Machine Learning for Forecasting: A Comparative Analysis

Performance Assessment on Electricity Load and Traffic Datasets

K. Papadakis S. Kollias (supervisor) P. Tzouveli (cosupervisor)

Thesis Presentation, March 2024

National Technical University of Athens
School of Electrical and Computer Engineering
MSc Data Science and Machine Learning

Introduction

Challenges in Time Series Forecasting

- Traditional methods (ARIMA, ETS, et al.) [1] struggle with nonlinearities and complex interactions often present in real-world data.
- Fully automated methods for traditional models [2] often produce suboptimal results.
- Machine Learning methods, can learn more complex patterns, with minimal tuning.

State of the Art in ML

- **Prophet**: Additive model with trend, seasonality, and holidays components. Developed by Facebook [3].
- **DeepAR**: Probabilistic forecasting with RNNs. Developed by Amazon [4].
- **DeepVAR**: Vector probabilistic forecasting with RNNs. Developed by Amazon [5].
- **N-BEATS**: Fast and interpretable model without RNNs. Developed by Bengio et al. [6].
- **TFT**: Quantile forecasting using Variable Selection Networks, RNNs, and Attention. Developed by Google [7].

Important Benchmark Datasets

- Electricity Load Diagrams: Electricity consumption of multiple clients.
- PeMS-SF: Occupancy rate of various car lanes of the San Francisco Bay Area freeways.

Models

Models

Prophet

Overview

- Fits a single series.
- Easily interpretable parameters.
- Fitting can be automated much easier than classical models [2].
- Its additive structure is easily interpretable and allows for the injection of domain knowledge.
- Robust to missing values.
- Fast fitting.

Classical Automated Procedures

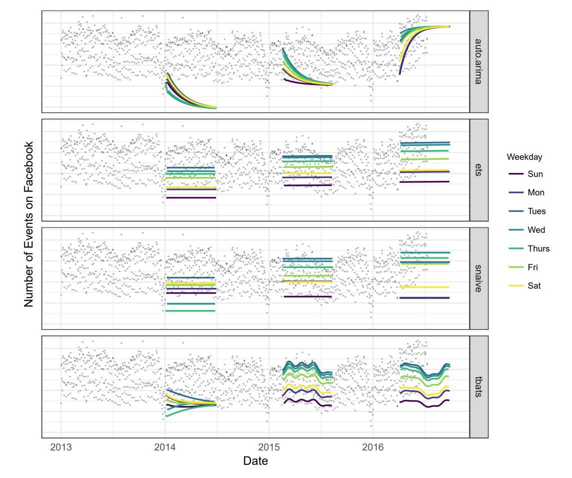


Figure 1: Forecasts from classical automated procedures. Figure from [3].

Prophet Forecasts

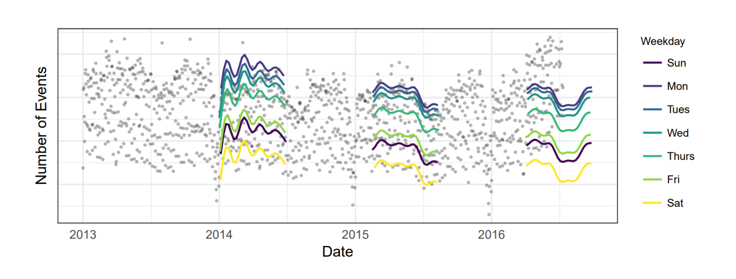


Figure 2: Prophet forecasts. Figure from [3].

Formulation

The Prophet Model

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

- $g(t)$ is the trend component
- $s(t)$ is the seasonality component
- $h(t)$ is the holiday component
- ϵ_t is the error term, assumed to be normally distributed around 0.

Multiplicative seasonality can be accomplished through a log transform.

Trend Component

The trend component can be either a *piecewise logistic growth* model or a *piecewise linear growth* model.

Basic Logistic Growth

Basic Logistic Growth

$$g(t) = \frac{C}{1 + \exp(-k(t - m))}$$

- C is the carrying capacity
- k is the growth rate
- m is an offset parameter

Piecewise Logistic Growth

Piecewise Logistic Growth

$$g(t) = \frac{C(t)}{1 + \exp(-(k + \mathbf{a}(t)^T \boldsymbol{\delta})(t - (m + \mathbf{a}(t)^T \boldsymbol{\gamma})))}$$

- $C(t)$ is a time-varying capacity
- $s_j,\ j = 1, \dots, S$ are change points
- $a_j(t) = 1$ if $t \geq s_j$ else 0
- δ_j are changes in rate
- γ_j make the function continuous and are computed by a closed formula

Piecewise Linear Growth

With the variables being the same as the logistic case:

Piecewise Linear Growth

$$g(t) = (k + \mathbf{a}(t)^T \boldsymbol{\delta})t + (m + \mathbf{a}(t)^T \boldsymbol{\gamma})$$

Change Point Selection

Change points can be manually set or selected automatically by imposing a sparsity-inducing prior

$$\delta_j = \mathrm{Laplace}(0, \tau)$$

As the hyperparameter $\tau \to 0$ the fit reduces to standard non-piecewise logistic or linear growth.

Change Point Extrapolation

Future change points are extrapolated as follows:

- τ is replaced by its MLE $\lambda = \frac{1}{S}\sum_{j=1}^{S} |\delta_j|$, or estimated with a hierarchical prior on τ.
- Future change points are then generatively sampled to match the average frequency of historical change points, i.e. $\delta_j \sim \mathrm{Laplace}(0, \lambda)$ with probability S/T else $\delta_j = 0$.
- Using multiple samples of future trends, uncertainty intervals can be constructed.
- Those intervals are only indicative since the assumption that the changepoints will have the same frequency and magnitude in the future is quite strong.

Seasonality Component

Modeled as a Fourier Series:

Seasonality Component

$$
\begin{gathered}
s(t) = \mathbf{X}(t)\boldsymbol{\beta} \\
\mathbf{X}(t) = \left[\cos\left(\frac{2\pi(1)t}{P}\right), \dots, \sin\left(\frac{2\pi(N)t}{P}\right)\right] \\
\boldsymbol{\beta} = [a_1, b_1, \dots, a_N, b_N]^T \sim \mathcal{N}(0, \sigma^2)
\end{gathered}
$$

- P is the period of the seasonality
- N is the Fourier order, a hyperparameter.

Multiple seasonality components can be used, e.g. daily or weekly.

Holiday Component

Each holiday is assumed to have an independent effect on the model.

Holiday Component

$$
\begin{aligned}
h(t) &= Z(t)\boldsymbol{\kappa} \\
Z(t) &= \left[\mathbf{1}_{t \in D_1}, \ldots, \mathbf{1}_{t \in D_L}\right] \\
\boldsymbol{\kappa} &\sim \mathcal{N}(0, \sigma^2)
\end{aligned}
$$

where D_i are holidays

Typically, the days around the holidays are considered as holidays as well.

