Multivariate Long Sequence Time-Series Forecasting with SOTA Transformers

A critical evaluation and implementation of the 2023 paper

"*A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*"

as part of the 'Term Paper' presented to

Prof. Dr. Florian Elsässer

by

Neelesh Bhalla,

Nils Carlos Heinrich Marthiensen,

Kirtesh Pushpakbhai Patel and

Chia-Jung Chang

on November 04, 2023

for the subject Deep Learning taught to

Master of Applied Data Science [2022-24] class at

Frankfurt School of Finance & Management

Table of Content

List of Figures

List of Tables

Introduction

The paper introduces an improvement to long-term forecasting within the realm of time series analysis and highlights the surge in research endeavors, particularly driven by the proliferation of deep learning models (Bryan & Stefan, 2021; Torres et al., 2021; Lara-Benítez et al., 2021). Deep learning models, such as Transformers, have exhibited remarkable performance not only in forecasting tasks but also in the extraction of abstract representations that can be leveraged for various downstream applications like classification and anomaly detection. The Transformer architecture, renowned for its success in natural language processing, computer vision, and speech recognition, has recently extended its prowess to time series data (Wen et al., 2022) due to its intrinsic attention mechanism that facilitates automatic learning of interrelationships within sequences. This paper delves into a critical question brought to light by recent research (Zeng et al., 2022): can a simple linear model outperform the complex Transformer-based models in common time series forecasting benchmarks, thereby challenging the utility of Transformers in this domain? To address this query, the paper presents the Channel-Independence Patch Time Series Transformer (PatchTST) model, which incorporates two key innovations. First, it introduces "patching" to capture local semantic information by aggregating time steps into subseries-level patches, enhancing the understanding of correlations in time series data. Second, the model uses "channel-independence", where each input token contains information from a single channel, distinct from channel-mixing approach in the existing transformer variants. The paper offers an in-depth exploration of the PatchTST model and supports its efficacy through supervised forecasting results, ablation studies, and noteworthy achievements in self-supervised representation learning and transfer learning performance.

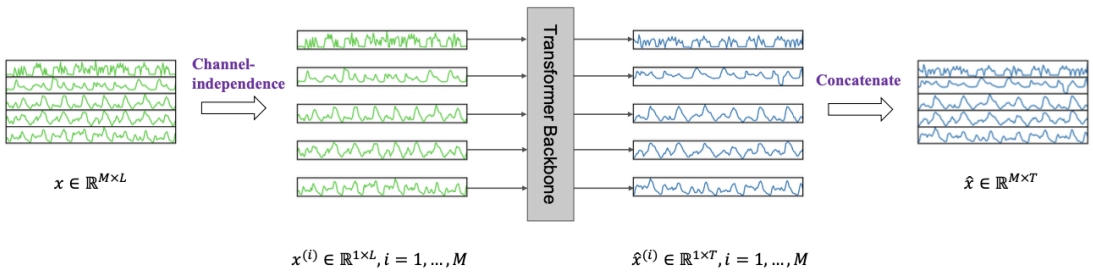

Figure 1: General depiction of the PatchTST architecture.

Paper Summary

The main objectives of this research are to enhance long-term time series forecasting accuracy and explore self-supervised representation learning for multivariate time series data. The motivation behind this work stems from the growing importance of deep learning models in the field of time series analysis, particularly with multivariate datasets. The authors utilized eight popular multivariate datasets that are publicly available on (Wu et al., 2021). Among others these include a weather dataset, which collects meteorological indicators, an hourly electricity consumption dataset and a transformer temperature dataset collected over two years.

