Deep Learning Models for Multivariate Time-series Analytics

BITS Pilani, Pilani Campus

Ayushi Gaur

INTRODUCTION

- A time-series is a series of data points ordered in time.
- Time is the independent variable and the goal is to make a forecast in the future.
- Time Series Modelling – An integral part of applications in topics such as climate modeling, biological sciences and medicine.
- Autometric Parametric model selection and traditional ML methods such as Kernel Regression.
- Modern ML methods provide a means to learn temporal dynamics in a purely data-driven manner.
- Forecasting is usage of time-series data to extrapolate the past observations into the future using Deep Learning models such as CNNs, RNNs, LSTMs.

Time-Series Forecasting with Deep Learning : A Survey

(Bryan Lim and Stefan Zohren)

- Common approaches to time-series prediction using deep neural networks.
- State-of-the-art techniques available for forecasting problems – multi-horizon forecasting and uncertainty estimation.
- Emergence of new trend in hybrid models – combine domain specific quantitative models with deep learning.
- Facilitate decision support – Methods in interpretability and counterfactual prediction.

Deep Learning Architecture

One-step ahead forecast assume the following model :

$$\hat{y}_{i,t+1} = f(y_{i,t-k:t}, x_{i,t-k:t}, s_i), \tag{2.1}$$

where $\hat{y}_{i,t+1}$ is the model forecast, $y_{i,t-k:t} = \{y_{i,t-k}, \ldots, y_{i,t}\}$, $x_{i,t-k:t} = \{x_{i,t-k}, \ldots, x_{i,t}\}$ are observations of the target and exogenous inputs respectively over a look-back window k, s_i is static metadata associated with the entity (e.g. sensor location), and $f(.)$ is the prediction function

Basic Building Blocks

- Learn predictive relationships by using a series of non-linear layers.
- Encode relevant historical information into a latent variable.
- Encoders and decoders form the basic building blocks of deep learning architectures.
- Choice of network determine the types of relationship learnt by model.

$$f(y_{t-k:t}, x_{t-k:t}, s) = g_{\text{dec}}(z_t),$$

$$z_t = g_{\text{enc}}(y_{t-k:t}, x_{t-k:t}, s),$$

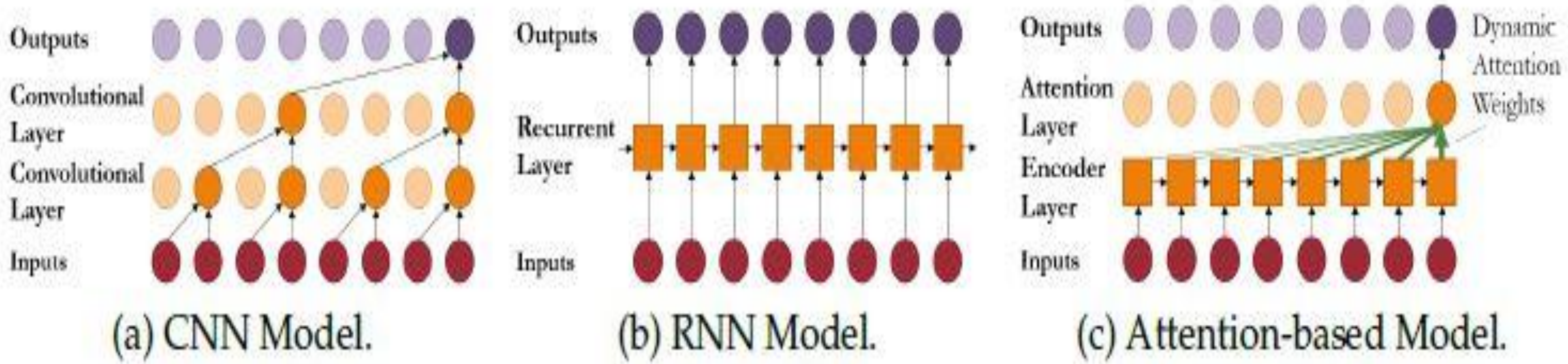


Figure 1: Incorporating temporal information using different encoder architectures.