Models

N-BEATS

Architecture Visualisation

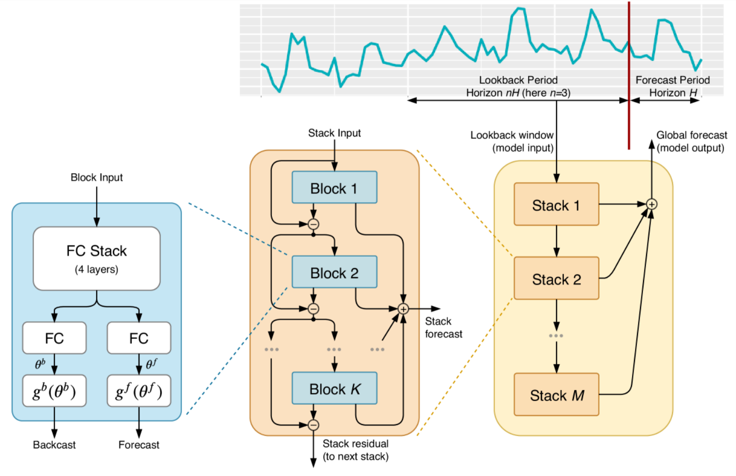


Figure 3: The N-BEATS Architecture. Figure from [6].

Overview

- Based on FC layers with ReLU. No CNNs or RNNs.
- Forecasts are partial and then summed up.
- Backcasts of a block are subtracted from the block's input and then are fed to the next block.
- Blocks within a stack share weights.
- Interpretable version using a trend and a seasonality stack.
- A single model can be trained on data from multiple related time series.

Generic Architecture

Generic Basis Layers

$$\hat{\mathbf{y}}_l = \mathbf{V}_l^f \boldsymbol{\theta}_l^f + \mathbf{b}_l^f, \quad \hat{\mathbf{x}}_l = \mathbf{V}_l^b \boldsymbol{\theta}_l^b + \mathbf{b}_l^b$$

- Affine transformation of $\dim \boldsymbol{\theta}_l^{(\cdot)}$ basis vectors of dimension H.
- Each vector can be thought of as a waveform.
- In the Generic architecture, the bases are learnable, thus the waveforms don't have inherent structure.

Interpretable Architecture

In the Interpretable Architecture, two stacks are used: the *trend* stack and the *seasonality* stack. The bases are fixed in this case.

Interpretable Basis Layers

$$\hat{\mathbf{y}}_{s,l}^{\text{trend}} = \mathbf{T}\boldsymbol{\theta}_{s,l}^{f}$$

$$\hat{\mathbf{y}}_{s,l}^{\text{seas}} = \mathbf{S}\boldsymbol{\theta}_{s,l}^{f}$$

$$\mathbf{T} = [\mathbf{1}, \mathbf{t}, \dots \mathbf{t}^{p}] \text{ where } \mathbf{t} = [0, \dots, H-1]^{T}/H$$

$$\mathbf{S} = [\cos(2\pi k\mathbf{t}),\ \sin(2\pi k\mathbf{t});\ k = 0,\ \dots,\ \lfloor H/2-1 \rfloor]$$

- The trend stack outputs a polynomial, typically of a low degree (e.g. 2 or 3).
- The seasonality stack outputs a Fourier series.

Models

DeepAR

Architecture Visualization

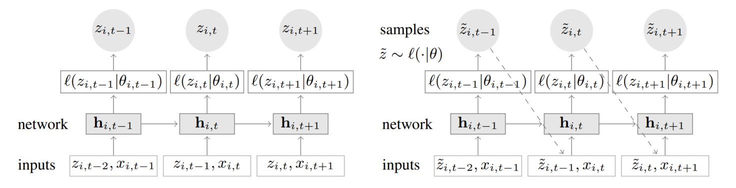


Figure 4: The DeepAR Architecture. Figure from [4].

Overview

- Autoregressive model.
- Probabilistic forecasts — facilitating risk management and decision making.
- Using RNNs to learn parameters of a distribution, for each horizon point.
- A single model can be trained on data from multiple related time series.
- Adept at handling datasets with series of largely varying magnitudes — a common phenomenon.
- Supports covariates with known past and future values.

Likelihood

The distribution

$$P\Big(\mathbf{z}_{i,t_0:T} \mid \mathbf{z}_{i,1:t_0-1}, \mathbf{x}_{i,1:T}\Big)$$

is assumed to have a parametric form that consists of a product of likelihood factors

$$\prod_{t=t_0}^{T} \ell(z_{i,t} \mid \theta(\mathbf{h}_{i,t}, \Theta))$$

parametrized by

$$\mathbf{h}_{i,t} = h(\mathbf{h}_{i,t-1}, z_{i,t-1}, \mathbf{x}_{i,t}, \Theta)$$

where h is a function implemented by a multi-layer recurrent neural network with LSTM cells, and Θ represents the model parameters.

Training

Samples $\{\mathbf{z}_{i,1:T}\}_{i=1,\ldots,N}$ are taken across time series, along with covariates $\mathbf{x}_{i,1:T}$. The model is then trained to maximize the log-likelihood of the observed data

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{t=t_0}^{T} \log \ell(z_{i,t} \mid \theta(\mathbf{h}_{i,t}))$$

with respect to the parameters of the RNN $h(\cdot)$ and the parameters of $\theta(\cdot)$ using stochastic gradient descent.

Inference

Multiple samples can be produced via ancestral sampling in an autoregressive fashion. These samples can be used to compute quantiles of interest.

Distributions

- Normal distribution for real-valued data
- Negative Binomial for positive count data
- Beta for data in $[0, 1]$
- etc.

Varying Scales

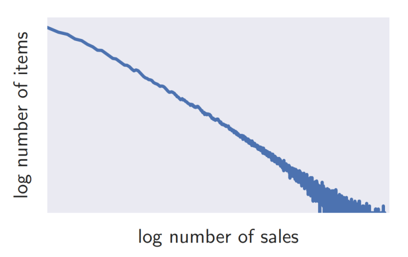


Figure 5: Exhibition of a power-law distribution of time series entities. Sales velocity (i.e. average weekly sales of an item) across millions of items sold by Amazon. Figure from [4].

Varying Scales Handling

The autoregressive input $z_{i,t-1}$ and the network output (e.g. μ) in the model are directly influenced by the observations $z_{i,t}$, but the non-linearities of the network have a limited operating range.

This challenge is addressed by using an item-dependent scaling factor ν_i that is used to divide the input and adjust the distribution parameters appropriately. This scaling factor is typically chosen to be the heuristic $\nu_i = 1 + \frac{1}{t_0} \sum_{t=1}^{t_0} z_{i,t}$.

Models

DeepVAR

Overview

- Autoregressive model with probabilistic forecasts.
- Combines RNNs (LSTM) and a Gaussian copula process.
- Single RNN that is unrolled for each time series separately.
- Efficient parametrization of the covariance matrix.
- Learns the joint series distribution at each time point.
- Ingrained handling of datasets with series of largely varying magnitudes.
- Supports covariates with known past and future values.