The primary methodologies employed in this paper revolve around the development of a channel-independent patch time series Transformer (PatchTST) model. This model leverages two main design components: Patching, which involves dividing time series data into subseries-level patches, and channel-independence. The researchers opted for these methodologies after considering the unique characteristics of multivariate time series data and the challenges of long-term forecasting. For patching, the team divided the data into subseries-level patches to capture local semantic information and reduce computation and memory usage. This allowed the model to attend to longer historical data, enhancing its forecasting accuracy. Channel-independence, inspired by successful applications in CNN and linear models, is introduced to the transformer model as well. This design decision ensures that each input token contains information from a single channel, improving the model's adaptability and forecasting accuracy.

The results and findings of this study emphasize the effectiveness of PatchTST in various aspects. In terms of long-term time series forecasting, PatchTST outperforms state-of-the-art Transformer-based models, reducing mean squared error (MSE) and mean absolute error (MAE) on multiple datasets. The two versions of the model, PatchTST/64 and PatchTST/42 achieve overall MSE reductions of 21 and 20% respectively. The reductions in MAE amount to 17 and 16% respectively. The model's performance extends to self-supervised representation learning, where it achieves excellent fine-tuning results, surpassing supervised training on large datasets. The transfer of pre-trained representations from one dataset to another leads to state-of-the-art forecasting accuracy. These findings highlight the model's potential in improving time series forecasting and its adaptability to various downstream tasks.

In the broader context of deep learning, this research makes significant contributions to the field of time series forecasting, especially with multivariate datasets. By focusing on specific datasets and the unique challenges they present, the researchers have provided valuable insights into improving forecasting accuracy and model adaptability. PatchTST's innovative design components, patching, and channel-independence, demonstrate the potential for enhancing forecasting performance in applications such as weather, traffic, and electricity consumption prediction. The model's adaptability and improved self-supervised representation learning open up possibilities for various downstream tasks, including classification and anomaly detection. This research showcases the versatility of Transformer-based models in different domains and sets the stage for further advancements in deep learning for time series analysis.

Analysis

Generally, the paper exhibits a high level of clarity and readability. The authors adeptly introduce their subject matter and subsequently delve into model configurations and a series of experiments that effectively highlight the efficacy of their proposed model. Although the paper demonstrates robust numerical performance, its technical novelty is somewhat constrained as the model structure draws heavily from the conventional vision transformer (ViT). The concept of employing patches to handle time series data and dividing multivariate time series into independent channels is intriguing, although it must be noted that both concepts are not entirely novel in this context as well. Despite these limitations, the paper is commendable for its clear motivation and well-crafted prose. Subsequently, the strengths, weaknesses and identified areas of improvement will be explored in more detail.

Strengths

The research addresses two significant tasks in time series analysis, namely multivariate time series forecasting and self-supervised representation learning. Notable improvements are achieved in both areas, while keeping the model structure relatively simple overall. State-of-the-art (SOTA) results are achieved on real datasets. Several baselines are used as a comparison and extensive experiments around the performance are included in the appendix. Two key designs differentiate PatchTST from previous research in the area, namely patching and channel independence. Taken on its own these

are not new concepts, but by integrating them with a transformer-based time series forecasting model, they generate new insights for the field.

Let us first focus on patching. It is used to aggregate time steps into a subseries level. Previous work mainly focused on point-wise input tokens or handcrafted information from a given series. This can be vastly improved using patches that include locality and semantic information. Patch-generation is based on conventional vision transformers, keeping the implementation as simple as possible, sequence lengths are effectively reduced, and locality capture is successfully improved.

The second novelty is channel independence. To achieve this, a given multivariate time series is broken up into the single channels that make it up. Each input token for the transformer consists of only one single channel, which allows for the design of spatial temporal correlation across different series. While this technique has proven to be beneficial for Convolutional Neural Networks (CNN) and linear models, it has not been applied to transformers yet. The paper offers some cutting-edge research into this field.

