Transformers for Time Series Forecasting

Marco Zanotti

University Milano-Bicocca

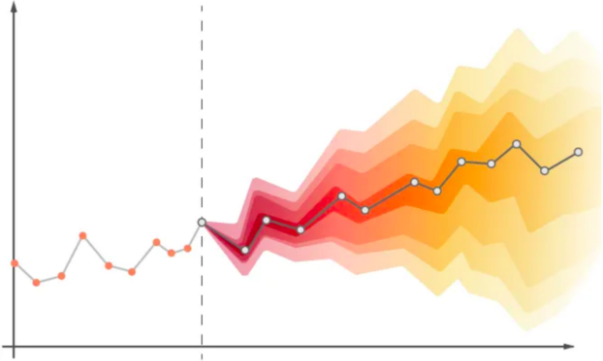
UNIVERSITA' DEGLI STUDI DI MILANO BICOCCA

Contents

1. The TSF Problem
2. Vanilla Transformer
3. TSF Transformers
4. Conclusions

1. The TSF Problem

Time series forecasting (TSF) is the task of predicting future values of a given sequence based on previously observed values.



The TSF problem may be essentially identified by the following aspects:

- **Prediction objective**: point forecasting vs probabilistic forecasting
- **Forecast horizon**: short-term vs long-term forecasting
- **Input-Output dimension**: univariate vs multivariate forecasting
- **Forecasting task**: single-step vs multi-step forecasting

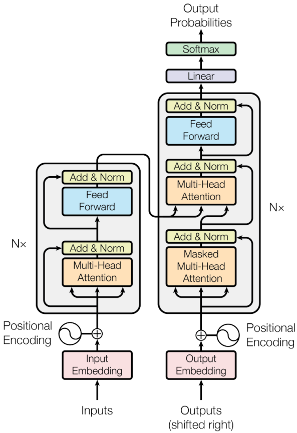
The TSF problem is usually faced with statistical models (ARIMA, ETS) or deep learning models (RNN, LSTM).

The main challenges of the TSF problem are:

- **uncertainty** increases as the forecast horizon increases
- difficulty in capturing **multiple complex patterns** over time
- difficulty in capturing **long-term dependencies** (critical for long-term forecasting)
- difficulty to handle **long input sequences**

2. Vanilla Transformer

- Based on Encoder-Decoder architecture
- Uses self-attention mechanism to access any part of the sequence history
- Positional encoding allows to account for element positions
- Residual connections and layer normalization help to stabilize the learning process
- Each encoder and decoder layer is composed of a self-attention layer and a feed-forward layer



Can vanilla Transformers be used for TSF?

The TSF problem can be seen as a sequence learning problem such as machine translation.

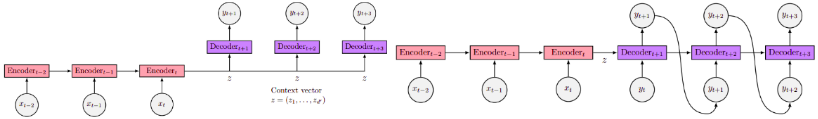
Main ingredients allowing to use vanilla Transformers for TSF:

- **Multi-head Self-attention** mechanism allows to access any part of the sequence history, capturing both short-term and long-term dependencies (but it is invariant to the order of elements in a sequence)
- **Positional encoding** allows to account for the sequence ordering
- **Masked self-attention** allows to avoid information leakage from future

Can vanilla Transformers be used for TSF?

Just few changes are needed to adapt Transformers to TSF:

- **Remove the final activation** function (softmax) from the output layer and set the dimension of the linear layer equal to the forecasting horizon
- **Adapt the structure** to the desired forecasting task (single-step or multi-step)



(a) Simple encoder-decoder

(b) Recursive encoder-decoder

Problems with vanilla Transformers

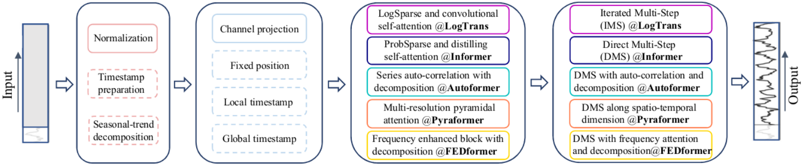
- **Locally agnostic**: the attention mechanism matches queries and keys without considering their local context being prone to temporal anomalies
- **Positional encoding**: only the order in which two elements occur is taken into account, but their temporal distance is not
- **Computational complexity**: given a sequence of length L, the time and memory burden is $O(L^2)$, making it difficult to learn patterns in long time series
- **Simple Architecture**: the architecture does not include any component of typical importance in TSF (e.g. autocorrelation, decomposition, recurrent layers, etc.)