CNNs

- A 1D CNN is very effective when we expect to derive interesting features from shorter (fixed-length) segments of the overall data set and where the location of the feature within the segment is not of high relevance.
- Researchers utilise multiple layers of causal convolutions – convolutional filters designed to ensure only past information is used for forecasting
- Two key implications for temporal relationships learnt by CNNs – assumption of time-invariant relationships and tuning of receptive window carefully.
- A single causal CNN layer with a linear activation function is equivalent to an auto-regressive (AR) model
- Dilated Convolutions – Need ?
- Convolutions of a down-sampled version of the lower layer features – reducing resolution to incorporate information from the distant past

RNN and LSTM

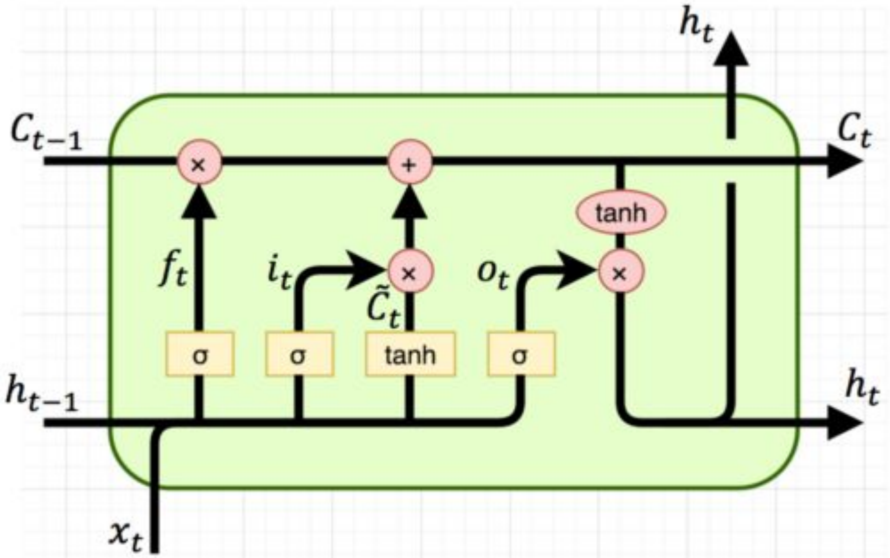
- Used in sequence modelling.
- RNN cells contain an internal memory state which acts as a compact summary of past information. The memory state is recursively updated with new observations at each $z_t = \nu\left(z_{t-1}, y_t, x_t, s\right),$
- Due to the infinite lookback window, older variants of RNNs can suffer from limitations in learning long-range dependencies in the data – due to issues with exploding and vanishing gradients .
- LSTMs improve gradient flow within the network.

Input gate: $i_t = \sigma(W_{i_1} z_{t-1} + W_{i_2} y_t + W_{i_3} x_t + W_{i_4} s + b_i),$

Output gate: $o_t = \sigma(W_{o_1} z_{t-1} + W_{o_2} y_t + W_{o_3} x_t + W_{o_4} s + b_o),$

Forget gate: $f_t = \sigma(W_{f_1} z_{t-1} + W_{f_2} y_t + W_{f_3} x_t + W_{f_4} s + b_f),$

z_{t-1} is the hidden state of the LSTM, and $\sigma(.)$ is the sigmoid activation function.



Attention Mechanisms

- Led to improvements in long-term dependency learning.
- Transformer architectures achieving state-of-the-art performance.
- Attention-based methods to enhance the selection of relevant time-steps in the past
- Attention layers allow the network to directly focus on significant time steps in the past – even if they are very far back in the lookback window because they aggregate temporal features using dynamically generated weights.
- Transformer architectures (RNN encoders + attention layers)

Outputs and Loss Functions

- For the binary classification case, the final layer of the decoder features a linear layer with a sigmoid activation function – allowing the network to predict the probability of event occurrence at a given time step.
- For one-step-ahead forecasts of binary and continuous targets, networks are trained using binary cross-entropy and mean square error loss functions.

