https://github.com/ysssishan/TiDE_Mini_Research

Mini Research on *Long-term Forecasting with TiDE: Time-series Dense Encoder*

Group Member:

Siyuan Wang, Hang Yang, Sishan Yang, Jiarong Zhou

1. Introduction

Long-term forecasting is one of the most fundamental research directions in time series analysis, which aims to predict future steps given a long context or ability to look back. In previous solution, several classical statistical approaches were implemented such as ARIMA or GARCH (Box et al., 2015), while recently deep learning models have been an emerging trend for forecasting tremendous multivariate time series data, whose performances exceed traditional methods (Wu et al., 2021; Nie et al., 2022).

Neural network architectures have been widely deployed for forecasting, including Recent Neural Network, Convolutional network, and Graph Neural Network. While Transformer, dominates in sequence modeling, which outperform LSTMs models regarding to language, speech and vision processing. Various transformer-based forecasting models in time-series community consist of state-of-the-art (SoTA) performance for long time frame. While a recent paper, 'are transformer effective for time series forecasting?' (Zeng et al., 2023), provided evidence that multivariate transformer-based architectures may not be powerful compared with simple linear univariate models on forecasting benchmark. However, such a linear model has disadvantages of non-linear dependencies in time-series sequence and time-independent covariates. Recently, a new Transformer-based architecture (Nie et al., 2022) captures SoTA performance for deep neural networks for multivariate forecasting standards.

1.1. Related Work

Vanilla Transformer has four significant limitations when solving the LSTF problem:

- **Less adaptive absolute position encoding.** Position encoding is added in the input embeddings to model the sequence information. Although this approach can extract some positional information from time series, they were unable to fully exploit the important features of time series data.
- **The quadratic computation of self-attention.** The atom operation of self-attention mechanism, namely canonical dot-product, causes the time complexity and memory usage per layer to be O(L2).
- **Layering multiple encoders or decoders causes memory bottlenecks**. This will restrict the handling of long sequences.
- **The speed plunge in predicting long outputs**. Dynamic decoding of vanilla Transformer makes the step-by-step inference as slow as RNN-based model.

To tackle the above issues, there has been lots of work done from embedding temporal features, to optimizing the self-attention efficiency, replacing attention mechanism, and to use linear model. In Figure 1,

we generated a simple pipeline of existing models to solve long-term time series forecasting (LTSF) problem based on the work of Zeng et al., 2023.

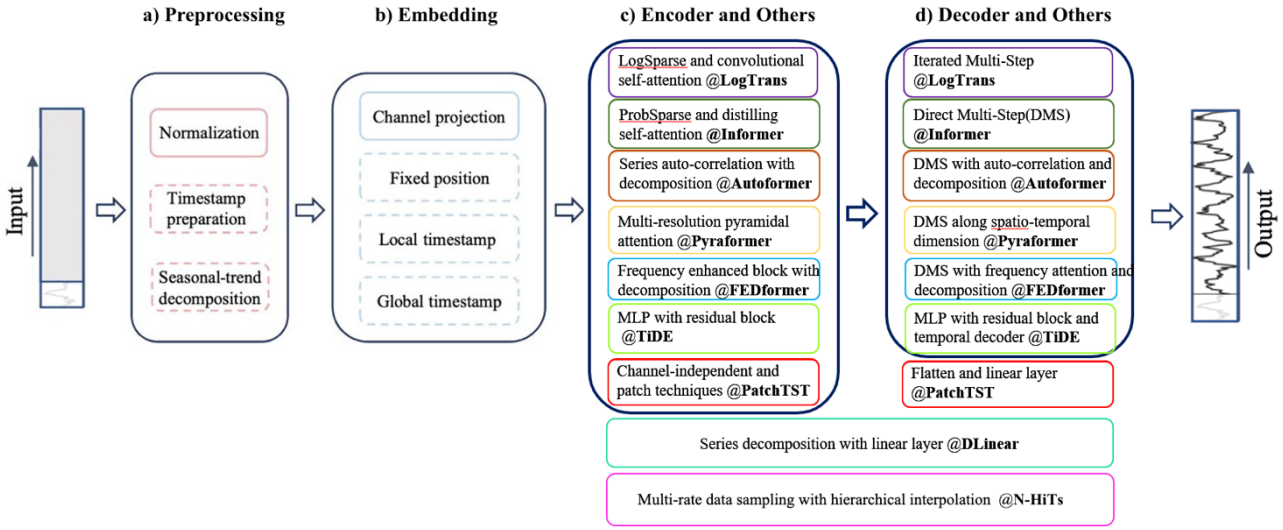


Figure 1: the pipeline of existing models to solve LSTF problem. According to the work by Zeng et al., 2023, in (a) and (b), the solid boxes are essential operations, and the dotted boxes are applied optionally.

Here, we illustrate the basic idea of related models, which are also the comparison in TiDE paper. More details can be found in Appendix A.

- **LongTrans** (Li et al., 2019a): it uses attention layer with LogSparse design to capture local information with near linear space and computational complexity.
- **Informer** (Zhou et al., 2021): it proposes improvements to the Transformer model by utilizing a sparse self-attention mechanism and generative-style decoder, inspiring a series of subsequent Transformer-based LTSF models.
- **Autoformer** (Wu et al., 2021): it combines trend decomposition techniques with an autocorrelation mechanism, inspiring subsequent work such as FEDformer.
- **Pyraformer** (Liu et al., 2021): it uses pyramidal self-attention that has linear complexity and can attend to different granularities.
- **FEDformer** (Zhou et al., 2022): it employs trend decomposition and Fourier transformation techniques to improve the performance of Transformer-based models in LTSF. It was the best-performing Transformer-based model before Dlinear.
- **N-HiTs** (Challu et al., 2023): It is an improvement over N-Beats. It combines two complementary techniques, multi-rate input sampling and hierarchical interpolation, to produce drastically improved, interpretable and computationally efficient long-horizon time-series predictions.
- **Dlinear** (Zeng et al., 2023): a highly insightful work that employs simple linear models and trend decomposition techniques, outperforming all Transformer-based models at the time. This work inspired PETformer to reflect on the utility of Transformer in LTSF.
- **PatchTST** (Nie et al., 2023): the current state-of-the-art LTSF model as of July 2023. It utilizes channel-independent and patch techniques and achieves the highest performance by utilizing the native Transformer.