Overall Structure

Let $\mathbf{h}_t$ be the collection of RNN state values $\mathbf{h}_{i,t}$ for all time series $i = 1, \dots, N$.

The joint distribution is parametrized using a Gaussian copula:

$$p(\mathbf{z}_t \mid \mathbf{h}_t) = \mathcal{N}\left([f_1(z_{1,t}), \dots, f_N(z_{N,t})]^T \mid \boldsymbol{\mu}(\mathbf{h}_t), \Sigma(\mathbf{h}_t)\right)$$

where $f_i = \Phi^{-1} \circ \hat{F}_i$

Copula

Definition (Copula)

A copula function $C : [0,1]^N \to [0,1]$ is the CDF of a multivariate distribution with uniform marginals.

Sklar's Theorem

Theorem (Sklar)

Any joint cumulative distribution F admits a representation in terms of its univariate marginals F_i and a copula function C,

$$F(z_1, \dots, z_N) = C(F_1(z_1), \dots, F_N(z_N))$$

When the marginals are continuous the copula C is unique and is given by the joint distribution of the probability integral transforms of the original variables, i.e. $\mathbf{u} \sim C$ where $u_i = F_i(z_i)$. Furthermore, if z_i is continuous then $u_i \sim \mathcal{U}(0, 1)$

Gaussian Copula

A common modeling approach, that DeepVAR also uses, is the Gaussian copula

Gaussian Copula

$$C(F_1(z_1), \dots, F_1(z_1)) = \phi_{\boldsymbol{\mu},\Sigma}(\Phi^{-1}(F_1(z_1)), \dots, \Phi^{-1}(F_N(z_N)))$$

Because the F_i are unknown, they are replaced by their empirical CDFs $\hat{F}_i(v) = \frac{1}{m}\sum_{t=1}^{m} \mathbb{1}_{z_{i,t} \leq v}$, linearly interpolated for piecewise differentiability. The number of observations m is a hyperparameter, with a typical value being $m = 100$.

Varying Scales Handling

The problem of varying scales is addressed by the usage of empirical CDFs for each series separately, as this decouples the estimation of marginal distributions from the temporal dynamics and the dependency structure.

Gaussian Copula Density

The log density is computed by

$$\begin{aligned}
\log p(\mathbf{z}; \boldsymbol{\mu}, \Sigma) &= \log \phi_{\boldsymbol{\mu},\Sigma}(\Phi^{-1}(\hat{F}(\mathbf{z}))) + \log \frac{d}{d\mathbf{z}} \Phi^{-1}(\hat{F}(\mathbf{z})) \\
&= \log \phi_{\boldsymbol{\mu},\Sigma}(\Phi^{-1}(\hat{F}(\mathbf{z}))) + \log \frac{d}{d\mathbf{u}} \Phi^{-1}(\mathbf{u}) + \log \frac{d}{d\mathbf{z}} \hat{F}(\mathbf{z}) \\
&= \log \phi_{\boldsymbol{\mu},\Sigma}(\Phi^{-1}(\hat{F}(\mathbf{z}))) - \log \phi(\boldsymbol{\Phi}^{-1}(\hat{F}(\mathbf{z}))) + \log \hat{F}'(\mathbf{z})
\end{aligned}$$

Covariance Structure

The covariance matrix $\Sigma(\mathbf{h}_t)$ is parametrized as

$$\Sigma(\mathbf{h}_t) = \mathbf{D}(\mathbf{h}_t) + \mathbf{V}(\mathbf{h}_t)\mathbf{V}(\mathbf{h}_t)^T$$

where

$$\mathbf{D}(\mathbf{h}_t) = \begin{bmatrix} d_1(\mathbf{h}_{1,t}) & & 0 \\ & \ddots & \\ 0 & & d_N(\mathbf{h}_{N,t}) \end{bmatrix} \in \mathbb{R}^{N\times N}$$

$$\mathbf{V}(\mathbf{h}_t) = \begin{bmatrix} \mathbf{v}_1(\mathbf{h}_{1,t}) \\ \vdots \\ \mathbf{v}_N(\mathbf{h}_{N,t}) \end{bmatrix} \in \mathbb{R}^{N\times r}$$

Shared Parametrization

Let $\mathbf{y}_{i,t} := [\mathbf{h}_{i,t}; \mathbf{e}_i]^T \in \mathbb{R}^p$, with $\mathbf{e}_i \in \mathbb{R}^E$ being covariates/embeddings for each time series i.

The mappings μ_i, d_i and $\mathbf{v}_i$ are parametetrized in terms of shared functions $\tilde{\mu}$, $\tilde{d}$ and $\tilde{\mathbf{v}}$,

$$\begin{aligned}
\mu_i(\mathbf{h}_{i,t}) &= \tilde{\mu}(\mathbf{y}_{i,t}) = \mathbf{w}_\mu^T \mathbf{y}_{i,t}, \\
d_i(\mathbf{h}_{i,t}) &= \tilde{d}(\mathbf{y}_{i,t}) = s(\mathbf{w}_d^T \mathbf{y}_{i,t}), \\
\mathbf{v}_i(\mathbf{h}_{i,t}) &= \tilde{\mathbf{v}}(\mathbf{y}_{i,t}) = W_{\mathbf{v}} \mathbf{y}_{i,t},
\end{aligned}$$

where $s(x) = \log(1 + e^x)$ maps to positive values (softplus), and $\mathbf{w}_\mu \in \mathbb{R}^{p \times 1}$, $\mathbf{w}_d \in \mathbb{R}^{p \times 1}$, $W_{\mathbf{v}} \in \mathbb{R}^{r \times p}$ are parameters.

Gaussian Process Perspective

Let $\mathbf{x}_t := f(\mathbf{z}_t) = [f_1(z_{1,t}), \dots, f_N(z_{N,t})]$.

All learnable parameters are shared across all the time series and thus the distribution of $\mathbf{x}_t$ can be viewed as a Gaussian Process evaluated at points $\mathbf{y}_{i,t}$,

$$\begin{aligned} \mathbf{x}_{i,t} &= g_t(\mathbf{y_{i,t}}) \\ g_t &\sim \mathcal{GP}(\tilde{\mu}(\cdot), k(\cdot, \cdot)) \\ k(\mathbf{y}, \mathbf{y}') &= \mathbb{1}_{\mathbf{y}=\mathbf{y}'} \tilde{d}(\mathbf{y}) + \tilde{\mathbf{v}}(\mathbf{y})^T \tilde{\mathbf{v}}(\mathbf{y}') \end{aligned}$$

Thus the model can be trained by calculating the Gaussian terms in the loss function on randomly selected subsets of the time series during each iteration. This means that the model can be trained with batches of size $B \ll N$.

Models

Temporal Fusion Transformer (TFT)

Overview

- State of the art model.
- Combines LSTM encoder-decoder, Gating Mechanisms, Variable Selection Networks, and Attention.
- Supports all types of covariates: static, future known, and, unlike DeepAR and DeepVAR, future unknown covariates.
- Multi-quantile predictions.
- Intepretability through attention and variable selection weights.