$$\mathcal{L}_{classification} = -\frac{1}{T}\sum_{t=1}^{T} y_t \log(\hat{y}_t) + (1 - y_t)\log(1 - \hat{y}_t)$$

$$\mathcal{L}_{regression} = \frac{1}{T}\sum_{t=1}^{T} (y_t - \hat{y}_t)^2$$

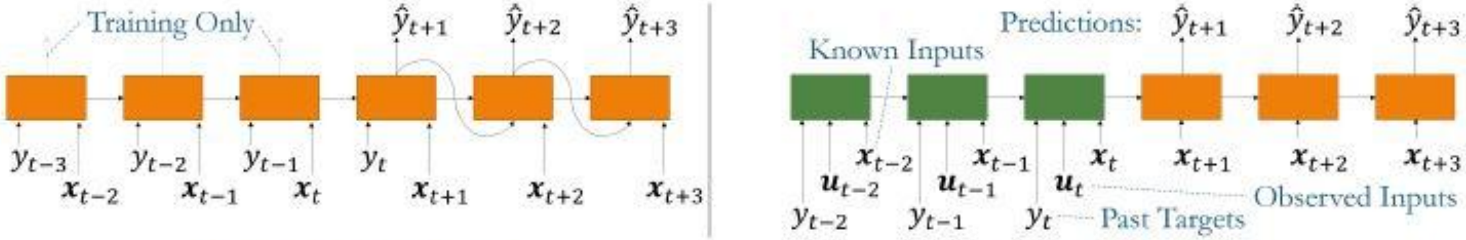
Multi-Horizon Forecasting Models

- It is often beneficial to have access to predictive estimates at multiple points in the future – allowing decision makers to visualise trends over a future horizon.

$$\hat{y}_{t+\tau} = f(y_{t-k:t}, x_{t-k:t}, u_{t-k:t+\tau}, s, \tau), \tag{2.23}$$

where $\tau \in \{1, \ldots, \tau_{max}\}$ is a discrete forecast horizon, u_t are known future inputs (e.g. date information, such as the day-of-week or month) across the entire horizon, and x_t are inputs

- Two methods – Iterative and Direct.



(a) Iterative Methods (b) Direct Methods

Hybrid Models

- Need?
- Hybrid methods combine well-studied quantitative time series models together with deep learning – using deep neural networks to generate model parameters at each time step.
- Especially useful for small datasets, where there is a greater risk of overfitting for deep learning models.
- An example of this is the Exponential Smoothing RNN (ES-RNN), which uses exponential smoothing to capture non-stationary trends and learn additional effects using RNN.
- In general, hybrid models utilise deep neural networks in two manners: a) to encode time-varying parameters for non-probabilistic parametric models and b) to produce parameters of distributions used by probabilistic models.