The benefits of this approach appear to be threefold. Time and space complexity is reduced, the capability to learn from a longer look-back window and representation learning are both improved. The standard transformer model has a time and space complexity of $O(N^2)$, with N being the number of input tokens. Without any preprocessing, N has the same value as a given input sequence L. The patching procedure reduces this relationship by a factor of the stride S, resembling a quadratic complexity reduction $N \approx L/S$. The possibility for a longer look-back window follows the reduction in time and space complexity as this is the main bottleneck when extending it. Patching seems to be a more capable alternative to previously used techniques like downsampling or sparse connections of attention. When it comes to representation learning, linear models are starting to struggle. PatchTST on the other hand is able to further enhance its forecasting ability, making use of representation learning. These improvements are predominantly visible in large datasets.

Overall, the paper offers a compelling and well-organized contribution to the field of transformer-based time series forecasting and representation learning. The experiments to test the results are extensive and strong baselines have been chosen. Finally, some remarks on possible further research are discussed and a detailed appendix is available offering further details on motivation, models, and findings.

Weaknesses

The simplicity praised in the previous chapter is a double-edged sword. While it is desirable to keep the model structure as simple as possible, if only previously known techniques are used, it compromises novelty. This paper is close to the edge between desired simplicity and required novelty. Specifically, this has to be investigated with regard to the similarities to standard vision transformers, the patching process, channel independence and the concept of masked modeling in representation learning. On top of that, the experimental setup raises some questions that will be discussed here.

The proposed model closely resembles the standard vision transformer in its approach, indicating limited technical novelty. It was introduced by (Dosovitskiy et al., 2020) and has achieved SOTA performance on several image recognition benchmarks. The encoder in both papers is nearly identical, with only the data that is fed to it being changed. Figure 1 shows how similar both designs are. With the exception of moving one normalization operation from the front to the back, the transformer encoder used by PatchTST is identical to the one used in the original vision transformer from Google.

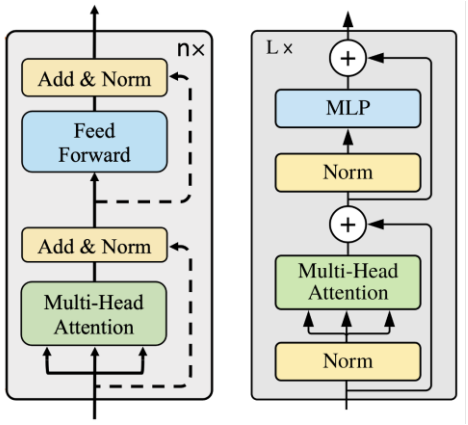

Figure 2: Comparison of transformer encoder. Left: PatchTST; Right: Vision Transformer (ViT).

The idea of patching is not entirely novel, as prior work such as LogTrans (Li et al., 2019) have suggested using convolutions to better incorporate local context into the attention mechanism. The linear processing approach in PatchTST, with a fixed patch size and stride, bears resemblance to a 1D convolution, even though the paper explores a wider range of configurations in experiments. This approach was also utilized by (Dosovitskiy et al., 2020) when introducing ViT. In that case patching is used to break down images into patches, which are then processed using attention mechanisms. This enables the model to capture global context from the entire image while maintaining sensitivity to local details within each patch. While it generally makes sense to apply this method to

time series in order to catch global trends without missing out on any local features, the implementation of PatchTST lacks any major adjustments. It would be interesting to observe the performance of a model that is more thoroughly adapted to its specific task, instead of bringing forward an architecture without any major own contributions.

The notion of channel independence is also not groundbreaking, as prior works like DeepAR (Salinas et al., 2020) and LogTrans have also assumed channel independence. These papers used shared backbone parameters during training, which decreased parameters and improved overfitting issues. DeepAR utilized an architecture similar to the widely known long short-term memory model (Hochreiter & Schmidhuber, 1997), while LogTrans applied transformers as introduced by (Vaswani et al., 2017). More generally, channel independence can be applied to any transformer model, leading to the question if the compared baselines would be improved if they incorporated it. While this is briefly covered in the appendix along with a general analysis of the benefits, it could be emphasized more thoroughly. The second question mark that remains here is that of correlations between the different channels of a time series. Cross-channel dependencies are completely left out in the current approach. While this is mentioned as an important future step, it would be interesting to investigate its impact already in this paper.