Covariates Visualization

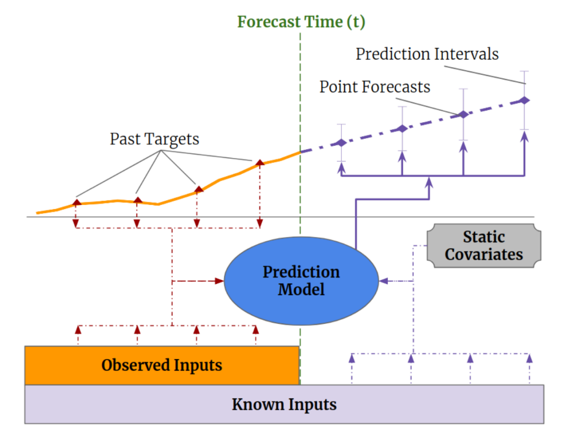


Figure 6: Illustration of multi-horizon forecasting with static covariates, past-observed and apriori-known future time-dependent inputs. Figure from [7].

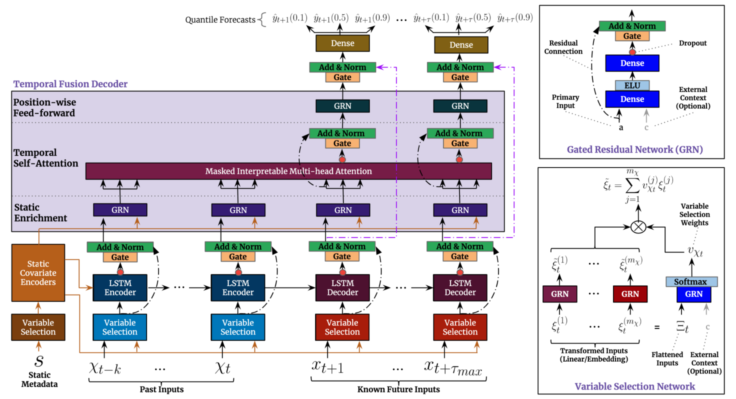


Figure 7: The architecture of the Temporal Fusion Transformer. Figure from [7].

Definitions

- I distinct time series entities.
- $\mathbf{s}_i \in \mathbb{R}^{m_s}$ static covariates
- $\boldsymbol{\chi}_{i,t} = \left[\mathbf{z}_{i,t}^T, \mathbf{x}_{i,t}^T\right]^T \in \mathbb{R}^{m_\chi}$
    - $\mathbf{z}_{i,t} \in \mathbb{R}^{m_z}$ observed (typically contains the target)
    - $\mathbf{x}_{i,t} \in \mathbb{R}^{m_x}$ known
- $y_{i,t} \in \mathbb{R}$ scalar targets

Gating Mechanisms

Gating mechanisms are introduced via the Gated Residual Network (GRN). GRNs make the model simpler if needed and they determine the impact of covariates.

Gated Residual Network (GRN)

$$
\begin{aligned}
\mathrm{GRN}_\omega(\mathbf{a}, \mathbf{c}) &= \mathrm{LayerNorm}(\mathbf{a} + \mathrm{GLU}_\omega(\boldsymbol{\eta}_1)) \\
\boldsymbol{\eta}_1 &= \mathbf{W}_{1,\omega}\boldsymbol{\eta}_2 + \mathbf{b}_{1,\omega} \\
\boldsymbol{\eta}_2 &= \mathrm{ELU}(\mathbf{W}_{2,\omega}\mathbf{a} + \mathbf{W}_{3,\omega}\mathbf{c} + \mathbf{b}_{2,\omega})
\end{aligned}
$$

where

$$
\mathrm{GLU}_\omega(\boldsymbol{\gamma}) = \sigma(\mathbf{W}_{4,\omega}\boldsymbol{\gamma} + \mathbf{b}_{4,\omega}) \odot (\mathbf{W}_{5,\omega}\boldsymbol{\gamma} + \mathbf{b}_{5,\omega})
$$

Separate Variable Selection Networks for static, past and future inputs. WLOG, the case for past inputs is presented.

Let

- $\boldsymbol{\xi}_t^{(j)} \in \mathbb{R}^{d_{\text{model}}}$ the transformed j-th variable at time t.
- $\boldsymbol{\Xi}_t := [\xi_t^{(1)^T}, \dots, \xi_t^{(m_\chi)^T}]^T$.
- $\mathbf{c}_s$ an external context vector. For static covariates $\mathbf{c}_s = \mathbf{0}$.

Variable Selection ii

Variable Selection Weights

$$\mathbf{v}_{\chi_t} = \mathrm{Softmax}(\mathrm{GRN}_{\mathbf{v}}(\boldsymbol{\Xi}_t, \mathbf{c}_s))$$

Per Variable GRN

$$\tilde{\boldsymbol{\xi}}_t^{(j)} = \mathrm{GRN}_{\tilde{\boldsymbol{\xi}}^{(j)}}(\boldsymbol{\xi}_t^{(j)}),$$

Variable Selection Output

$$\tilde{\boldsymbol{\xi}}_t = \sum_{j=1}^{m_\chi} v_{\chi_t}^{(j)} \tilde{\boldsymbol{\xi}}_t^{(j)},$$

Static Covariate Encoders

Static metadata is encoded, by separate GRNs, in four distinct context vectors:

- $\mathbf{c}_s$ for temporal variable selection,
- $\mathbf{c}_e$ for enriching temporal features with static information,
- $\mathbf{c}_c$ for initializing the LSTM cell state,
- $\mathbf{c}_h$ for initializing the LSTM hidden state.

Interpretable Multi-Head Attention

Interpretable Multi-Head Attention

$$\text{InterpretableMultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \tilde{\mathbf{H}}\mathbf{W}_H,$$

$$\begin{aligned}\tilde{\mathbf{H}} &= \left\{ \frac{1}{H} \sum_{h=1}^{m_H} \mathrm{A}(\mathbf{Q}\mathbf{W}_Q^{(h)}, \mathbf{K}\mathbf{W}_K^{(h)}) \right\} \mathbf{V}\mathbf{W}_V \\ &= \frac{1}{H} \sum_{h=1}^{m_H} \text{Attention}(\mathbf{Q}\mathbf{W}_Q^{(h)}, \mathbf{K}\mathbf{W}_K^{(h)}, \mathbf{V}\mathbf{W}_V)\end{aligned}$$

TFT modifies multi-head attention because when different values are used in each head, attention weights alone might not be indicative of a particular feature's importance.
Positional encoding is not needed — the LSTM implicitly does it.

Temporal Fusion Decoder

The LSTM receives $\tilde{\boldsymbol{\xi}}_{t-k:t}$ as inputs to the encoder, and $\tilde{\boldsymbol{\xi}}_{t+1:t+\tau_{\max}}$ to the decoder.

