.562	0.378	2.206	1.221
	192	0.249	0.336	0.602	0.405	2.165	1.206
	336	0.281	0.363	0.641	0.438	2.251	1.231
	720	0.322	0.408	0.694	0.500	2.405	1.272
	mean	0.267	0.355	0.625	0.430	2.257	1.233
Traffic	96	0.103	0.186	0.269	0.156	4.173	1.654
	192	0.110	0.198	0.278	0.166	4.090	1.646
	336	0.124	0.219	0.294	0.183	4.111	1.649
	720	0.171	0.275	0.342	0.228	4.111	1.651
	mean	0.126	0.219	0.296	0.183	4.121	1.650
Weather	96	0.001	0.022	0.445	0.317	0.002	0.038
	192	0.001	0.027	0.520	0.383	0.002	0.038
	336	0.002	0.030	0.574	0.431	0.003	0.040
	720	0.002	0.036	0.683	0.513	0.003	0.043
	mean	0.002	0.028	0.556	0.411	0.003	0.040
ILI	24	0.471	0.470	0.313	0.215	1.102	0.887
	36	0.522	0.520	0.321	0.231	1.071	0.893
	48	0.605	0.572	0.334	0.245	1.210	0.949
	60	0.732	0.640	0.354	0.265	1.447	1.032
	mean	0.582	0.551	0.331	0.239	1.208	0.940



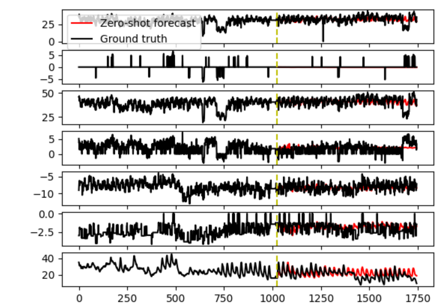
(a) ETTm1



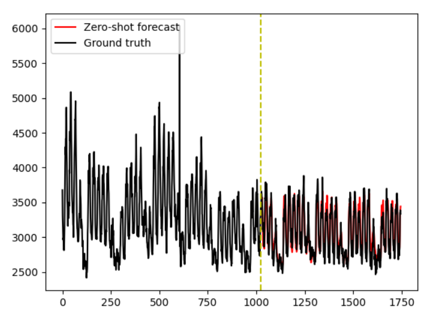
(b) ETTm2



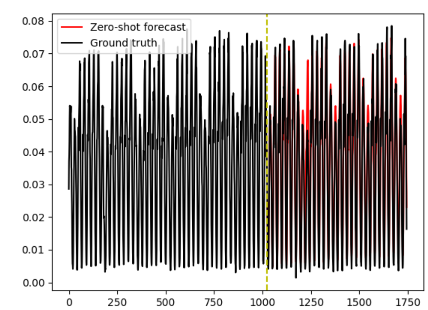
(c) ETTh1



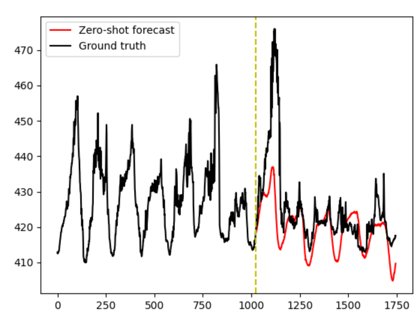
(d) ETTh2



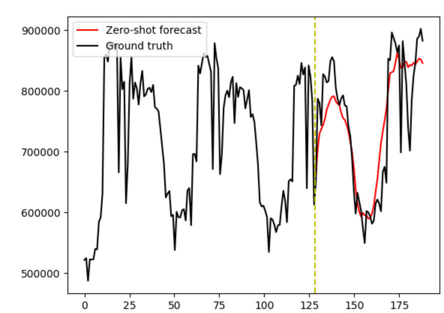
(e) Electricity



(f) Traffic



(g) Weather



(h) ILI

Figure 6: Examples of GTT's zero-shot forecast results on benchmark datasets