High Integrity Systems Project Time Series Analysis

Assignment 7

Hellyben Shah

December 8, 2024

1 Temporal Fusion Transformer [1]

The Temporal Fusion Transformer (TFT) is an advanced deep learning architecture designed for multi-horizon and multivariate time series forecasting.

1.1 What is multi-horizon time series forecasting?

Multi-horizon forecasting is the prediction of variables-of-interest at multiple future time steps. Unlike one-step-ahead predictions, multi-horizon forecasts provide estimates for several future time steps simultaneously, allowing users to optimize their actions at multiple steps ahead (e.g. retailers optimizing the inventory for the entire upcoming season, or clinicians optimizing a treatment plan for a patient) [1]. Multi-horizon forecasting often involves a complex mix of inputs as shown in the figure below.

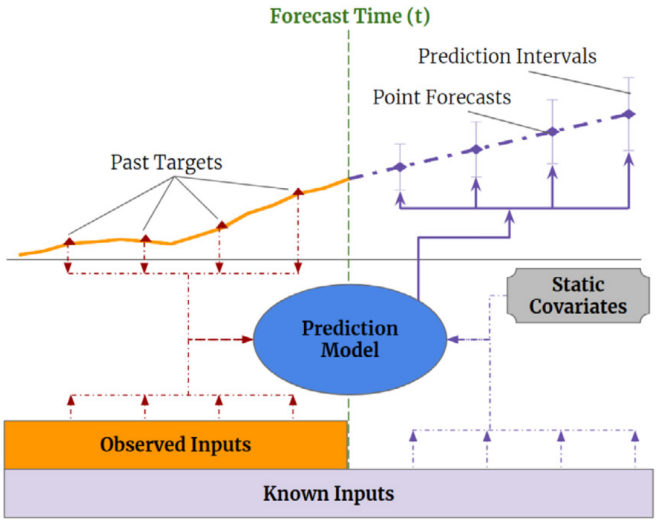

Figure 1: Multi-Horizon Inputs [1]

Inputs

- Static covariates (si) : provide contextual metadata about measured entities that does not depend on time. i.e., location, size, type
- Observed inputs (zi,t): Time-dependent input features which can only be measured at each step and are un-known beforehand. i.e., daily foot traffic, local weather conditions.
- Known inputs (xi,t): Time-dependent input features which can be predetermined. i.e., day of week, month, upcoming holidays.
- Past target values (yi,t-k:t): i.e., historical daily daily sales data

In many scenarios, the provision for prediction intervals can be useful for optimizing decisions and risk management by yielding an indication of likely best and worst-case values that the target can take. As such, we adopt quantile regression to our multi-horizon forecasting setting (e.g. outputting the 10th, 50th and 90th percentiles at each time step).