The output of this process is $\boldsymbol{\phi}(t, n),\ n = -k, \ldots, \tau_{\max}$.

A gated skip connection is employed over the LSTM part:

$$\tilde{\boldsymbol{\phi}}(t, n) = \mathrm{LayerNorm}(\tilde{\boldsymbol{\xi}}_{t+n} + \mathrm{GLU}_{\tilde{\phi}}(\boldsymbol{\phi}(t, n))), \quad n = -k, \ldots, \tau_{\max}$$

Static Enrichment Layer

Static covariate information is injected via a static enrichment layer:

$$\boldsymbol{\theta}(t,n) = \mathrm{GRN}\left(\tilde{\boldsymbol{\phi}}(t,n), \mathbf{c}_e\right), \quad n = -k, \dots, \tau_{\max}$$

Temporal Self Attention Layer

All statically-enriched temporal features are grouped into a matrix $\mathbf{\Theta}(t) = [\boldsymbol{\theta}(t, -k), \ldots, \boldsymbol{\theta}(t, \tau)]^T$ and interpretable multi-head attention is applied at each forecast time:

$$\begin{aligned}\mathbf{B}(t) &= \mathrm{InterpretableMultiHead}(\mathbf{\Theta}(t), \mathbf{\Theta}(t), \mathbf{\Theta}(t)) \\ &= [\boldsymbol{\beta}(t, -k), \ldots, \boldsymbol{\beta}(t, \tau_{\max})]\end{aligned}$$

A gated skip connection is also added here:

$$\boldsymbol{\delta}(t, n) = \mathrm{LayerNorm}(\boldsymbol{\theta}(t, n) + \mathrm{GLU}_{\delta}(\boldsymbol{\beta}(t, n))), \quad n = -k, \ldots, \tau_{\max}$$

Position-wise Feed-forward Layer

Additional non-linear processing is applied to the output of the self-attention layer:

$$\boldsymbol{\psi}(t,n) = \mathrm{GRN}(\boldsymbol{\delta}(t,n)), \quad n = -k, \ldots, \tau_{\max}$$

A gated skip connection over the entire transformer block is also applied:

$$\tilde{\boldsymbol{\psi}}(t,n) = \mathrm{LayerNorm}\left(\tilde{\boldsymbol{\phi}}(t,n) + \mathrm{GLU}_{\tilde{\boldsymbol{\psi}}}(\boldsymbol{\psi}(t,n))\right), \quad n = -k, \ldots, \tau_{\max}$$

Quantile Ouputs

$$\hat{y}_i(q,t,\tau) = \mathbf{W}_q \tilde{\boldsymbol{\psi}}(t,\tau) + b_q$$

e.g. $q = 0.1, 0.5, 0.9$

Loss Function

Quantile Loss

$$\mathcal{L}(\Omega, \mathbf{W}) = \sum_{y_t \in \Omega} \sum_{q \in \mathcal{Q}} \sum_{\tau=1}^{\tau_{\max}} \frac{\mathrm{QL}(y_t,\, \hat{y}(1, t-\tau, \tau),\, q)}{M \tau_{\max}}$$

$$\mathrm{QL}(y_t,\, \hat{y},\, q) = q(y - \hat{y})_+ + (1-q)(\hat{y} - y)_+$$

where Ω is the domain of the training data containing M samples, $\mathbf{W}$ represents the weights of the TFT, $\mathcal{Q}$ is the set of output quantiles (a typical choice is $\mathcal{Q} = \{0.1, 0.5, 0.9\}$), and $(\cdot)_+ = \max(0, \cdot)$

Datasets

Datasets

Electricity Load Diagrams Dataset

Original Dataset

- From the UCI ML Repository [8]
- Electricity consumption of 370 clients
- Data from 2011 to 2014
- 15 min frequency
- Some clients' history starts after the dataset's minimum date

Usage in the Thesis

- The first 50 time series were selected (all clients have data)
- Data from January 2, 2014, to September 1, 2014.
- Resampled frequency to 1 hour (averaged).

Time Series Heatmap

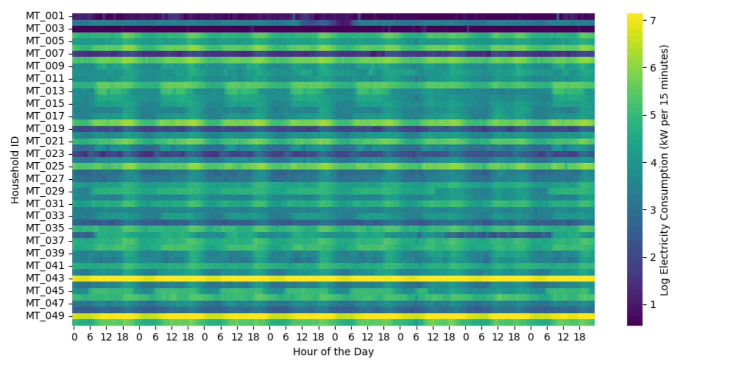
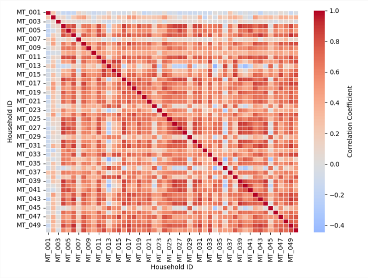
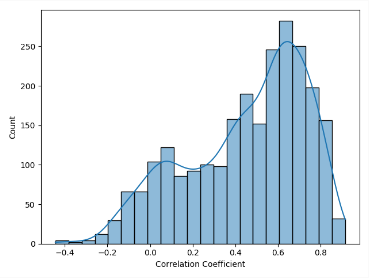


Figure 8: Heatmap of Hourly Electricity Consumption. Data from Monday 2014–02–03 to Sunday 2014–02–09. Seasonal patterns, as well as varying varying scales, are evident.

Correlation



(a) Correlation Heatmap



(b) Correlation Histogram (values of 1.0 excluded)

Figure 9: Electricity Load Diagrams Correlation. Moderate to high positive correlations, with seasonality being a common underlying factor.