2. Paper Summary

2.1. Main objective and motivation

This paper insists on simple but effective deep learning architecture for long-term forecasting with SoTA performance compared to other proposed neural network-based models. Our model is founded on Multi-Layer Perceptron (MLP) without any advanced mechanism such as self-attention, recurrent or convolutional functions. The model is named as Time-Series Dense Encoder, abbreviated as TiDE, which encodes past time-series with covariates using dense and then decodes this encoded part with future covariates. A critical part of this model is residual block as the fundamental layer in this structure which is an MLP with a hidden layer with ReLU activation and a fully linear skip connection. Merely relying on linear computational scaling, this model streamlines computation in regard to context and horizontal frames without supplementary mechanism as many Transformer-based solutions.

2.2. Key methodologies and techniques

2.2.1. Multivariate forecasting

Multivariate forecasting refers to predicting the future values of multiple interrelated time series within a dataset simultaneously. TiDE is designed to address problems in long-term multivariate forecasting. Figure 2 below shows a simple example of multivariate time series dataset.

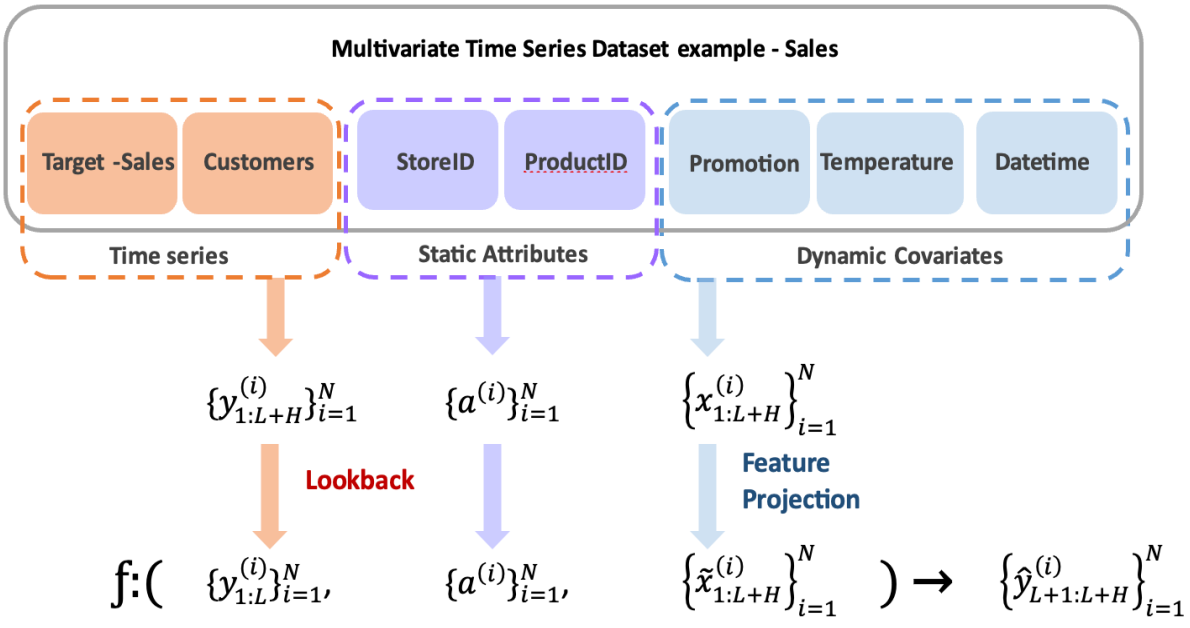


Figure 2 Multivariate time series forecasting problem

Some general notation and explanation:

- N, number of time series in a dataset
- i, index of time series
- L, context length/look back period

- H, horizon length/prediction horizon
- $y_{1:L}^{(i)}$, the lookback data of the i [th] time series
- $y_{L+1:L+H}^{(i)}$, the predict horizon
- $x_t^{(i)}$, the r-dimensional dynamic covariates of the i[th] time series at time t
- $a^{(i)}$, static attributes of the i[th] time series
- ƒ, a function maps the lookback data(1:L), the static attributes and the dynamic covariates(1:L+H) to an accurate prediction of the future

Lookback refers to the number of previous time steps or observations that the model considers. Longer lookback allows the model to capture longer-term trends and patterns, while a shorter lookback focuses on more recent information.

Static covariate, which is defined as static attributes in TiDE, refers to characteristics of a time series which remain constant or unchanging over time.

Dynamic covariates refer to characteristics that change over time and are commonly used to explain variation. It can be further classified into two categories. Global Covariates are factors that have a broad or overall impact on all time-series. They are commonly used to identify and account for broad patterns, trends, or fluctuations in a dataset. Specific covariates, on the other hand, are variables that have a more limited or localized impact, typically on one or a few subsets of time series. They are often employed to capture the impacts of factors that only pertain to a subset of the dataset's time series.

The task of TiDE is to predict the horizon time-points given access to the look-back, which is the ƒ function.