1.2 Architecture Overview

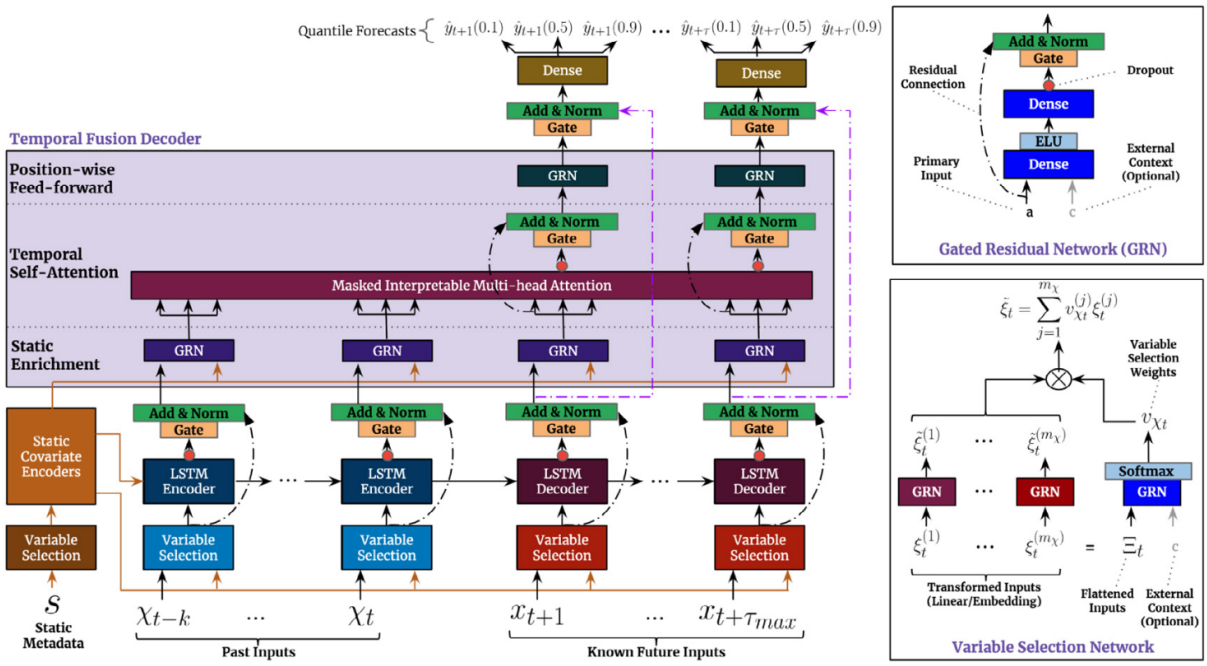

Figure 2: TFT Architecture [1]

1.3 Key Components and Mathematical Foundations

1.3.1 Static Covariate Encoders

Static covariates (s_i) are processed using entity-specific encoders to create context vectors. These context vectors are used throughout the network to incorporate time-invariant information.

1.3.2 Gating Mechanisms and Variable Selection

Although there may be a number of variables available, it is usually unknown how relevant and specific each one is to the result.

Furthermore, the amount of non-linear processing that is necessary is hard to predict, and in some cases, such as when datasets are noisy or small, simpler models may be useful. The ability to use non-linear processing only where necessary is provided by the Gated Residual Network (GRN). A primary input (a) and an optional context vector (c) are fed into the GRN, which controls how much of the original input (a) the GRN contributes. If necessary, the GRN may skip over the layer completely because the GLU outputs may all be near 0 to suppress the nonlinear effect.

GRNs are used to improve the model's flexibility and generalization. The GRN formula is:

$$GRN(a, c) = \text{LayerNorm}(a + \text{GLU}(W_1 \cdot \text{ELU}(W_2 \cdot a + W_3 \cdot c + b_2) + b_1))$$

Where:

- x is the input

- W_1, W_2 are weight matrices
- b_1, b_2 are bias terms
- ELU is the Exponential Linear Unit activation function
- GLU is the Gated Linear Unit
- LayerNorm is Layer Normalization

The GLU is defined as:

$$\text{GLU}(\eta) = \sigma(W_4 \cdot \eta + b_4) \odot (W_5 \cdot \eta + b_5)$$

Where σ is the sigmoid function and $\odot$ denotes element-wise Hadamard multiplication.

TFT employs gating layers and sample-dependent variable selection to minimize the contributions of irrelevant inputs. The variable selection network can be represented as:

$$v_s = \text{softmax}(W_s \cdot \text{GRN}(Flaternedip, c) + b_s)$$

Where GRN is the Gated Residual Network, W_s is a learnable weight matrix, and b_s is a bias term.

1.3.3 Temporal Self-Attention - Interpretable Multi-Head Attention Mechanism

The Temporal Fusion Transformer (TFT) uses an interpretable multi-head attention mechanism for multi-horizon time series forecasting. Key aspects of the TFT's multi-head attention include:

1. TFT modifies the standard multi-head attention to ensure interpretability:
   - Value weights are shared across all attention heads, allowing easier tracing of relevant values.
   - Outputs of all heads are additively aggregated instead of concatenated.
2. The interpretable multi-head attention formula is:

$$\text{MultiHeadAttention}(Q, K, V) = \sum_{h=1}^{H} \text{Attention}(QW_Q^h, KW_K^h, VW_V)$$

   Where:
   - Q, K, V are query, key, and value matrices
   - W_Q^h, W_K^h are head-specific projection matrices
   - W_V is a shared value projection matrix
   - H is the number of attention heads
3. This mechanism enables the model to:
   - Capture long-range temporal dependencies

- Identify relevant time steps for multi-horizon forecasting
- Distinguish the importance of different features at a given timestep

4. The interpretable attention contributes to TFT's ability to provide insights into temporal dynamics while maintaining high performance in multi-horizon forecasting tasks.