Median

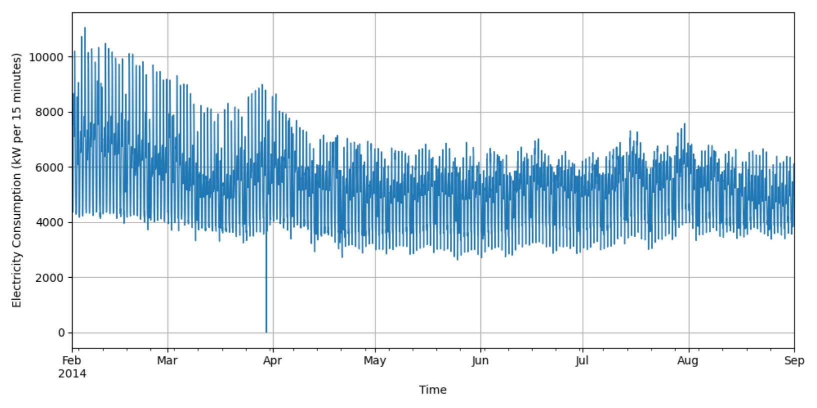
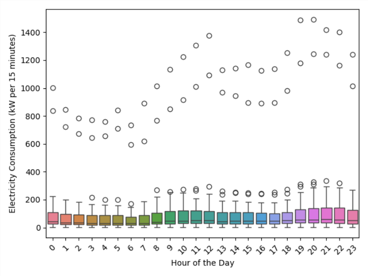
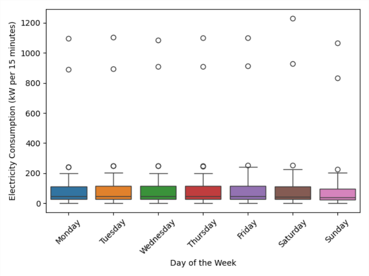


Figure 10: Median Electricity Consumption. The consumption is 0 on the last Sunday of March, due to daylight savings.

Seasonal Boxplots



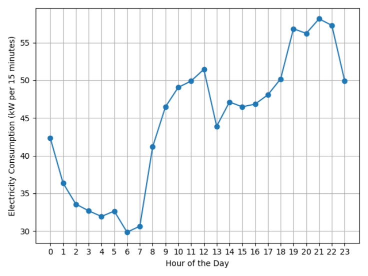
(a) Hourly Boxplot



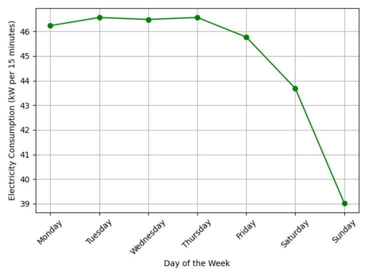
(b) Weekly Boxplot

Figure 11: Boxplots of hourly mean values of each time series.

Seasonal Medians



(a) Median Hourly Electricity Consumption



(b) Median Weekly Electricity Consumption

Figure 12: Median of mean values of each time series.

Datasets

PeMS-SF Dataset

Original Dataset

- Stands for “Performance Measurement System — San Francisco”.
- From the UCI ML Repository [9].
- Occupancy rates from 963 lanes in the Sans Francisco Bay area in California.
- Data from January 1, 2008 to March 30, 2009.
- 10 min frequency.
- Missing values at 4 entire days (holidays and sensor malfunctioning).
- Dataset in form for classification tasks.

Usage in the Thesis

- Data restructured for forecasting, through concatenations, inverse permutations, and inferring timestamps and missing days.
- The first 50 time series were selected (all clients have data).
- Data from January 1, 2008, to June 25, 2008.
- Resampled frequency to 1 hour (averaged).
- Missing value replaced with 0 for DL models, left as is for Prophet.

Time Series Heatmap

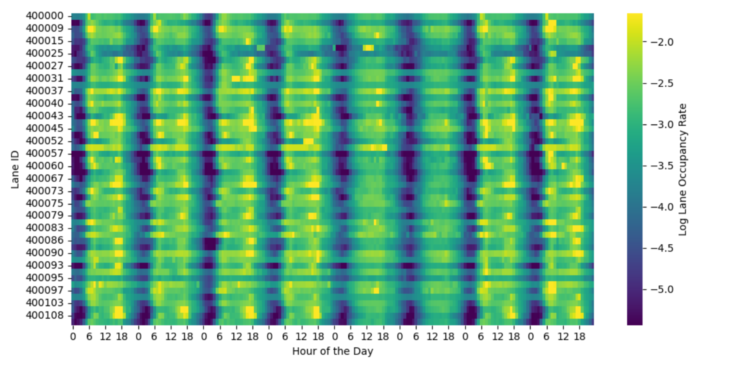
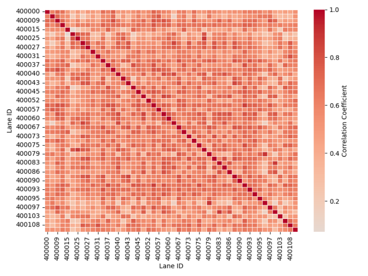
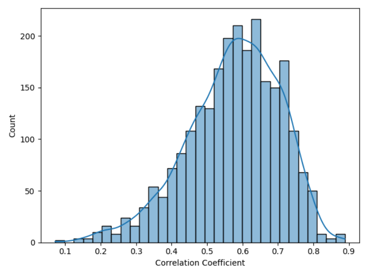


Figure 13: Heatmap of Hourly Lane Occupancy Rates. Data from Monday 2008–01–07 to Sunday 2008–01–14.

Correlation



(a) Correlation Heatmap



(b) Correlation Histogram (values of 1.0 excluded)

Figure 14: PeMS-SF Correlation. Moderate to high positive correlations, with seasonality being a common underlying factor.

Median

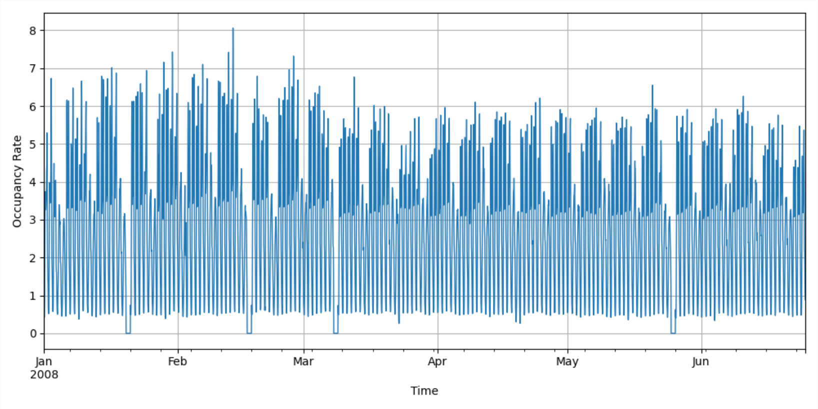
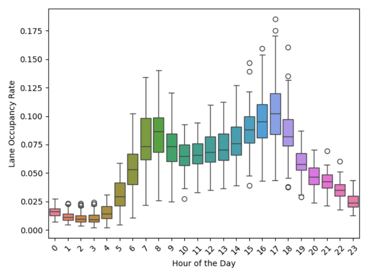
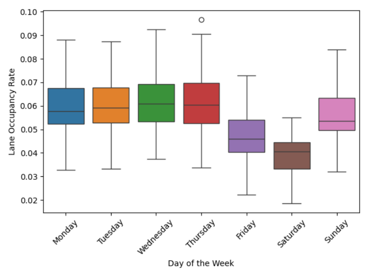


Figure 15: Median Occupancy Rates. Missing days are visible.