The concept of masked modeling in representation learning, while introduced in the paper, may not be entirely novel, and its motivation in certain sections remains unclear. The argument that timeseries data's temporal redundancy makes the reconstruction process trivial lacks specificity and clarity. It is mentioned that successive steps contain similar values sometimes, not leading to any improvements to the model, but the exact identification and treatment of such cases is not discussed. The paper could benefit from more detailed explanations to substantiate this point.

The experimental setup raises concerns about the default lookback window range. The best performing out of 6 values is chosen. This approach may be limited, as the look-back window varies strongly for the different datasets. Best practices for setting this parameter need to be clarified, as a suboptimal choice could lead to underestimating baseline performance. Figure 2 in the paper shows how strongly different look-back windows affect each model for the different datasets, further supporting questions around choosing the best performing value out of a shared default array. On top of that, this shows how strongly these parameters deviate with varying datasets. Testing on more diverse data would help to better assess the model’s general performance. Moreover, a pertinent concern within the realm of representation learning pertains to the size of pretraining

datasets. In comparison to expansive domains like computer vision or natural language processing, the datasets employed here remain relatively modest. This, in turn, raises questions about the potential for these datasets to offer substantial knowledge enrichment. While marginal improvements can be observed in Tables 4 and 5 of the paper, it remains questionable whether the pretraining stage truly holds meaningful relevance for the datasets in question.

Regarding the comparison of self-supervised methods to standard supervised fine-tuning, reporting mean and standard deviation from multiple runs could offer a more comprehensive view of their relative performance. This way, the significance of the experienced results could also be investigated. The robustness and reproducibility of the results is a general concern in this context.

In summary, while the paper's idea of utilizing patches and channel independence for time series is intriguing and backed by empirical results, concerns about the novelty of certain elements and experimental settings need to be addressed to strengthen its scientific contribution and significance.

Potential Improvements

Following the strengths and weaknesses, we will now present some areas that we believe could be improved. These follow closely to the identified weaknesses but do introduce some new aspects especially on the technical side.

The paper needs to clearly articulate its contribution and novelty. It's important to explicitly state what makes this work unique compared to existing methods, especially given the similarities to previous models. This could be achieved through a dedicated section or by highlighting key differentiators throughout the paper. Furthermore, it should be elaborated how the patching process is tailored to the specific needs of time series data. Instead of simply adopting an existing architecture, it would be beneficial to explain any modifications or innovations that were made to better suit the time series context. The paper could also explore the impact of channel independence and consider addressing cross-channel dependencies, rather than leaving them as a future step. Providing insights or experimental results on how these aspects affect model performance could enhance the paper's completeness.

A more detailed and specific explanation of the concept of masked modeling in the context of time series data could be introduced. It should clarify why it's relevant and how

it contributes to the model's effectiveness. Examples or cases where masked modeling has a notable impact on performance to support the case would be useful.

In terms of the experimental setup and the parameter choice the paper should address concerns about the default lookback window range and discuss how it may vary across different datasets. Providing best practices or guidelines for choosing this parameter would be valuable. Additionally, expanding the dataset diversity for testing could provide a better assessment of the model's generalizability. Larger datasets could prove beneficial, considering that the datasets used are relatively modest in size compared to domains like computer vision and natural language processing. The potential for these datasets to provide meaningful knowledge enrichment and why they were chosen over others should be discussed.

To address concerns about robustness and reproducibility, the paper could report mean and standard deviation from multiple runs when comparing self-supervised methods to supervised fine-tuning. This would provide a more comprehensive view of the results and help assess their significance.

