Machine Learning Project, 2021 1–7

A review of Temporal Fusion Transformers

Carl Bylin
Higher School of Economics

1. Introduction

The focus of this project is to better understand how we can deal with high-dimensional time series with multiple inputs, missing values and irregular timestamps. In this situation, the performance of the classical approaches is not satisfying, and naïve applications of deep learning also fail. Even the deep learning models that have shown promising results tend to be "black boxes" with little insight into how to interpret the results.

The authors of the TFT model (Lim et al. (2020)) propose a modern attention-based architecture to combine multi-horizon forecasting functionality with interpretable insights related to temporal dynamics.

Classical time series approaches have generally performed better on one-step-ahead predictions. Still, decision-making in many real situations require forecasting to be done multiple steps into the future.

For example, monthly revenue forecast for the next semester in order to plan hiring, inventory, production, etc. Accurate forecasts of this type usually require access to multiple inputs including invariant or static features (e.g. country), information about the future (e.g. weather forecasts), or parallel and exogenous time series (e.g. website traffic).

The attention-based architecture proposed by the authors of the analyzed paper has been designed specifically to leverage prior knowledge with suitable inductive biases for multi-horizon forecasting. More specifically, the Temporal Fusion Transformer includes the following novel components:

- Static covariate encoders that produce context vectors.
- Gating mechanisms and variable selection to reduce the impact of irrelevant inputs.
- Local processing of known and observed inputs through a sequence-to-sequence layer.
- Long-term dependency learning through a temporal self-attention decoder.

Interpretation of the results is centered around the identification of globally important variables, persistent temporal patterns and significant events.

The main objective of this project is to reproduce the architecture proposed by the authors and evaluate its performance on generated toy datasets, as well as some of the datasets used by the authors in the original paper.

2. Temporal Fusion Transformer Architecture

2.1. Gated Linear Unit and Gated Residual Network

Gated Residual Network blocks are among the main basic components of this network. They enable efficient information flow along with the skip connections and gating layers.

The gating mechanisms basically allow the network to adapt both depth and complexity in order to perform well on a wide range of datasets and tasks.

Gated Linear Unit It is hard to know which variables are actually relevant for the prediction task from the outset. The gates of the Gated Linear Unit make it possible to suppress parts of the architecture that are not necessary in a particular scneario or with a specific dataset.

Gated Residual Network The Gated Residual Network is a flexible block that can apply non-linear processing when required. The Gated Linear Unit defined above helps the GRN how much to contribute to its input and could potentially skip the layer altogether if necessary. GLU outputs close to 0 would suppress the non-linear contribution.

2.2. Variable Selection Network

The Variable Selection Network is a critical component of the TFT architecture as it makes it possible for the model to accept a wide variety of inputs.

- Observed inputs are time dependent variables that are known only up until the moment when we want to forecast the target variable (this includes past values of the target variable).
- Known inputs are time dependent variables that can be known ahead of time (e.g. holidays, special events, etc.)
- Static covariates can also be used to enrich the model (e.g. region of a store).

With so many variables we might end up with unnecessary noise that can have a negative impact on the performance of the model and the Variable Selection Network makes it possible for the model to eliminate this noise.

It can also be used during prediction to evaluate which variables are most important for the prediction task. This is critical for interpretability of the trained model.

2.3. Interpretable Multi-Head Attention

This particular block is used to learn long-term relationships from observed time-varying inputs. It is a modified version of the more general multi-head attention block used in transformer-based architectures, in order to improve explainability.

Scaled Dot-Product Attention and Multi-Head Attention were both presented in the paper "Attention Is All You Need" by Vaswani et al.

It is well-known that the dot-product is a very simple but powerful tool to evaluate similarity between two vectors. For this same reason, it is also a great tool to help our model know what parts of the inputs to focus on based on the keys and queries. The

scaling factor helps improve the performance of dot product attention by not allowing the softmax to move into regions with very small gradients.

Multi-head attention allows us to compute multiple attention computations in parallel on different projections of the keys, queries and values. This makes it possible for the model to leverage different types of information in the input which would otherwise be lost by the averaging effect in a single attention head.

The original version fails in allowing us to be able to interpret the importance of each feature. *The TFT proposes a modification of multi-head attention such that there are shared value weights among the different heads with an additive aggregation of the heads for better interpretability.*

3. Experiments

We ran multiple experiments with the model, including an experiment on generated toy data as well as on the OMI realized library focused on the financial application of volatility forecasting.