Seasonal Boxplots



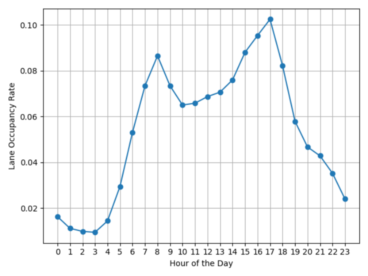
(a) Hourly Boxplot



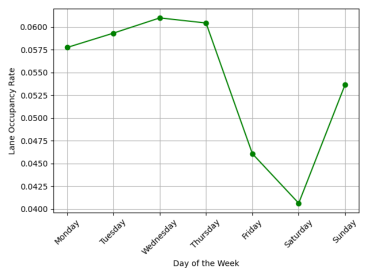
(b) Weekly Boxplot

Figure 16: Boxplots of hourly mean values of each time series.

Seasonal Medians



(a) Median Hourly Occupancy Rates



(b) Median Weekly Occupancy Rates

Figure 17: Median of mean values of each time series.

Methodology

Methodology

Overview

Goal

- The goal is to accurately forecast the last day of each dataset.
- Prophet and DL methods are handled differently.

Depiction

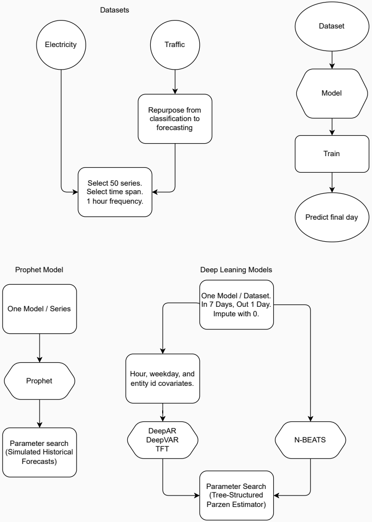


Figure 18: Overview of the methodology.

Prophet Methodology

- One model for each time series.
- Separate hyperparameter search for each model.
- Validation with Rolling Historical Forecasts with half Horizon strides.

Rolling Historical Forecasts

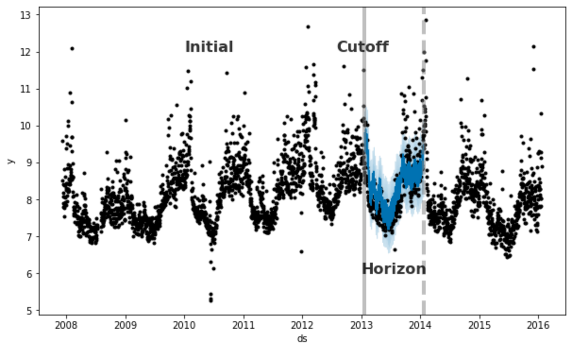


Figure 19: Cross-validation for tuning Prophet hyperparameters.

DL Models Methodology

- A single model for each dataset.
- Input 7 days, output 1 day.
- Time series name, hour, weekday as covariates (except N-BEATS).
- Validation set using the last 3 weeks (excluding the final day) — approximately 10% of the data.
- Validation set used for hyperparameter tuning and early stopping.
- Hyperparameter tuning with a mix of
    - Optuna's [10] *Tree-Structured Parzen Estimator* (TPE) [11]
    - Pytorch Lightning's `Tuner` that uses *Cyclical Learning Rates* (CLR) [12]

Methodology

Evaluation Metrics

Mean Squared Error (MSE)

Mean Squared Error (MSE)

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Gives a higher weight to larger errors due to the squaring part of the formula, making it particularly useful when large errors are undesirable.

Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE)

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}$$

The square root of MSE.

Mean Absolute Error (MAE)

Mean Absolute Error (MAE)

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MAE is particularly useful for understanding how big the error will be on average.

Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE)

$$\frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

MAPE is easy to interpret but can be undefined or infinite for values of $y_i = 0$ and can disproportionately penalize overestimates — for example, if $\hat{y} = 2$, then $\mathrm{ape} = 100\%$ when $y = 1$, while $\mathrm{ape} = 33.\overline{3}\%$ when $y = 3$.

Symmetric Mean Absolute Percentage Error (SMAPE)

Symmetric Mean Absolute Percentage Error (SMAPE)

$$\frac{100\%}{n}\sum_{i=1}^{n}\frac{2|y_i-\hat{y}_i|}{|y_i|+|\hat{y}_i|}$$

SMAPE adjusts MAPE to be symmetric, ensuring that overestimates and underestimates are penalized equally. It is bounded between 0 and 200%.

Mean Absolute Scaled Error (MASE)

Mean Absolute Scaled Error (MASE)

$$\frac{\text{MAE}}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|}$$

MASE measures the accuracy of forecasts relative to a naïve baseline prediction, scaling the MAE by the average absolute difference between consecutive observations in the training dataset. MASE is particularly useful because it is scale-independent and can be used to compare forecast performance across different datasets.

Methodology

Hyperparameter Spaces

Prophet

- $\tau \in \{0.001, 0.01, 0.1, 0.5\}$ for the change point prior
- Daily seasonality of Fourier order 4
- Weekly seasonality of Fourier order 3
- No smoothing priors for seasonalities — regularization delegated to the Fourier order.
- Multiplicative seasonality for Electricity, additive for Traffic.
- Linear growth for Electricity, Logistic growth saturated at 0 and 1 for Traffic.
- The first 80% of the training data was used to find change points, due to relatively stable trends.
- No holiday component — unknown country for electricity, missing from traffic.
- Validation first cutoff day 205 for electricity, day 169 for traffic. Half horizon steps.

DL Models

- dropout $\in \{0.1, 0.2, 0.3, 0.4, 0.5\}$
- Gradient norm clipping in the logarithmic space of $[10^{-1}, 10^{2}]$
- Batch size 128
- Early stopping on the validation loss with patience 5 and tolerance 10^{-4}.
- Learning rate: CLR in $[10^{-8}, 1]$ if successful and $< 10^{-2}$, else TPE suggestion in the logarithmic space of $[10^{-4}, 10^{-2}]$.

- Interpretable architecture
- Trend polynomial degree: 3
- 3 blocks per stack
- 3 FC layers per block
- Trend stack FC layer size: 64, 256
- Seasonality stack FC layer size: 512, 2048

- 2 LSTM layers
- LSTM hidden size: 80, 160, 320
- Normal distribution for electricity, Beta distribution for traffic
- Monte Carlo samples for prediction: 100
- Embedding size:
  $\min\{\mathrm{round}(1.6n^{0.56}),\, 100\} \quad \text{if} \quad n > 2 \quad \text{else} \quad 1$

DeepVAR

- Rank of the non-diagonal part of the covariance matrix: 10
- LSTM configuration, embedding size, Monte Carlo samples same as DeepAR.