There are several more technical aspects to consider here. The model structure is relatively shallow, so changes towards more attention heads might prove to be useful. On top of that, the paper does not detail any hyperparameter optimization. Instead, a proof of concept is showcased. Tuning the hyperparameters for specific use cases might further improve the quality of the results.

Implementation

Several improvements have been proposed on the vanilla transformer model introduced by (Vaswani et al., 2017) for performing multivariate timeseries forecasting. We try our hands on two of them specifically – Informer (Zhou et al., 2020) and PatchTST (Nie et al., 2022).

Datasets and Model

We introduce diversification in the problem by forecasting vectors in three different datasets:

1. Electricity Transformer Temperature [ETTh1] - The ETT dataset has a 2-year data from two separated counties in China at 1-hour level. Each data point consists of the target value ”oil temperature” and 6 power load features.

2. Electricity Consuming Load [ECL] - This is the hourly electricity consumption (Kwh) readings of 321 clients over a period of 2 years with ‘MT 320’ as the target value.
3. Weather [WTH] - This dataset contains local climatological data for nearly 1,600 U.S. locations, 4 years from 2010 to 2013, data points are collected every 1 hour. Each data point consists of the target value “wet bulb” and 11 climate features.

The dimensions of the datasets are as indicated in Table 1.

Table 1: Dimensions of datasets used in our analysis.

Dataset	Features	Observations
ECL	321	26,304
ETTh1	7	17,420
WTH	12	35,064

All datasets are normalized using the StandardScaler method from Scikit Learn. As standard practice, the time series datasets are divided into train, validation and test splits. In lines of optimizing the computing capacity, we fix the forecast window to 96 hours (equivalent to 4 days) and for the sake of not compromising the quality of the forecast, we set a large fixed lookback window of 336 hours (equivalent to 14 days or 2 weeks). In lines with the original implementation, MSE and MAE are chosen as performance evaluation metrics. In essence, from the results documented in the PatchTST paper, we try and reproduce the ones highlighted below:

Table 2: Results adapted from PatchTST.

Models		PatchTST/64		PatchTST/42		DLinear		FEDformer		Autoformer		Informer		Pyraformer		LogTrans	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	**0.149**	**0.198**	0.152	0.199	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	**0.194**	**0.241**	0.197	0.243	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	**0.245**	**0.282**	0.249	0.283	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	**0.314**	**0.334**	0.320	0.335	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	**0.360**	**0.249**	0.367	0.251	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	**0.379**	**0.256**	0.385	0.259	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	**0.392**	**0.264**	0.398	0.265	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	**0.432**	**0.286**	0.434	0.287	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	**0.129**	**0.222**	0.130	**0.222**	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**	**0.240**	0.148	**0.240**	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	**0.163**	**0.259**	0.167	0.261	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	**0.197**	**0.290**	0.202	0.291	0.203	0.301	0.233	0.344	0.236	0.342	0.351	0.427	0.376	0.445	0.283	0.376
ILI	24	**1.319**	**0.754**	1.522	0.814	2.215	1.081	2.624	1.095	2.906	1.182	4.657	1.449	1.420	2.012	4.480	1.444
	36	1.579	0.870	**1.430**	**0.834**	1.963	0.963	2.516	1.021	2.585	1.038	4.650	1.463	7.394	2.031	4.799	1.467
	48	**1.553**	**0.815**	1.673	0.854	2.130	1.024	2.505	1.041	3.024	1.145	5.004	1.542	7.551	2.057	4.800	1.468
	60	**1.470**	**0.788**	1.529	0.862	2.368	1.096	2.742	1.122	2.761	1.114	5.071	1.543	7.662	2.100	5.278	1.560
ETTh1	96	**0.370**	0.400	0.375	**0.399**	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	0.413	0.429	0.414	0.421	**0.405**	**0.416**	0.423	0.446	0.456	0.457	1.007	0.786	0.790	0.681	1.037	0.824
	336	**0.422**	0.440	0.431	**0.436**	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	**0.447**	0.468	0.449	**0.466**	0.472	0.490	0.469	0.492	0.515	0.517	1.144	0.857	0.963	0.782	1.135	0.852