3.1. Web Traffic Generator

First of all, we assume $\mathcal{I}$ unique entities in the time series dataset. In the Web Traffic Generator case we can think of the entities as different countries, regions or even websites.

The TFT architecture makes it possible to use many different types of inputs. First of all, each entity has a set of static covariates $\boldsymbol{s_i}$ that don't change over time. If we think of the entities as countries, then a potential static covariate could be the continent the country belongs to. In our toy dataset we generated three different clusters corresponding to high, medium and low traffic entities.

$X_{i,t}$ corresponds to each of the time-dependent features and therefor depends on the entity and the time step. Within the time-dependent features we have both known inputs ($\boldsymbol{x_{i,t}}$) that can be determined ahead of time (like the month) and we have observed inputs ($\boldsymbol{z_{i,t}}$) that are only seen retroactively.

Taking all of this into account we can say that our prediction task is defined by the following function incorporating static covariates as well as observed and known time-dependent features:

$$\hat{y}_i(q, t, \tau) = f_q(\tau, \boldsymbol{y_{i,t-k:t}}, \boldsymbol{z_{i,t-k:t}}, \boldsymbol{x_{i,t-k:t+\tau}}, \boldsymbol{s_i}) \tag{1}$$

$\boldsymbol{\tau}$ is the number of steps ahead we want to forecast.

After training on the toy data for six epochs we obtain a model that not only predicts future time steps, but in contrast to many classical methods gives us powerful insights into how the model makes its predictions.

First is the Multi-head attention mechanism described earlier. In Figure 1 we can see the data that was used as input to the model in blue, with the corresponding attention weights in yellow.

In this case it is quite interesting to see that there is a certain periodicity to the peaks. Especially when we compare it to the next graph (Figure 2) that shows the average weights of each of the past inputs as defined by the Variable Selection Network.

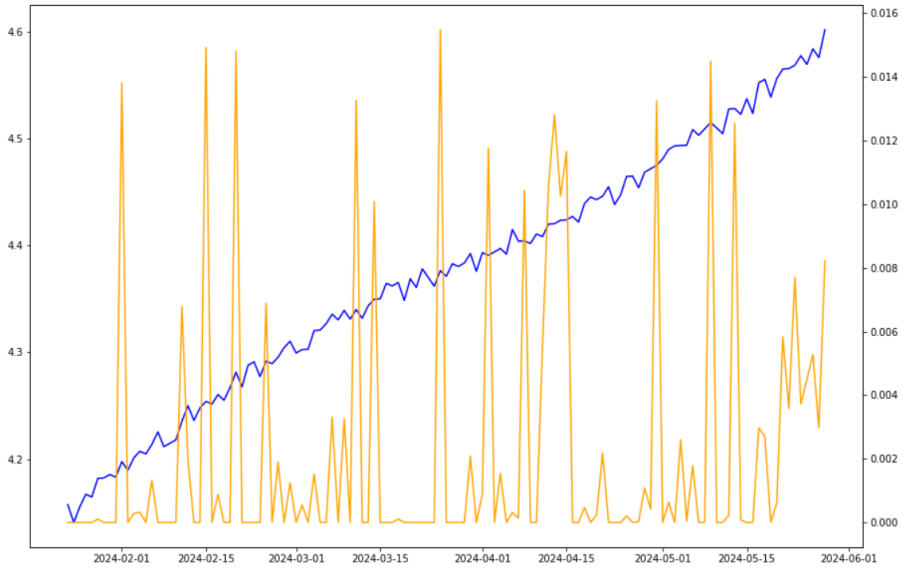

Figure 1: Attention Toy Example

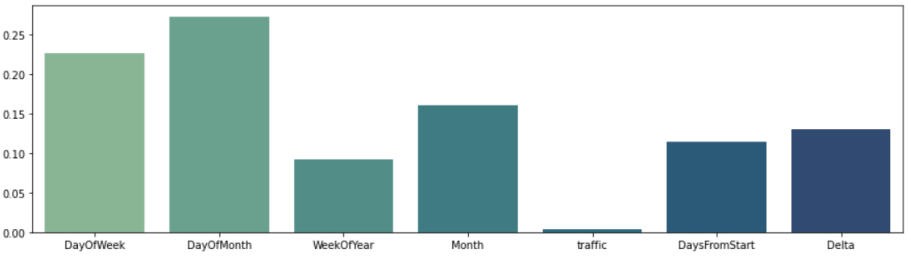

Figure 2: Past Weights Toy Example

Day of the week, day of the month and month have much higher weights than other features like the observed values of the target variable that has the lowest weight. This makes sense when we keep in mind that the Web Traffic Generator actually does not generate data based on the previous values. Instead it generates data based on weekly and yearly periodicities on top of a base trend.