2.2.2. Model architectures and the processing flow of data

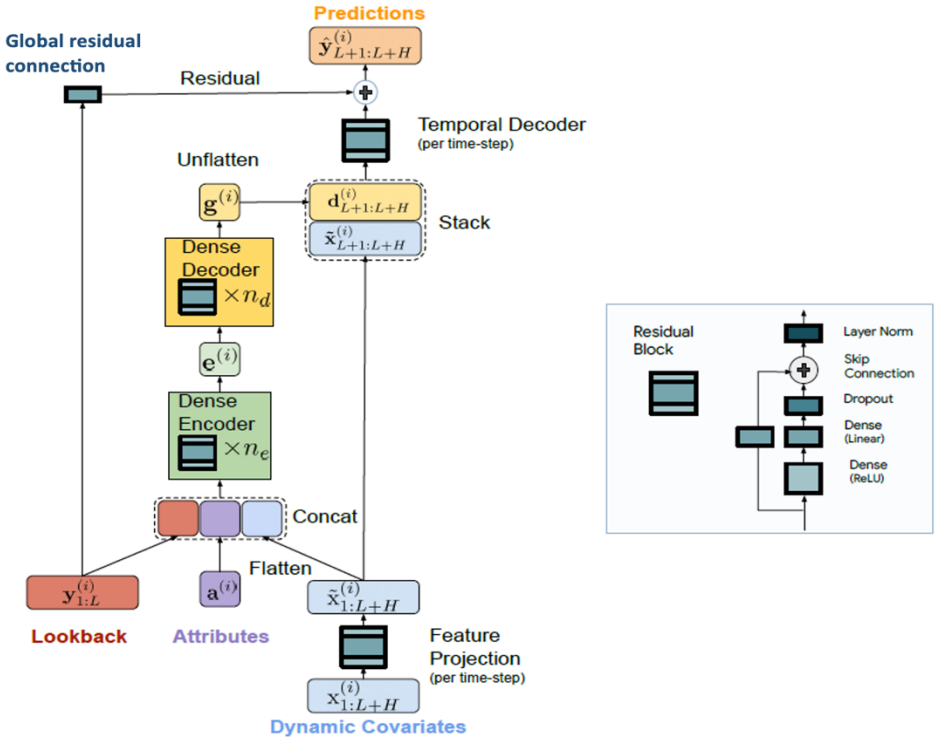


Figure 3 Overview of TiDE architecture

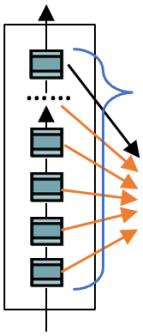
Residual block is the basic layer in TiDE architecture. In TiDE architecture, dynamic covariates are projected into a lower dimensional space using a MLP residual block. Dense encoder and dense decoder contain multiple MLP residual blocks. Finally, a temporal decoder, which is a MLP residual block, is used to generate the final prediction. The graph on the right side of Figure 3 depicts the structure of a residual block. It is an MLP with:

- one hidden layer with ReLU activation, which allows to learn complex, non-linear relationships
- linear layer that maps the hidden layer to output and the usage of dropout, which helps in addressing overfitting
- a fully linear skip connection, which allows the direct passage of information from the original input to the next block and the utilization of original features, ensuring the information flow and increasing the model's capacity, and mitigates the vanishing gradient problem as well as degradation issues
- layer norm on the output, which normalizes the output, helping to stabilize the training process and reduce the risk of vanishing or exploding gradients.

Feature Projection. The dynamic covariates are firstly processed using a residual block. The residual block takes past and future dynamic covariates at each time step as input to create a lower-dimensional representation with a reduced dimension of *temporalWidth*. This helps to decrease complexity and improve efficiency.

Input concatenation. The lookback data, static attributes and projected dynamic covariates are flattened and concatenated. The concatenated data are then used to encode the past.

Dense Encoder and Dense Decoder. Dense encoder maps the concatenated input to an embedding, which is a low-dimensional hidden representation. The decoder takes the encoding output as input and maps it to a vector, to generate future time step predictions.



Total number of layers (residual block) $\begin{cases} \boldsymbol{n_e} & dense\ encoder \\ \boldsymbol{n_d} & dense\ decoder \end{cases}$

Internal layer sizes are set to the same **hidden Size** for:

1. all layers of dense encoder
2. all layers of dense decoder except the last layer
   the last layer of dense decoder is unique with an output dimension of H × p

Figure 4: Structure of dense encoder and dense decoder

Temperal Decoder. Temporal decoder is a residual block with output size 1. It takes the stack of decoded vector and future projected covariates as input and produce a single scalar value as the prediction for each time step in the time-series. The authors stated that this operation add a "highway" from the future covariate at certain timestep to the prediction at the same time-steps. That is, TiDE can utilize information from future time steps in making predictions for a specific time step, ensuring that potential future changes are considered, facilitating more accurate and adaptable forecasts.

Global residual connection. The global residual connection is a linear mapping operation that take the lookback as input and linearly transforms them into a vector of the same size as the horizon. The output of

this linear mapping is then added to the predictions for the horizon. This addition acts as a correction or adjustment to the predictions, allowing the model to consider the global patterns and trends in the past when making predictions for the future. By adding this residual connection, TiDE can not only capture non-linear patterns and dependencies via MLP-based encoder and decoder, but also capture linear relationships.

2.2.3. Experimental Details

The authors performed a theoretical analysis under Linear Dynamical Systems (LDS). Then they evaluate three models, linear, long short-term memory (LSTM) and Transformer, on the synthetic dataset generated from the LDS. The linear model yields the best performance, measured by Mean Squared Error (MSE). Experiments performed by the authors:

- Experiment on seven popular long-term time forecasting datasets (listed in Table 1) to evaluate performance
- A demand forecasting to testify TiDE's ability to handle static attributes and complex dynamic covariates on dataset from Kaggle M5 forecasting competition
- An ablation study on the usefulness of the temporal decoder on modified Electricity dataset
- A study on the dependence of prediction accuracy with different look back periods in {720, 336, 192}
- An ablation study of the residual connections on the Electricity dataset

Dataset	#Time-Series	#Time-Points	Frequency
Electricity	321	26304	1 Hour
Traffic	862	17544	1 Hour
Weather	21	52696	10 Minutes
ETTh1	7	17420	1 Hour
ETTh2	7	17420	1 Hour
ETTm1	7	69680	15 Minutes
ETTm2	7	69680	15 Minutes

Table 1: Summary of datasets

Training and evaluation. TiDE is trained using mini-batch gradient descent where each batch consists of a *batchSize* number of time-series and the corresponding look-back [t-L:t-1] and horizon [t:t+H-1] time-points. L of 720 is used for all H in {96, 192, 336, 720}. Each epoch consists of all look-back and horizon pairs that can be constructed from the training period. MSE is used to describe training loss. The model is evaluated using rolling validation/evaluation.