Temporal Fusion Transformers for Multi-Horizon Time Series Forecasting

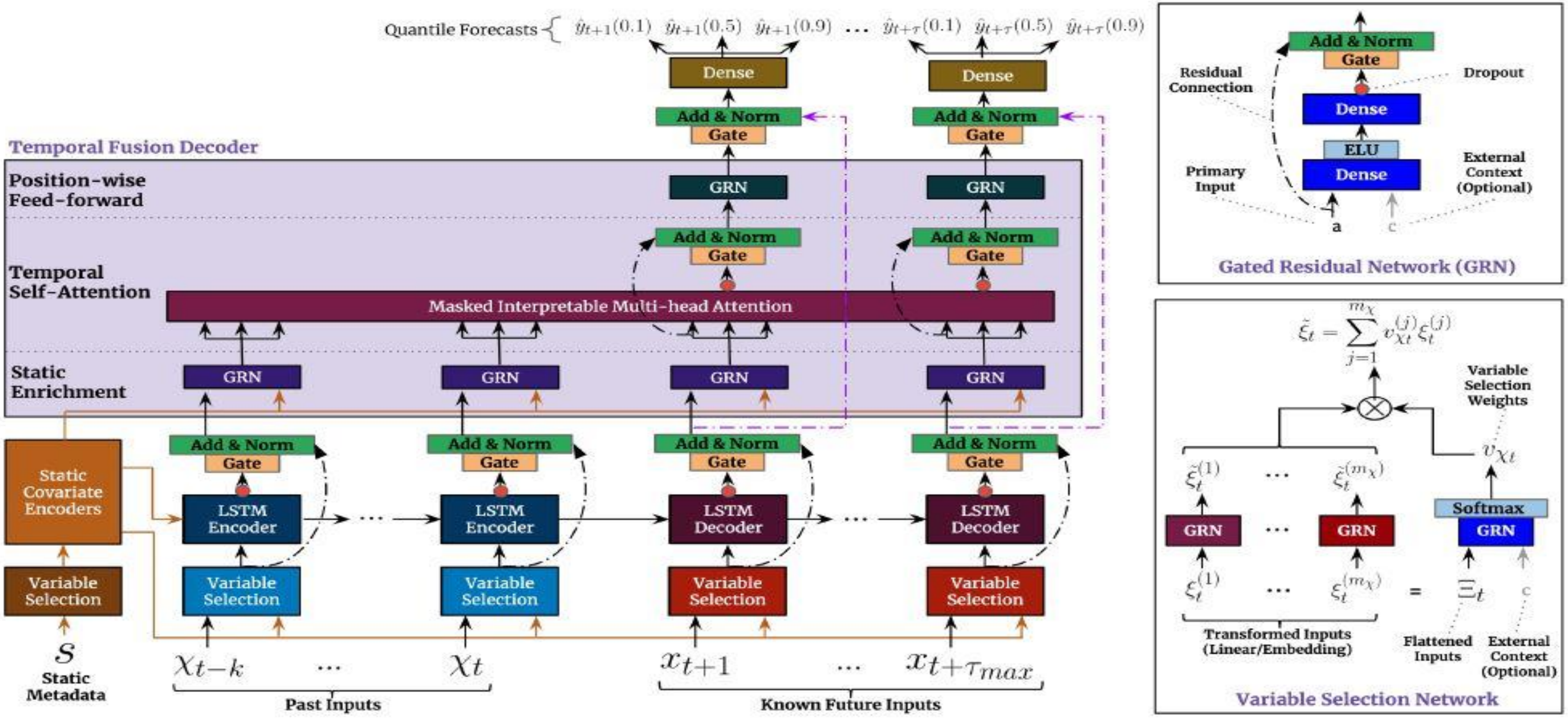
(Bryan Lim, Nicolas Loeff, Tomas Pfister)

- Multi-horizon forecasting is the prediction of variables-of-interest at multiple future time steps providing users with access to estimates across the entire path, allowing them to optimize their actions at multiple steps in future
- Temporal Fusion Transformer (TFT) a novel attention based architecture which combines high-performance multi horizon forecasting with interpretable insights into temporal dynamics.
- TFT uses recurrent layers for local processing and interpretable self-attention layers for long-term dependencies.
- Data Sources - complex mix of inputs including static (i.e. time-invariant) covariates, known future inputs, and other exogenous time series that are only observed in the past (without any prior information on how they interact with the target)
- This heterogeneity of data sources together with little information about their interactions makes multi-horizon time series forecasting particularly challenging.
- Most current architectures are `black-box' models where forecasts are controlled by complex nonlinear interactions between many parameters. This makes it difficult to explain how models arrive at their predictions (no interpretability).
- Ideas included - (1) static covariate encoders which encode context vectors for use in other parts of the network, (2**) gating mechanisms** throughout and sample-dependent variable selection to minimize the contributions of irrelevant inputs, (3) a **sequence-to-sequence layer** to locally process known and observed inputs, and (4) a **temporal self-attention decoder** to learn any long-term dependencies present within the dataset.

Related work

- Traditional multihorizon forecasting methods, recent deep learning methods can be categorized into iterated approaches using autoregressive models or direct methods based on sequence-to-sequence models.
- Iterated approaches utilize one-step-ahead prediction models, with multistep predictions obtained by recursively feeding predictions into future inputs.
- Direct methods are trained to explicitly generate forecasts for multiple predefined horizons at each time step. Their architectures typically rely on sequence-to-sequence models, e.g. LSTM encoders to summarize past inputs, and a variety of methods to generate future predictions.

TFT Architecture



TFT Architecture --- Continued

- **Gating mechanisms** to skip over any unused components of the architecture, providing adaptive depth and network complexity to accommodate a wide range of datasets and scenarios.
- **Variable selection networks** to select relevant input variables at each time step.
- **Static covariate encoders** to integrate static features into the network, through encoding of context vectors to condition temporal dynamics.
- **Temporal processing** to learn both long- and short-term temporal relationships from both observed and known time-varying inputs. A **sequence- to- sequence layer** is employed for local processing, whereas long-term dependencies are captured using a novel interpretable multi-head **attention block.**
- Prediction intervals via quantile forecasts to determine the range of likely target values at each prediction horizon.