3. TSF Transformers

Classification

Transformers are **very appealing for long-term TSF** due to their ability to learn long-range dependencies.

cambiarla con una più sensata con le modifiche su positional, architettura e moduli



Normalization

Timestamp preparation

Seasonal-trend decomposition

(a) Preprocessing

Channel projection

Fixed position

Local timestamp

Global timestamp

(b) Embedding

LogSparse and convolutional self-attention @**LogTrans**

ProbSparse and distilling self-attention @**Informer**

Series auto-correlation with decomposition @**Autoformer**

Multi-resolution pyramidal attention @**Pyraformer**

Frequency enhanced block with decomposition @**FEDformer**

(c) Encoder

Iterated Multi-Step (IMS) @**LogTrans**

Direct Multi-Step (DMS) @**Informer**

DMS with auto-correlation and decomposition @**Autoformer**

DMS along spatio-temporal dimension @**Pyraformer**

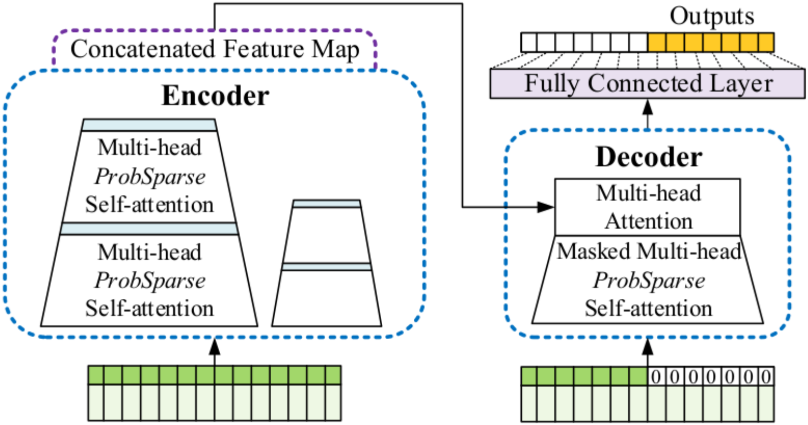
DMS with frequency attention and decomposition@**FEDformer**

(d) Decoder

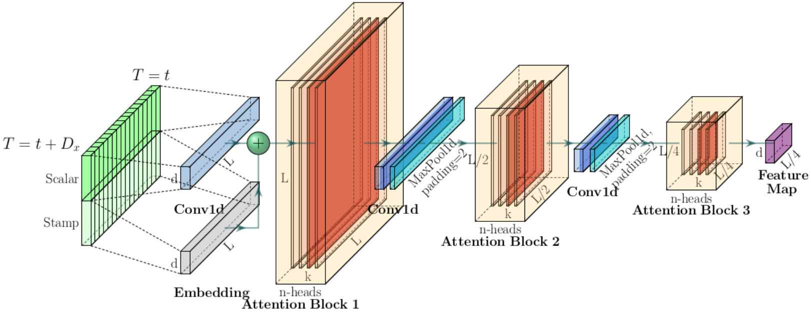
Informer

aa

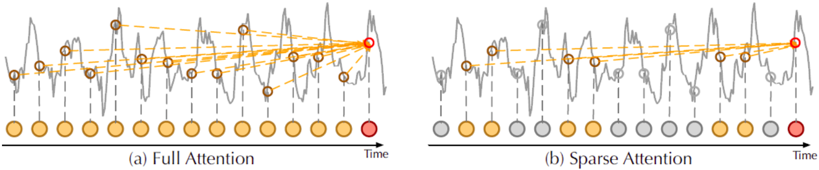
Informer - Architecture



Informer - Causal Convolution Layers



Informer - ProbSparse Attention

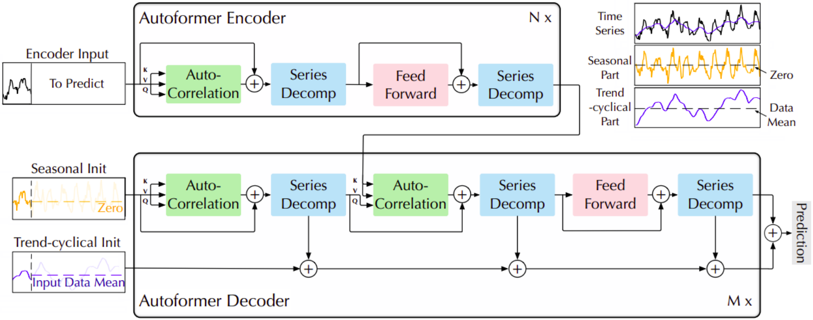


Autoformer

Autoformer builds upon the traditional method of decomposing time series into seasonality and trend-cycle components. This is achieved through the incorporation of a Decomposition Layer, which enhances the model's ability to capture these components accurately. Moreover, Autoformer introduces an innovative auto-correlation mechanism that replaces the standard self-attention used in the vanilla transformer.

the two key contributions of Autoformer: the Decomposition Layer and the Attention (Autocorrelation) Mechanism.