Data loader. Each training batch consists of a look-back $y^B_{t-L:t-1}$ and a horizon $y^B_{t:t+H-1}$, and t can range from L + 1 to H steps before the end of the training set. B denotes the indices of the time-series in the batch. When *batchSize* is greater than N, all the time-series are loaded in a batch.

Hyperparameters chosen. Table 2 below illustrates the range of hyperparameters used in the experiment. Table 3 below shows specific hyperparameters chosen for each dataset.

Parameter	Range
`hiddenSize`	[256, 512, 1024]
`numEncoderLayers`	[1, 2, 3]
`numDecoderLayers`	[1, 2, 3]
`decoderOutputDim`	[4, 8, 16, 32]
`temporalDecoderHidden`	[32, 64, 128]
`dropoutLevel`	[0.0, 0.1, 0.2, 0.3, 0.5]
`layerNorm`	[True, False]
`learningRate`	Log-scale in [1e-5, 1e-2]
`revIn`	[True, False]

Table 2: Range of hyperparameters

Dataset	`hiddenSize`	`numEncoderLayers`	`numDecoderLayers`	`decoderOutputDim`	`temporalDecoderHidden`	`dropoutLevel`	`layerNorm`	`learningRate`	`revIn`
Traffic	256	1	1	16	64	0.3	False	6.55e-5	True
Electricity	1024	2	2	8	64	0.5	True	9.99e-4	False
ETTm1	1024	1	1	8	128	0.5	True	8.39e-5	False
ETTm2	512	2	2	16	128	0.0	True	2.52e-4	True
ETTh1	256	2	2	8	128	0.3	True	3.82e-5	True
ETTh2	512	2	2	32	16	0.2	True	2.24e-4	True
Weather	512	1	1	8	16	0.0	True	3.01e-5	False

Table 3: Hyperparameters chosen for each dataset

2.3. Results and findings

Overall, the findings showed how well the suggested TiDE model performed long-term time-series forecasting, particularly when both dynamic and static covariates needed to be taken into consideration. Additionally, the authors demonstrated that the TiDE model outperforms Transformer-based models in terms of speed and performance.

2.3.1. Results

Models		TiDE		PatchTST/64		N-HiTS		DLinear		FEDformer		Autoformer		Informer		Pyraformer		LogTrans	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.166	0.222	**0.149**	**0.198**	0.158	**0.195**	0.176	0.237	0.238	0.314	0.249	0.329	0.354	0.405	0.896	0.556	0.458	0.490
	192	0.209	0.263	**0.194**	**0.241**	0.211	0.247	0.220	0.282	0.275	0.329	0.325	0.370	0.419	0.434	0.622	0.624	0.658	0.589
	336	0.254	0.301	**0.245**	**0.282**	0.274	0.300	0.265	0.319	0.339	0.377	0.351	0.391	0.583	0.543	0.739	0.753	0.797	0.652
	720	**0.313**	0.340	**0.314**	**0.334**	0.401	0.413	0.323	0.362	0.389	0.409	0.415	0.426	0.916	0.705	1.004	0.934	0.869	0.675
Traffic	96	**0.336**	0.253	0.360	**0.249**	0.402	0.282	0.410	0.282	0.576	0.359	0.597	0.371	0.733	0.410	2.085	0.468	0.684	0.384
	192	**0.346**	**0.257**	0.379	**0.256**	0.420	0.297	0.423	0.287	0.610	0.380	0.607	0.382	0.777	0.435	0.867	0.467	0.685	0.390
	336	**0.355**	**0.260**	0.392	0.264	0.448	0.313	0.436	0.296	0.608	0.375	0.623	0.387	0.776	0.434	0.869	0.469	0.734	0.408
	720	**0.386**	**0.273**	0.432	0.286	0.539	0.353	0.466	0.315	0.621	0.375	0.639	0.395	0.827	0.466	0.881	0.473	0.717	0.396
Electricity	96	**0.132**	0.229	**0.129**	**0.222**	0.147	0.249	0.140	0.237	0.186	0.302	0.196	0.313	0.304	0.393	0.386	0.449	0.258	0.357
	192	**0.147**	0.243	**0.147**	**0.240**	0.167	0.269	0.153	0.249	0.197	0.311	0.211	0.324	0.327	0.417	0.386	0.443	0.266	0.368
	336	**0.161**	0.261	0.163	**0.259**	0.186	0.290	0.169	0.267	0.213	0.328	0.214	0.327	0.333	0.422	0.378	0.443	0.280	0.380
	720	**0.196**	0.294	**0.197**	**0.290**	0.243	0.340	0.203	0.301	0.233	0.344	0.236	0.342	0.351	0.427	0.376	0.445	0.283	0.376
ETTh1	96	**0.375**	0.398	0.379	0.401	0.378	**0.393**	0.375	**0.399**	0.376	0.415	0.435	0.446	0.941	0.769	0.664	0.612	0.878	0.740
	192	**0.412**	0.422	0.413	0.429	0.427	0.436	**0.412**	**0.420**	0.423	0.446	0.456	0.457	1.007	0.786	0.790	0.681	1.037	0.824
	336	**0.435**	**0.433**	**0.435**	0.436	0.458	0.484	0.439	0.443	0.444	0.462	0.486	0.487	1.038	0.784	0.891	0.738	1.238	0.932
	720	0.454	**0.465**	**0.446**	**0.464**	0.472	0.561	0.501	0.490	0.469	0.492	0.515	0.517	1.144	0.857	0.963	0.782	1.135	0.852
ETTh2	96	**0.270**	**0.336**	0.274	**0.337**	0.274	0.345	0.289	0.353	0.332	0.374	0.332	0.368	1.549	0.952	0.645	0.597	2.116	1.197
	192	**0.332**	0.380	0.338	**0.376**	0.353	0.401	0.383	0.418	0.407	0.446	0.426	0.434	3.792	1.542	0.788	0.683	4.315	1.635
	336	**0.360**	0.407	0.363	**0.397**	0.382	0.425	0.448	0.465	0.400	0.447	0.477	0.479	4.215	1.642	0.907	0.747	1.124	1.604
	720	0.419	0.451	**0.393**	**0.430**	0.625	0.557	0.605	0.551	0.412	0.469	0.453	0.490	3.656	1.619	0.963	0.783	3.188	1.540
ETTm1	96	0.306	0.349	**0.293**	**0.346**	0.302	0.350	0.299	0.343	0.326	0.390	0.510	0.492	0.626	0.560	0.543	0.510	0.600	0.546
	192	**0.335**	**0.366**	**0.333**	0.370	0.347	0.383	0.335	**0.365**	0.365	0.415	0.514	0.495	0.725	0.619	0.557	0.537	0.837	0.700
	336	**0.364**	**0.384**	0.369	0.392	0.369	0.402	0.369	0.386	0.392	0.425	0.510	0.492	1.005	0.741	0.754	0.655	1.124	0.832
	720	**0.413**	**0.413**	0.416	0.420	0.431	0.441	0.425	0.421	0.446	0.458	0.527	0.493	1.133	0.845	0.908	0.724	1.153	0.820
ETTm2	96	**0.161**	**0.251**	0.166	0.256	0.176	0.255	0.167	0.260	0.180	0.271	0.205	0.293	0.355	0.462	0.435	0.507	0.768	0.642
	192	**0.215**	**0.289**	0.223	0.296	0.245	0.305	0.224	0.303	0.252	0.318	0.278	0.336	0.595	0.586	0.730	0.673	0.989	0.757
	336	**0.267**	**0.326**	0.274	0.329	0.295	0.346	0.281	0.342	0.324	0.364	0.343	0.379	1.270	0.871	1.201	0.845	1.334	0.872
	720	**0.352**	**0.383**	0.362	**0.385**	0.401	0.413	0.397	0.421	0.410	0.420	0.414	0.419	3.001	1.267	3.625	1.451	3.048	1.328