Results

- TFT yields 7% lower P50 and 9% lower P90 losses on average compared to the next best model demonstrating the benefits of explicitly aligning the architecture with the general multi-horizon forecasting problem.

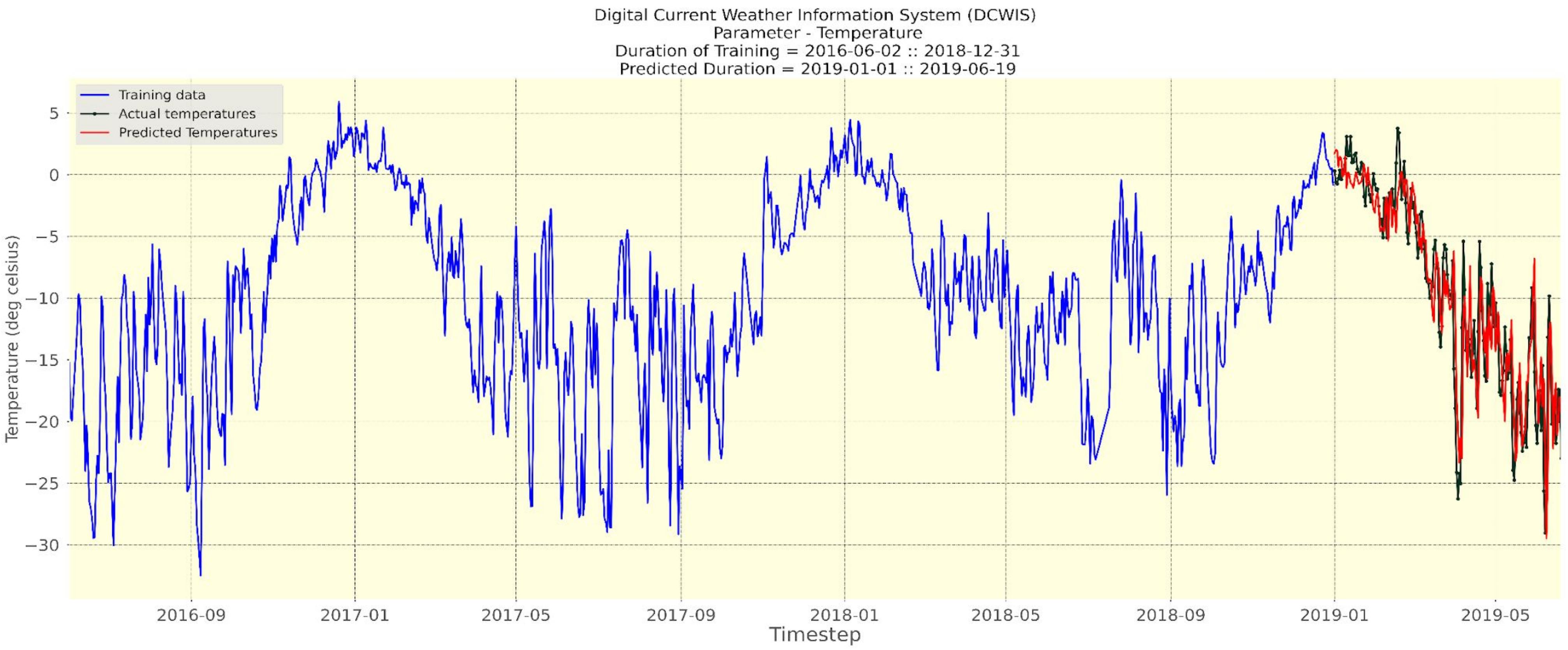
	DeepAR	CovTrans	Seq2Seq	MQRNN	TFT
Vol.	0.050 (+28%)	0.047 (+20%)	0.042 (+7%)	0.042 (+7%)	**0.039***
Retail	0.574 (+62%)	0.429 (+21%)	0.411 (+16%)	0.379 (+7%)	**0.354***

(c) P50 losses on datasets with rich static or observed inputs.