PatchTST configuration: PatchTST utilizes 3 encoder layers, with 4-8 attention heads. A stride value of 2-8 determines the time series data overlap. Data normalization is achieved through RevIN (Reversible Instance Normalization), enhancing training efficiency and stability. The Transformer's model dimension is 64-512, with fully connected layers having a dimension of 16-128, a dropout ratio of 0.3-0.4, and the

utilization of the GeLU activation function. The model is trained with a batch size of 16/128 and a learning rate of 0.0001-0.00005, running on an NVIDIA 3060 RTX GPU with 8 GB of memory. For more information check out the code available on GITHUB.

Informer configuration: The generative style decoder in the Informer configuration, while conceptually simple, predicts the long time-series sequences at one forward operation rather than a step-by-step way, which drastically improves the inference speed of long-sequence predictions. We set the start token length of the decoder as 48 for the Informer configurations meaning we feed 48 hours of data as an input to the decoder to trigger and generate the future 96 (additional) hours data vectors. The dimensions of the input that the Informer model's encoder and decoder expect are equal to the dimensions of the data. The Probsparse Attention Factor, i.e. the level of attention that the Informer model pays to different parts of the input data when processing it is fixed at 5. The number of attention heads are 8, and the number of encoder and decoder layers are fixated at 2 and 1 in an attempt towards optimizing computations. 'Distilling' is used in the encoder. The dimension of the feed-forward neural network layer is 2048 with dropout ratio of 0.1 and 'GeLU' activations. The batch size is set to 32 while the learning rate is initiated at 0.0001with an adjustment function. For each dataset, the model is trained for 6 heavy epochs with an early stopping criterion implemented for 3-epoch patience. The model training and testing is carried out on Google Colab with standard Tesla 4 GPUs with an average runtime of 3 hours for each dataset.

Results

The results obtained by both the models are presented in Table 3 aside to their counterparts listed in the PatchTST introduction paper of 2023. As evident from above, our model implementations managed to reproduce results in close proximity of the ones produced by the authors of 'PatchTST' and 'Informer' for some datasets .In the case of Informer, the average deviance from the results obtained in the paper only amounts to 0.13 for MSE and 0.07 for MAE and about 0.132 MSE and 0.126 MAE for patchTST.

Table 3: Comparison of PatchTST and our results.

Model	Datasets	Paper Results		Our Results	
		MSE	MAE	MSE	MAE
PatchTST	ECL	**0.130**	**0.222**	0.153	0.257
	ETTh1	**0.375**	**0.399**	0.415	0.457
	WTH	**0.152**	**0.199**	0.485	0.485
Informer	ECL	0.304	0.393	**0.297**	**0.391**
	ETTh1	**0.941**	**0.769**	1.096	0.848
	WTH	**0.354**	**0.405**	0.573	0.545

Our implementation of PatchTST relies on a scaled-down architecture compared to the one featured in the original paper, primarily due to computational constraints. An intriguing insight gleaned from our experiments is that an increase in model size to a certain value does not consistently lead to lower MSE and MAE values respectively. Instead, it suggests that further augmenting the model's size may be necessary to achieve improved performance, which had computational constraints in our case.