1.4 Forecasting Process

1. Input Processing: The model processes static covariates (s_i), observed inputs ($z_{i,t}$), and known inputs ($x_{i,t}$)[1].
2. Variable Selection: The network selects relevant features using the variable selection mechanism.
3. Local Processing: The sequence-to-sequence layer processes inputs locally.
4. Long-term Dependency Learning: The temporal self-attention decoder captures long-term dependencies.
5. Quantile Forecasting: Quantiles represent specific points in the probability distribution of the predicted values. The TFT model generates quantile forecasts for multiple horizons, which can be expressed as:

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

Where:

- $\hat{y}_i(q, t, \tau)$ is the predicted qth quantile of the τ-step-ahead forecast at time t
- $f_q(.)$ is the prediction model
- q represents the desired quantile (e.g., 0.1 for 10th percentile, 0.5 for median, 0.9 for 90th percentile)

To obtain quantile forecasts:

The TFT model is trained to minimize the quantile loss function, also known as the pinball loss:

$$L_q(y, \hat{y}) = q \max(y - \hat{y}, 0) + (1 - q) \max(\hat{y} - y, 0)$$

During inference, the model outputs multiple quantile predictions for each forecast horizon. These quantile predictions provide a range of possible outcomes, allowing for the construction of prediction intervals.

For example, the model might output the 10th, 50th, and 90th percentiles, giving a median forecast (50th percentile) along with a likely range of values (between the 10th and 90th percentiles).

1.5 Interpretability

TFT provides interpretability through:

1. Variable importance analysis
2. Temporal pattern identification

3. Significant event detection

These interpretability features are enabled by analyzing attention weights and variable selection mechanisms[1].

TFT's unique architecture, combining specialized components for different input types and temporal relationships, allows it to achieve high performance in multi-horizon forecasting while providing valuable interpretable insights.