	DeepAR	CovTrans	Seq2Seq	MQRNN	TFT
Vol.	0.024 (+21%)	0.024 (+22%)	0.021 (+8%)	0.021 (+9%)	**0.020***
Retail	0.230 (+56%)	0.192 (+30%)	0.157 (+7%)	0.152 (+3%)	**0.147***

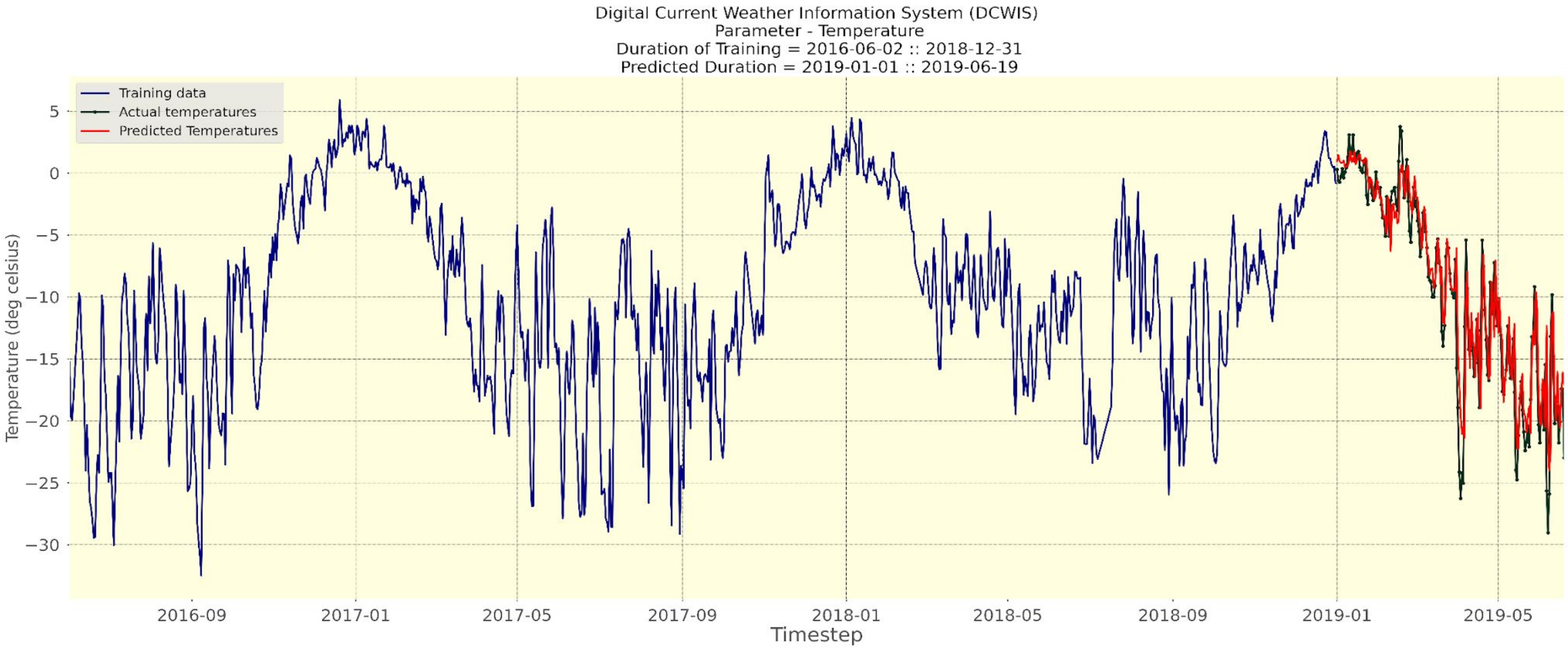
(d) P90 losses on datasets with rich static or observed inputs.

MODEL USED - CNN



MAE (Mean Absolute Error)	MSE (Mean Squared Error)	RMSE (Root Mean Squared Error)
2.58	12.12	3.48

MODEL USED – LSTM



MAE (Mean Absolute Error)	MSE (Mean Squared Error)	RMSE (Root Mean Squared Error)
2.09	8.92	2.98