The future weights tell us a slightly different story as they seem to pick up on the beforementioned base trend that is always increasing in this particular run of the model. So it makes sense that the further we move into the future, the higher the predictions should be.

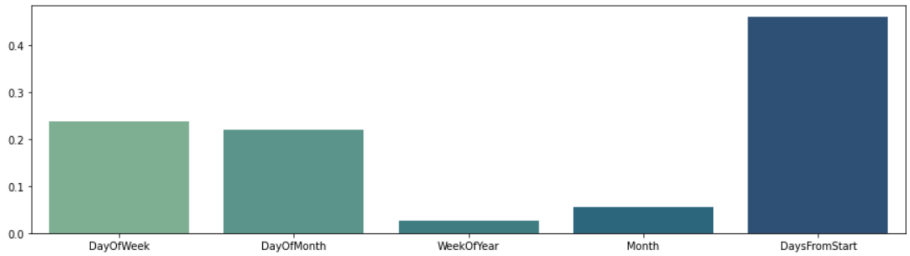

Figure 3: Future Weights Toy Example

3.2. Stock volatility forecast

In this case, the entities in the dataset correspond to each of the stock ticker symbols. The main static covariate used it the region the stock belongs to.

The observed features that can't be known beforehand are the log-volume of the sales and the open-to-close ratio. Known time-dependent features include the day of the week, day of the month, week of the year, month and days from the start of the evaluation period.

It was not possible to achieve the same performance as the original authors due mainly to computational resources. For example, it took slightly more than 6 hours to train the Electricity dataset using a NVIDIA Tesla V100 GPU.

Google Colab was used for these experiments and due to different limitations it was necessary to train on only a subset of the data.

Still, just like with the toy dataset, it is possible to study the different weights produced by the model for interpretability.

Figure 4 shows us one example where the model shows higher levels of attention specifically in large drops of the stock as well as some additional focus on the last days before the prediction. This is quite logical due to the high volatility of stocks.

Another interesting view can be seen in Figure 5. It is common in NLP tasks when using attention to analyze which words had a larger weight when generating a specific prediction (e.g. which words in English were more influential in generating a word in French in a translation task).

These visualizations allow us to better understand what influenced the model in its prediction. This can be a source of new insights as well as a protection against decisions made based on erroneous predictions. A domain expert can analyze these relationships in time to better understand if the weight actually makes sense or not,