Table 4: Multivariate long-term forecasting results

Results from the experiments are presented in Table 4, where Mean Squared Error (MSE) and Mean Absolute Error (MAE) for all datasets and methods are used to evaluate the models. The results show that

the proposed TiDE model outperforms all of the baselines on all of the datasets, achieving state-of-the-art performance. TiDE, PatchTST, N-HiTS, and DLinear are much better than the other baselines in all datasets. This can be attributed to the fact that sub-quadratic approximations to the full self-attention mechanism are perhaps not best suited for long-term forecasting. The same was observed in the PatchTST. Overall, the results demonstrate the effectiveness of the proposed TiDE model for long-term time-series forecasting, especially in scenarios where there are dynamic and static covariates that need to be taken into account.

Furthermore, the experimental results show that the proposed TiDE model outperforms DLinear significantly in all settings except for horizon 192 in ETTh1 where the performances are equal. This demonstrates the value of the additional non-linearity in the model.

2.3.2. Demand Forecasting

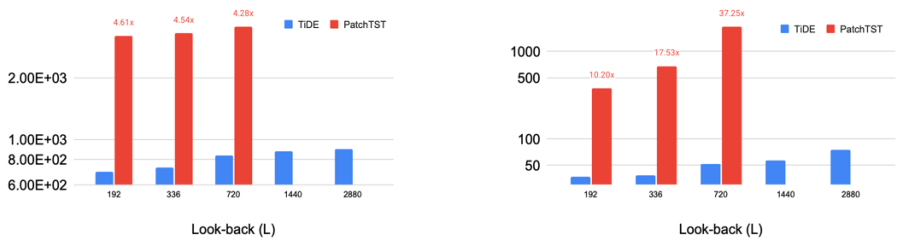
Model	Covariates	Test WRMSSE
TiDE	Static + Dynamic	$\mathbf{0.611 \pm 0.009}$
TiDE	Date only	0.637 ± 0.005
DeepAR	Static + Dynamic	0.789 ± 0.025
PatchTST	None	0.976 ± 0.014

Table 5: Forecasting results on the private test set

The M5 forecasting competition benchmarks were used in the paper to show that the model could handle both complex dynamic covariates and static attributes. The manuscript adhered to the format specified in the sample notebook published by Alexandrov et al. Over 30,000 time-series with both dynamic and static covariates, such as promotions and hierarchical categories, can be found in the M5 dataset.

It indicates that the test set results for the competition metrics that match the private leaderboard. the paper contrast with PatchTST (the best model) and DeepAR, whose implementation can handle all variables. Take note that covariates are not handled by the PatchTST implementation. the paper presents the resultant standard errors and the score over three separate runs. Given that PatchTST does not employ covariates, the paper may observe that it performs badly. the paper finds that the model with all the covariates performs up to 20% better than DeepAR. Provide the measure for the model that solely employs variables obtained from dates as covariates as well, for the purpose of ablation. Even though there is a performance drop when the dataset-specific covariates are not used, this version of the model still performs better than the other baselines.

2.3.3. Training and Inference Efficiency



(a) Inference time per batch in microseconds

(b) Training time for one epoch in seconds.

Figure 5

https://github.com/ysssishan/TiDE_Mini_Research

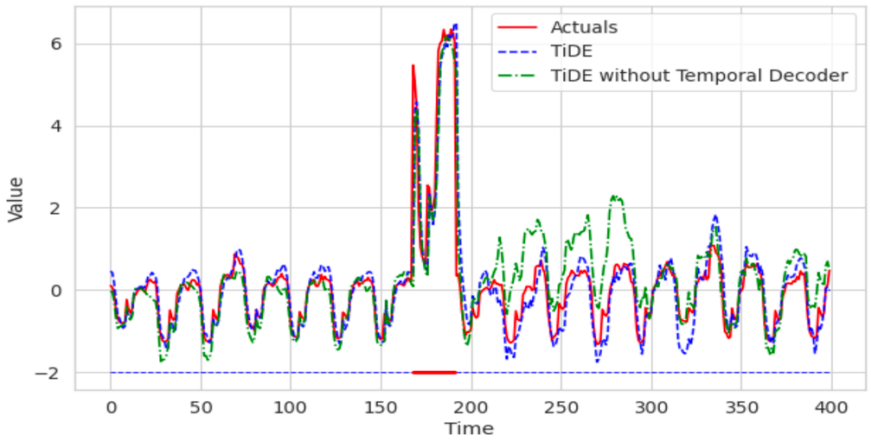


Figure 6

In this section, the paper demonstrates the usefulness of this temporal decoder component with a semi-synthetic example using the electricity dataset.