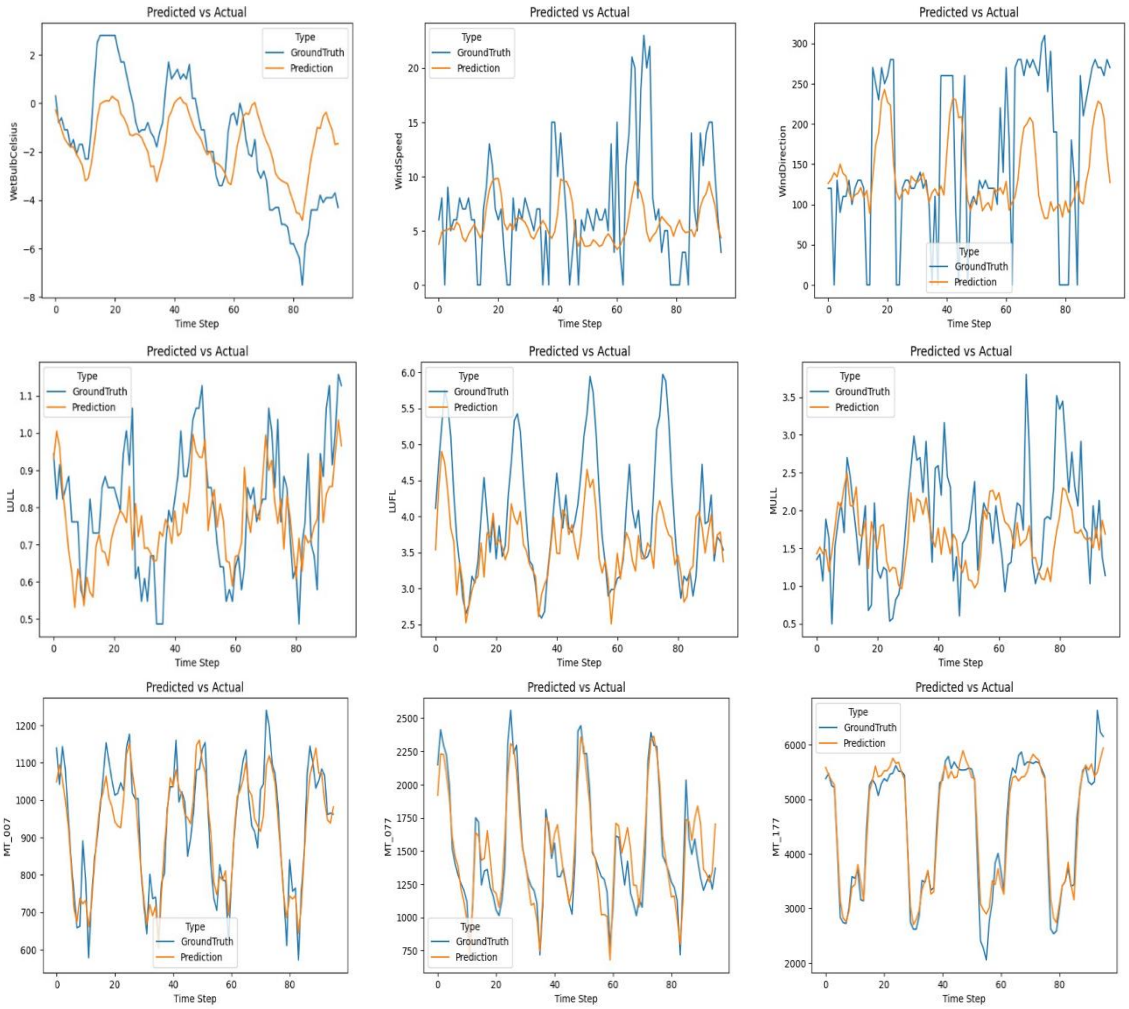

Figure 3: Predictions vs Ground Truth from top to bottom: WTH (Features are Wet bulb Celsius, Wind Speed, Wind Direction); ETTH1 (Features are LULL, LFULL, MULL); ECL (Features are MT007, MT077, MT177).

For informer configurations, we visualize below prediction performance of one (out of many) vector from each of the three datasets. Signals are clearly observable of the forecasts following the three components of a time series (trend, seasonality and irregularity).