2 Temporal Kolmogorov-Arnold Transformers [2]

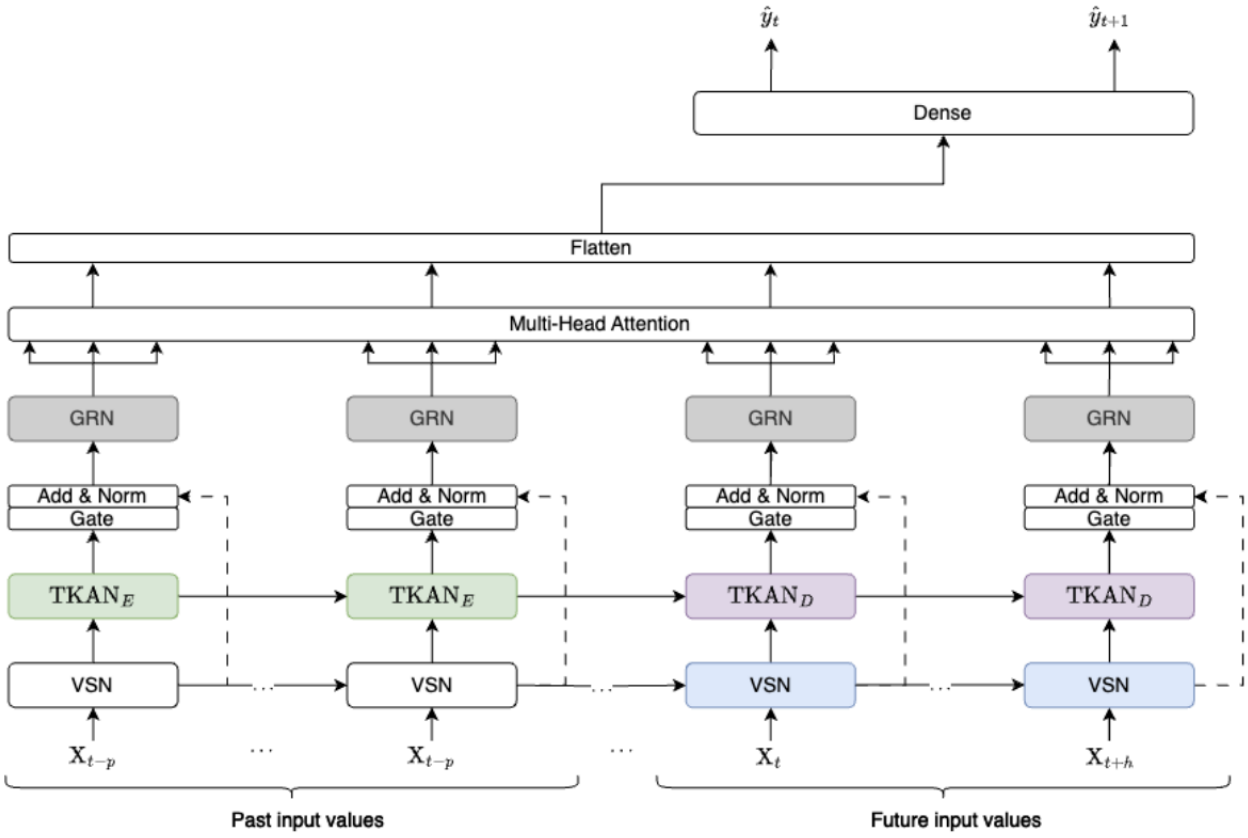

Figure 3: TKAT Architecture [2]

TKAT combines the strengths of **Temporal Kolmogorov-Arnold Networks (TKANs)** with transformer architecture to improve forecasting accuracy.

TKAT is specifically designed for problems where observed inputs are more prevalent than known inputs, making it particularly suitable for financial tasks.This is a significant departure from TFT's design, which was optimized for scenarios with more known inputs.

2.1 Key Components

1. **TKAN Layers**:

   - The core of TKAT is built upon TKAN layers, which integrate memory management and temporal

dependencies. The transformation function

$$\phi_{l,j,i}$$

is defined to be time-dependent, allowing the model to incorporate historical information effectively.

- TKAT replaces the LSTM layers used in TFT with Temporal Kolmogorov-Arnold Network (TKAN) layers for both encoding and decoding tasks. This change allows for better handling of temporal dependencies and memory management.
- The output from each layer is computed as:

$$x_j^{l+1}(t) = \sum_{i=1}^{n_l} \phi_{l,j,i,t}(x_i^l(t), h_{l,i}(t))$$

- Here, $h_{l,i}(t)$ captures the history of node i in layer l:

$$h_{l,i}(t) = W_{hh} h_{l,i}(t-1) + W_{hz} x_{l,i}(t)$$

2. **Recurrent Kernel and Memory Management**:
   - The Recurrent KAN (RKAN) layers facilitate short-term memory retention across processing layers.
   - Memory management is crucial for maintaining context within sequences, allowing the model to learn from previous states effectively.
3. **Gating Mechanisms**: This architecture is almost entirely similar to the TFT architecture, except that the entire sequence is used after the multihead attention layer, whereas it is only used for the future part in the TFT architecture.

   Inspired by LSTM architectures, gating mechanisms control information flow:

   - Forget Gate:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

   - Input Gate:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

   - Output Gate:

$$o_t = \sigma(KAN(\vec{x}, t))$$

   The hidden state h_t and cell state c_t are updated as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot \tanh(c_t)$$

4. **Encoder-Decoder Framework**: The encoder consists of multiple TKAN layers that process past inputs while maintaining memory. The decoder also utilizes TKAN layers but initializes its state from the encoder's final cell state, enabling effective sequence-to-sequence mapping.

5. **Attention Mechanism**: TKAT employs a self-attention mechanism to capture long-term dependencies in time series data. A common choice is the scaled dot-product attention.

$$A(Q, K, V) = A(Q, K)V$$

   Multi-head attention enhances representation learning by allowing different heads to focus on various aspects of the input data.

6. **Variable Selection Networks (VSN)**: VSNs enhance model performance by focusing on salient covariates through learned selection weights:

$$v_{\chi,t} = Softmax(GRN_v(\Xi_t))$$

References

[1] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[2] R. Genet and H. Inzirillo, "A temporal kolmogorov-arnold transformer for time series forecasting," *arXiv preprint arXiv:2406.02486*, 2024.