In order to accomplish this, the paper created a new dataset by excluding numerical features for two different types of events from the electricity dataset. When a Type A event (Type A events are indicated by the red portion of the horizontal line.) takes place, a uniformly determined factor between [3, 3.2] is applied to the time-series value. A uniformly determined factor between [2, 2.2] reduces the value of a time-series when an event of Type B occurs. Approximately 80% of the time-series exhibit these effects.

On this updated electricity dataset, the paper next plotted the predictions from the TiDE model after only one training session, both with and without the temporal decoder. Type A events are indicated by the red portion of the horizontal line. It is evident that there is beneficial to using the temporal decoder during that time frame. More essential, though, is that the model without the temporal decoder behaves strangely in the time instances that follow the event, maybe because it hasn't yet readjusted its past to what it should have been in the absence of the incident. Even after a single training period, this effect is minimal in the model that employs the temporal decoder.

2.4. Implications and significance of the study in the broader context of deep learning

It might be counter intuitive that simple MLP models are probably capable of matching or exceeding performance of superior neural network on widely used long-term forecasting benchmarks. Due to linear structure, this simpler deployment of TiDE model leverages on less GPU memory while PatchTST comparably runs out of GPU memory when look-back size is more than 1440.

Besides, the model runs faster since TiDE is less sensitive to look-back size resulting in shorter training time (5x faster in terms of inference and above 10x faster for straining compared to Transformer based model). Though covariates are commonly introduced to transformer-based models, previous MLPs were not extended with supporting covariates for better efficiency as TiDE attributes. This implementation of TiDE for MLP modelling direction raises a new idea of leveraging on covariates from transformer basis structure, which is valuable for future detection.

3. Analysis

3.1. Strengths

TiDE follows the logic of exploring the effectiveness of linearity on Long-term time series forecasting but renovates pure linear connection into multi-layer perceptron with residual block architecture.

Covariates and Channel–Independence. Compared with fixed position encoding in vanilla Transformer, TiDE introduces local and global timestamp embeddings (static attributes and dynamic covariates) to capture global important properties of time series. Compared with PatchTST and other MLP based models, TiDE not only utilizes past sequential values (Look-back) but also leverages additional covariate information, such as static attributes and dynamic covariates that are known at any point in time.

Same with PatchTST, TiDE uses a channel independent manner, which means each input token only contains information from a single channel -past data and covariates of one time-series at a time in this case. This was mainly used in CNN (Zheng et al., 2014) and linear models (Zeng et al., 2023). Recently, it also has been proven to work well in a Transformer-based model (PatchTST) (Nie et al., 2022). Channel-independence reduces the time and memory complexity.

Capability of linear and non-linear learning. After DLinear outperforms several Transformer-based approaches, linear model is considered to better inflect the changing on time features and is more sensitive to time order while maintaining a simple time and memory complexity. However, purely linear model ill-suited non-linearity relationship caused a huge pitfall. TiDE tackled this by introducing Multi-layer Perceptron (MLP) with activation function and the addition of residual and projected inputs. We can see from Table2 that TiDE generally has better results than DLinear, which indicates it successfully leverage the non-linear information in the datasets.

While improving the model to fit non-linearity, TiDE still holds the advantage of linear modelling by adding a global linear residual connection from the look-back to the horizon. This look-back is also implemented in each residual block, which provides a “highway” for model to maintain a linear connection to the past so that it is still sensitive to time order.

Reduce time and memory complexity to capture longer look-back. By allowing the network to focus on learning the difference between the input and output, rather than learning the entire output from scratch, residual block naturally decreases the number of parameters that need to be learned and makes training more efficient. Above research has shown TiDE is more efficient in training and inference times while it performs better or comparable to PatchTST on most datasets which allows TiDE to use a longer look-back window.

3.2. Weaknesses and potential improvements

Lack of empirical validation for covariates. The TiDE framework indeed incorporates covariates into its forecasting model, which is unique compared to other methods. However, due to the limitation of other models, there is no comparison on covariates considering performance. Also, TiDE shows the best in traffic data, which may be caused by the inherent cyclic nature of traffic dataset(morning and evening rush hours). This nature makes covariates played an important role here. So, when it comes to other types of datasets, like weather data, TiDE does not demonstrate a worse performance than PatchTST.

https://github.com/ysssishan/TiDE_Mini_Research

Customized hyperparameter settings may lose generalization. TiDE's hyperparameter design varies for different datasets, especially in the number of layers for encoders and decoders, which limits its general applicability and suggests potential overfitting. This specialization might make TiDE less effective when faced with new data types, as its tailored configurations may not capture the diverse patterns outside of the datasets it was optimized for.

Not consider channel-wise correlation and seasonal/trend decomposition. Although channel independence allows the model to independently learn the temporal dynamics of each channel, rather than looking for complex interactions between channels. This can simplify the learning process and may make the model easier to interpret because each input token corresponds to a clear physical quantity (or feature). However, this could also be a limitation, as it overlooks possible correlations between different channels. Figure 7 illustrates various channel-wise correlations among seasonal components and trend components using the ETTm1 and Traffic datasets as an example. Regarding the seasonal components, there are clear relationships between the channels in these two datasets. In comparison, whereas the trend components of Traffic and ETTm1 are not significantly correlative.