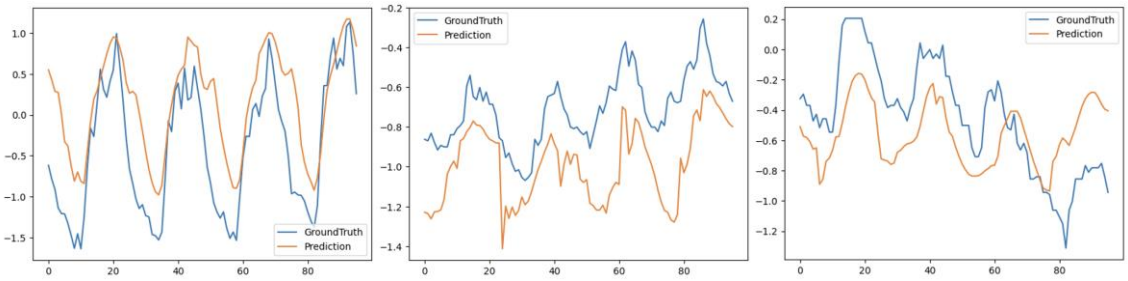

Figure 4: Informer model's predictions of "MT_320" vector in the ECL dataset; "OT" vector in the ETTh1 dataset; "WetBulbCelcius" vector in the WTH dataset.

Conclusion

Under the scope of this project, we exploit the attention capabilities of transformers for multivariate long sequence time-series forecasting (LSTF). A distinguished state-of-the-art configuration of transformers namely 'PatchTST' was thoroughly studied, and its performance was put to test against another precursor transformer variant called 'Informer'. Three distinct datasets, ECL, ETTh1 and WTH, were utilized for forecasting, each with varying features and observations. Both transformer implementations were configured and evaluated using MSE and MAE as performance metrics. The results were then compared to those reported in the original papers, revealing that the implemented models closely reproduced the results.

PatchTST introduced two main innovations in time series analysis using transformers. Firstly patching, which aggregates time steps to enhance locality and semantic information. Secondly channel independence, which breaks down multivariate time series into individual channels for improved learning. These approaches reduce time and space complexity, extend the look-back window, and enhance representation learning in large datasets. The study offers state-of-the-art results and paves the way for future research in transformer-based time series forecasting and representation learning.

However, the implementation of PatchTST also raises concerns about novelty and experimental settings. The model's similarity to standard vision transformers, the patching method, and the channel independence concept lack significant innovation. The paper's motivation for masked modeling in representation learning is unclear, and the experimental setup's default lookback window choice and dataset sizes are questioned. To enhance its scientific contribution, addressing these concerns and reporting results' robustness and reproducibility are essential.

Code and References

GitHub - https://github.com/neelblabla/transformers_for_time_series_forecasting

Weights & Biases - https://wandb.ai/kirteshpatel98/transformer_timeseries

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. (2022). Are transformers effective for time series forecasting? arXiv preprint arXiv:2205.13504.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Compututation, 9(8), 1735–1780.

Jose F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martınez-Alvarez, and Alicia Troncoso. (2021). Deep learning for time series forecasting: a survey. doi: 10.1089/big.2020.0159.

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

Lim Bryan and Zohren Stefan. (2021) Time-series forecasting with deep learning: a survey. Phil. Trans. R. Soc. A. doi: 10.1098/rsta.2020.0209.

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

Pedro Lara-Ben´ıtez, Manuel Carranza-Garc´ıa, and Jose C Riquelme. (2021). An experimental review on deep learning architectures for time series forecasting. International Journal of Neural Systems, arXiv: 2103.12057.

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

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

Weiqi Chen, Wenwei Wang, Bingqing Peng, Qingsong Wen, Tian Zhou, and Liang Sun. (2022). Learning to rotate: Quaternion transformer for complicated periodical time series forecasting. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 146–156.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021, May). Informer: Beyond efficient transformer for long sequence time-series forecasting. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 12, pp. 11106-11115).