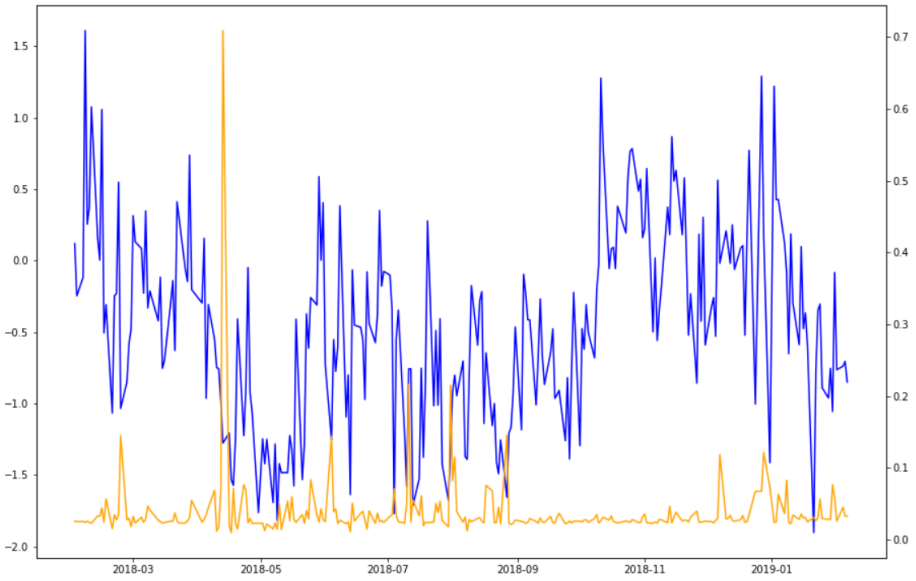

Figure 4: Attention Example

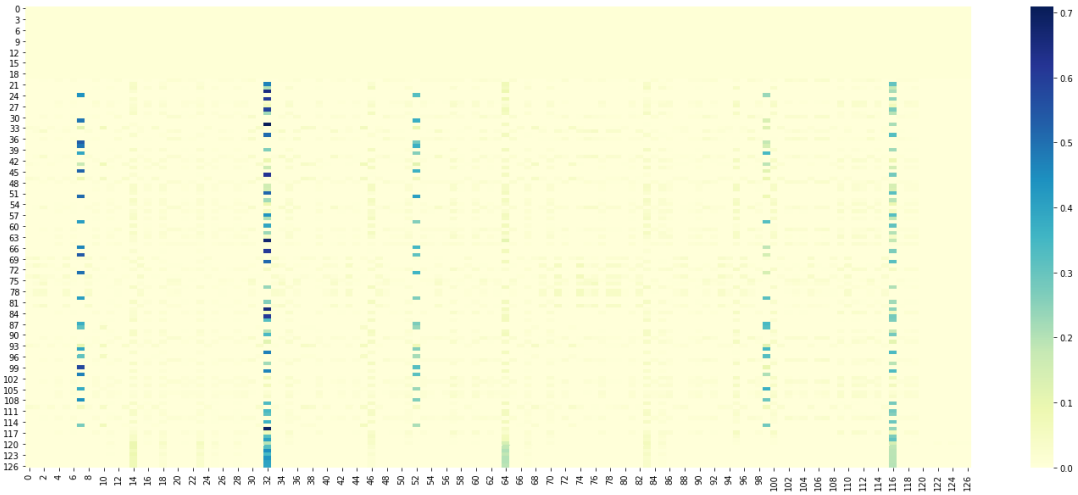

Figure 5: Attention Heatmap Example

Finally, Figure 6 shows us another of the benefits of this architecture. We used Quantile loss for training, which allows us to generate different ranges of predictions: worst case, best case and a base case.

Here we can see that in many cases the predictions (yellow) are not capable of producing a completely accurate estimate compared to the ground truth (black). Still, the light blue area representing the range of values seems to do a good job containing the true values most of the time.

The ability to also keep in mind worst and best case scenarios is critical in decision-making making this another benefit of this model.

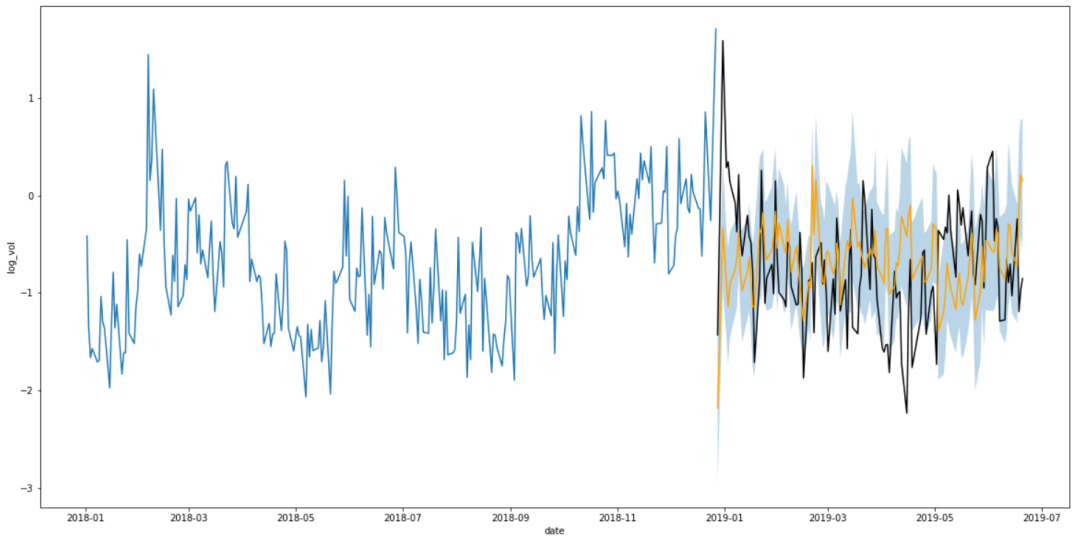

Figure 6: Quantiles Example

All of the code for this project can be found here: https://github.com/KalleBylin/temporal-fusion-transformers

4. Citations and Bibliography

References

Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting, 2020.

Acknowledgments

Special thanks to Evgenii Egorov for his mentorship during the project and for his understanding during difficult times personally.