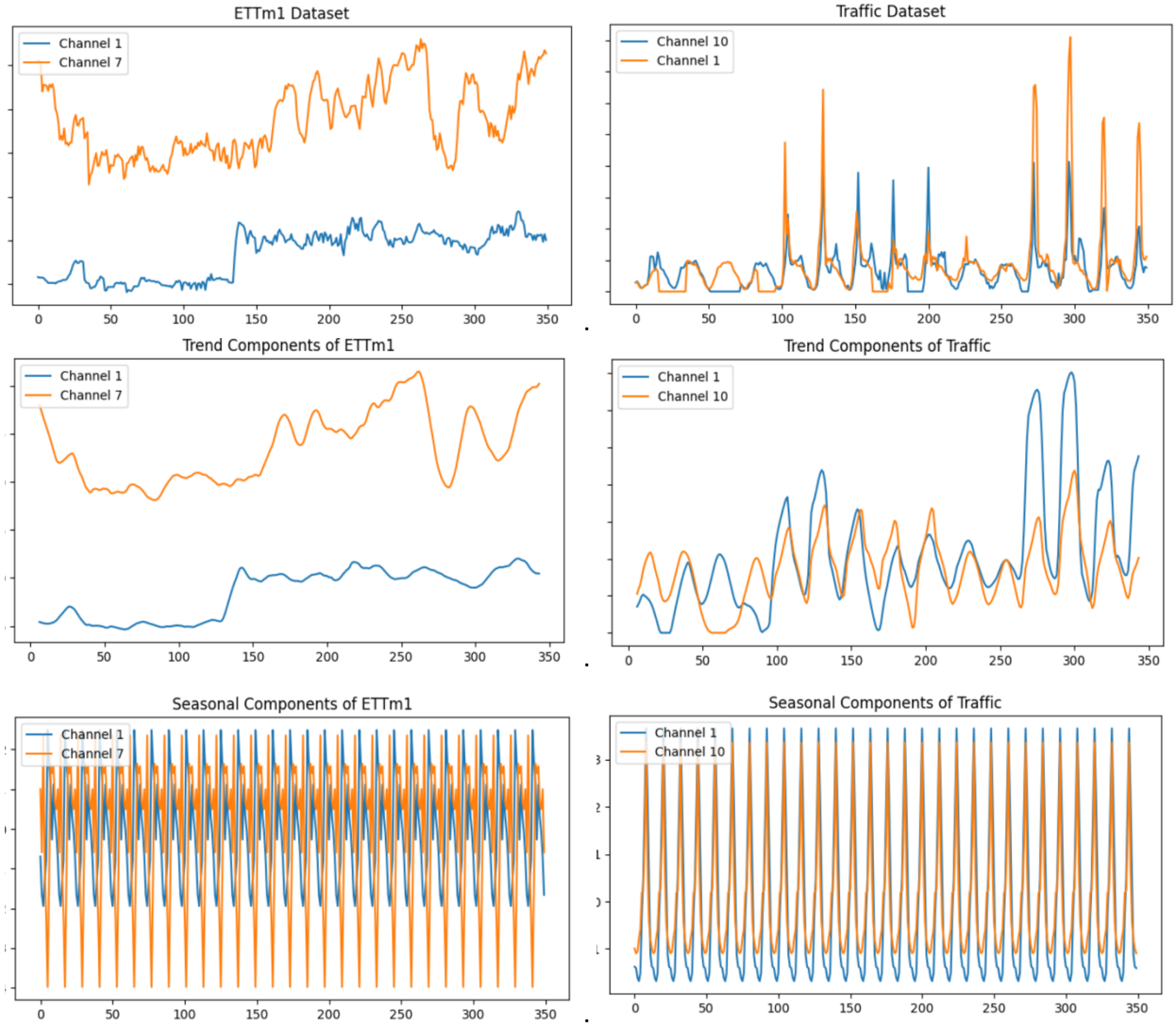


Figure 7: The top two subplots showcase the raw signals of ETTm1 and Traffic from two different channels. The lower four subplots depict the decomposed seasonal and trend components of the two datasets.

4. Implementation

4.1. Reproduction process and challenges faced

Within 22 hours (from 10am to 8am the next day), our device is only able to run 2 epochs for the first dataset - electricity. We also try to run on google Colab, though the running speed is higher (48it/s vs 9it/s on local environment), while personal trial of google account was occupied after running 2 epochs for electricity dataset. We suggest higher capacity of device when reproducing codes, such as cloud platform with higher capacity or at least i7/i10 core for higher running speed. Due to time limits, we are not able to complete all reproduction, but it is capable with recommendations above.

The process of reproduction and precise explanation are documented in our github page.

4.2. Potential improvement and algorithm

Add decomposition block into model. By adding the decomposition block, the model can better reveal the underlying structure of the data. Also, if we can predict the trend and seasonal parts separately, we can reconstruct and forecast the original series more accurately.

However, normal decomposition preprocessing has some limitations, raised by Autoformer. It has a plain effect of historical series and overlooks the hierarchical interaction between the underlying patterns of series in the long-term future. Thus, to progressively decompose the hidden series throughout the whole forecasting process, including both the past series and the predicted intermediate results, we need to harness the decomposition as an inner block of deep models.

Inside the SeriesDecomp Block, for length-L input series $X \in R_{L \times d}$, the process is:

$$X_{tr} = \text{AvgPool}(\text{Padding}(X))$$

$$X_{se} = X - X_{tr}$$

where $X_{se}, X_{tr} \in R_{L \times d}$ denote the seasonal and the extracted trend-cyclical part respectively. We adopt the AvgPool(·) for average with the padding operation to keep the series length unchanged. We use X_{se}, X_{tr} = SeriesDecomp(X) to summarize above equations, which is a model inner block.