Autoformer - Architecture



Autoformer - Decomposition Layer

Autoformer incorporates a decomposition block as an inner operation of the model, as presented in the Autoformer's architecture above. As can be seen, the encoder and decoder use a decomposition block to aggregate the trend-cyclical part and extract the seasonal part from the series progressively

For an input series X with length L, the decomposition layer returns Xtrend and Xseasonal

Autoformer - Decomposition Layer

aa

Autoformer - Attention Mechanism

In addition to the decomposition layer, Autoformer employs a novel auto-correlation mechanism which replaces the self-attention seamlessly. In the vanilla Time Series Transformer, attention weights are computed in the time domain and point-wise aggregated. On the other hand, as can be seen in the figure above, Autoformer computes them in the frequency domain (using fast fourier transform) and aggregates them by time delay.

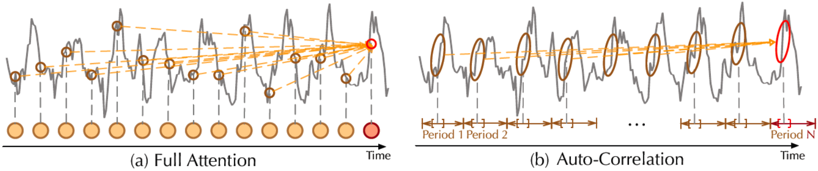
Autoformer - Attention Mechanism

Using autocorrelation, Autoformer extracts frequency-based dependencies from the queries and keys, instead of the standard dot-product between them. You can think about it as a replacement for the QK^T term in the self-attention.

In practice, autocorrelation of the queries and keys for all lags is calculated at once by FFT. By doing so, the autocorrelation mechanism achieves O(LlogL) time complexity (where L is the input time length), similar to Informer's ProbSparse attention.

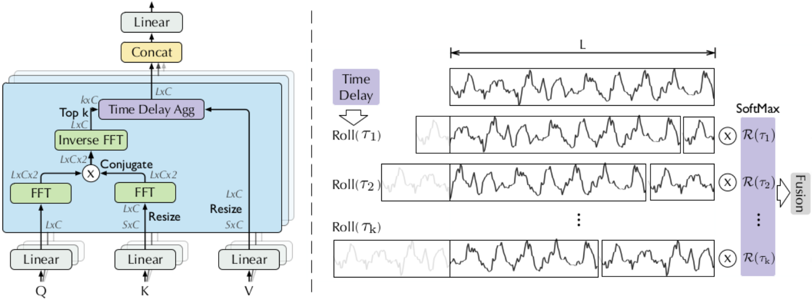
Next, we will see how to aggregate our attn_weights with the values by time delay, process which is termed as Time Delay Aggregation.

Autoformer - Autocorrelation Attention



(a) Full Attention (b) Auto-Correlation

Autoformer - Time Delay Aggregation



4. Conclusions

Conclusions

- aa

Bibliografy

Ailing Z., et al., 2023, 'Are Transformers Effective for Time Series Forecasting?', AAAI

Haixu W., et al., 2021, 'Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting', NeurIPS

Haoyi Z., et al., 2021, 'Informer: Beyond efficient transformer for long sequence time-series forecasting', AAAI

Lara-Benitez P., et al., 2021, 'Evaluation of the Transformer Architecture for Univariate Time Series Forecasting', Advances in Artificial Intelligence, CAEPIA

Qingsong W., et al., 2022, 'Transformers in Time Series: A Survey', AAAI

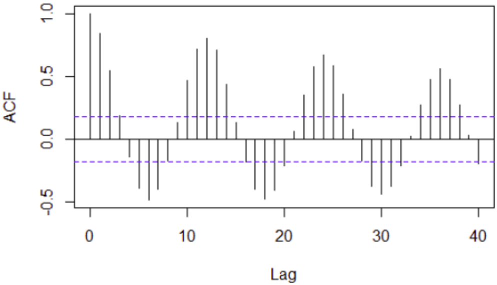
Thank you!

Appendix

Autocorrelation

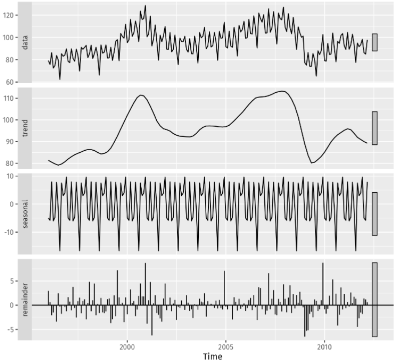
In theory, given a time lag k, autocorrelation for a single discrete variable Y is used to measure the "relationship" (pearson correlation) between the variable's current value at time t to its past value at time $t - k$.

$$Autocorrelation(k) = Corr(Y_t, Y_{t-k})$$



Time Series Decomposition

In time series analysis, decomposition is a method of breaking down a time series into three systematic components: trend-cycle, seasonal variation, and random fluctuations.


data
trend
seasonal
remainder
Time