Below is the pseudo-code:

Algorithm Series Decomposition Block

1: Input : Input historical time series Y

2: $\mathbf{Y_{tr}}$ = AvgPool(Paddding(Y))

3: $\mathbf{Y_{se}} = Y - Y_{tr}$

4: Return $\mathbf{Y_{se}}$, $\mathbf{Y_{tr}}$

Algorithm TiDE with decomposition

1: Input : Input historical time series Y; Input Dynamic Covariates X; Input Attributes A;
2: X = ResidualBlock(X)
3: $\mathbf{Y_{tr}}$, $\mathbf{Y_{se}}$= SeriesDecomp(Y)
4: For $\mathbf{Y_{tr}}$, $\mathbf{Y_{se}}$**:**
    B = Concat ($Y_{tr,se}$, A, X)
    $\mathbf{Y_{pre\text{-}tr}}$, $\mathbf{Y_{pre\text{-}se}}$ = TiDE(B)
5: $\mathbf{Y_{final}}$ = $Y_{pre\text{-}tr}$ + $Y_{pre\text{-}se}$
6: Return $\mathbf{Y_{final}}$

Other possible improvements can include:

- considering the channel-wise correlation instead simple channel independent method
- applying the model to other datasets with different effects of covariates to ensure that current performance is not overfitting because of the strong cyclical characteristic in the dataset.
- although the paper has used different horizon length, it didn't change the context length, which is 720 constantly. Thus, we can further check whether different context size can significantly affect the performance. Similar work has been done in Zeng et al., 2023, and by increasing the context length from 1 year to 1.5 years of the same dataset, transformer-based models showed worse performance because 1 year's data contains a more completed cycle while 1.5 years' data would actually introduce some noise that stems model to converge quickly.

5. Conclusion

The essay starts with the limitation of the vanilla transformers and how does it evolve to the Tide model. Then it elaborates the definition of Tide an encoder-decoder model that is based on a MLP and matches or outperforms previous neural network baselines on widely used long-term forecasting benchmarks. In addition, the model outperforms the best Transformer-based baselines by a factor of 5–10 in speed. Our analysis suggests that, at least for these long-term forecasting benchmarks, self-attention may not be required to learn the periodicity and trend patterns. The significance of the models lies in the Covariates and Channel–Independence, capabilities of both linear and non-linear learning, higher efficiency and longer looked back time. However, the limitations are also discussed. In more detail, Lack of empirical validation for covariates as well as the Customized hyperparameter settings may lose generalization. There are also a couple of potential improvements for instance.

References

Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, et al. Gluonts: Probabilistic and neural time series modeling in python. *The Journal of Machine Learning Research*, 21(1): 4629–4634, 2020.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *Proceedings of the AAAI conference on artificial intelligence*, 2023.

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 Asso- ciation for the Advancement of Artificial Intelligence Conference 2023 (AAAI 2023), 2023.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

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

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.

S Makridakis, E Spiliotis, and V Assimakopoulos. The m5 accuracy competition: Results, findings and conclusions. *Int J Forecast*, 2020.

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.

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*, pages 27268–27286. PMLR, 2022.

Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multichannels deep convolutional neural networks. In International conference on web-age information management, pp. 298–310. Springer, 2014.

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.

https://github.com/ysssishan/TiDE_Mini_Research

A Model Comparision

	Embedding	***Encoder***	***Decoder***	***Pros&Cons***
Vanilla Transformer *Transformer-based*	Fixed position encoding	1) Multi-head self-attention 2) Position-wise feed-forward fully connected network 3) Residual connection	1) Cross attention 2) Multi-head self-attention 3) Feed-forward network 4) Residual connection	**Cons**: 1) Absolute position encoding loses time features 2) Fully connected attention module has a O(N2) time complexity 3) Cross-attention mechanism between the encoder and decoder restricts the performance of LTSF 4) The speed plunge in predicting long outputs.
Log-Trans *Transformer-based*	1) Fixed position encoding 2) Local timestamp embedding	1) LogSparse 2) Convolutional self-attention	1) Iterated Multi-Step (IMS)	**Pros**: 1) LogSparse breaks the memory bottleneck 2) Causal convolution enhances the locality **Cons**: Speed-up is limited in practice
Informer *Transformer-based*	1) Fixed position encoding 2) Local timestamp embedding 3) Global timestamp embedding	1) ProbSparse 2) Distilling self-attention	1) Direct Multi-Step (DMS)	**Pros**: 1) encode timestamps via learnable embedding layers 2) Speed up the attention computation with O(LlogL) time and memory complexity 3) Acquire long sequence output with only one forward step needed, avoiding cumulative error
Autoformer *Transformer-based*	Channel projection	1) Series auto-correlation 2) Series decomposition	1) DMS with auto-correlation and decomposition	**Pros:** 1) Decomposition can progressively aggregate the long-term trend part from intermediate prediction 2) Auto-correlation can capture more time features
Pyraformer *Transformer-based*	1) Fixed position encoding 2) Local timestamp embedding 3) Global timestamp embedding	1) Pyramidal attention module (PAM) 2) Coarser-scale construction module	DMS with spatio-temporal dimension	**Pros:** Catch both short and long temporal dependencies with low time and space complexity.
FEDformer *Transformer-based*	Channel projection	Frequency enhanced block with decomposition	DMS with frequency attention and decomposition	**Pros:** 1) By randomly selecting a fixed number of Fourier components, model leads to a linear complexity

				2) Frequency enhanced structure better captures global properties of time series
N-HiTS *Multi-Layer Perceptron (MLP)-based*		1) Multi-Rate Data Sampling 2) Hierarchical Interpolation 3) Doubly residual stacking with max pooling and MLP in each block		**Pros:** 1) Save time and computation complexity
Dlinear *Multi-Layer Perceptron (MLP)-based*	Channel projection	1) Series decomposition –seasonal and trend 2) DMS with linear layer		**Pros:** 1) Save time and computation complexity 2) Better capture long-term and short-term dependences **Cons:** 1) Ill-suited non- linearity 2) Doesn't consider covariates
PatchTST *Transformer-based*	1) Learnable position encoding 2) Channel projection	1) Channel independent with series decomposition 2) Patching data	No Decoder, instead just use Flatten and a Linear layer	**Pros:** 1) Patching increases locality while capturing point-wise information 2) Patching allows a longer look-back length **Cons:** 1) Need to explore channel-wise correlation 2) Doesn't explicitly mention supporting covariates 3) The flatten method causes heavy linear-flatten-layer, resulting in an excessive increase in parameter count compared to the Transformer feature extraction module itself