- $\text{quantiles} = (0.1, 0.5, 0.9)$
- $d_{\text{model}} \in \{80, 160, 320\}$
- 1 LSTM layer
- 4 attention heads

Source Code

`github.com/k-papadakis/dsml-thesis-code`

Install with `pip install --upgrade git+https://github.com/k-papadakis/dsml-thesis-code.git`

For usage info, run `thesis -h`

Results

Performance on Electricity

	MASE	SMAPE	RMSE	MAE
Prophet	1.81	0.231	16.8	14.2
N-BEATS	0.84	0.104	18.3	7.6
DeepAR	0.64	0.078	14.0	5.6
DeepVAR	0.66	0.080	15.1	5.9
TFT	0.86	0.103	20.6	7.9

Table 1: Performance on Electricity. MAPE was omitted due to the presence of zeros in the true values.

Performance of Traffic

	MASE	SMAPE	RMSE	MAE	MAPE
Prophet	0.943	0.290	0.01547	0.01149	0.345
N-BEATS	0.436	0.101	0.01195	0.00529	0.133
DeepAR	0.438	0.087	0.01083	0.00494	0.106
DeepVAR	0.542	0.161	0.01073	0.00618	0.229
TFT	0.408	0.089	0.01281	0.00495	0.101

Table 2: Performance on Traffic

MASE on Electricity

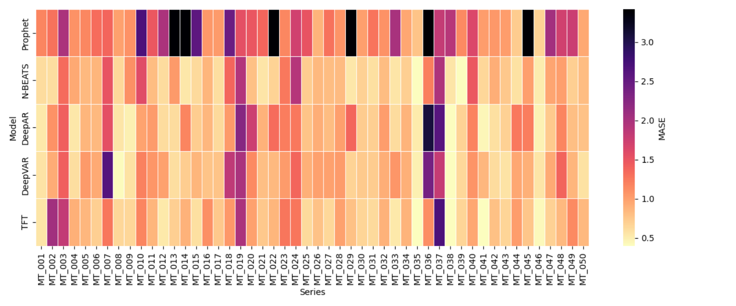


Figure 20: MASE on Electricity

MASE on Traffic

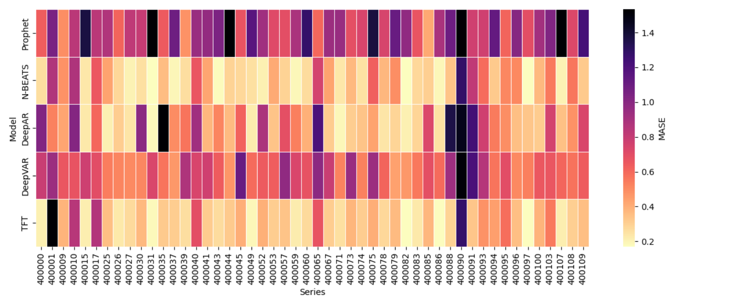


Figure 21: MASE on Traffic

Remarks

To represent results from all scales, the series with the lowest, median, and highest mean values from each dataset are shown in what follows.

The main focus is on MASE, due to its interpretability, scale independence, robustness to outliers, handling of zeros, and unbiasedness towards over/under-predicting.

Results

Prophet

Remarks

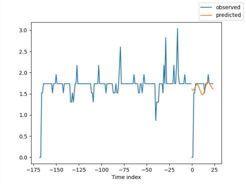
Prophet consistently underperforms compared to the other models across all metrics.

This highlights Prophet's lower learning capacity compared to DL models.

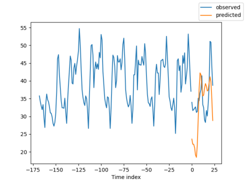
Manual adjustments might improve performance, but this approach lacks scalability.

Uncertainty intervals do generally cover the true values appropriately, which is useful for further decision-making.

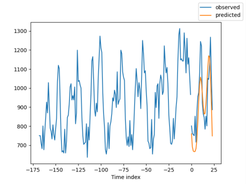
Electricity Prediction



(a) MT003



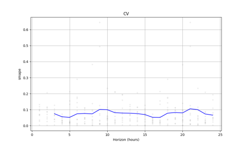
(b) MT015



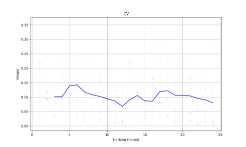
(c) MT043

Figure 22: Predictions of Prophet on the Electricity dataset.

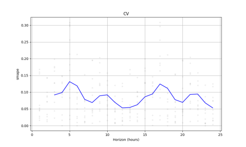
Electricity CV



(a) MT003

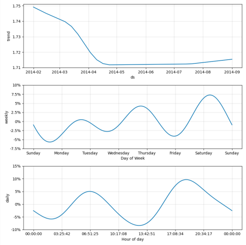


(b) MT015

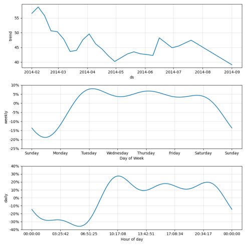


(c) MT043

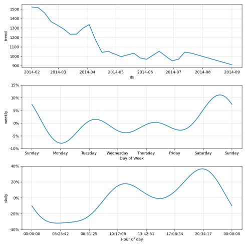
Figure 23: Cross-validation SMAPE of Prophet on the Electricity dataset. The grey dots represent SMAPEs on that specific time index, one from each simulated historical forecast, and the blue line represents the average SMAPE on that time index.



(a) MT003



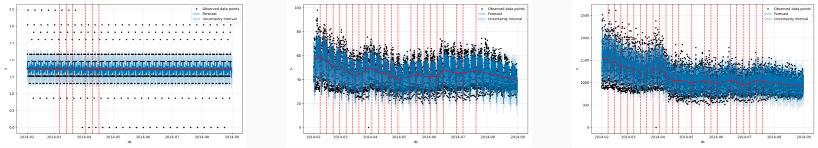
(b) MT015



(c) MT043

Figure 24: Trend and seasonality components of Prophet on the Electricity dataset.

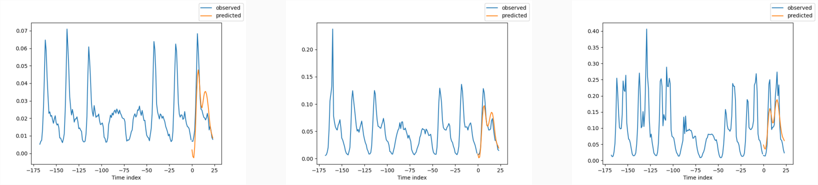
Electricity Models



(a) MT003 (b) MT015 (c) MT043

Figure 25: Model overview of Prophet on the Electricity dataset. Forecasts in blue lines, uncertainty in light blue, true values in black dots, and important change points in dashed vertical red lines.

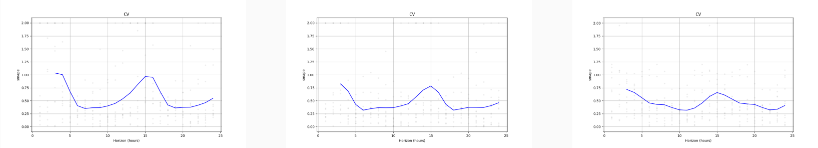
Traffic Prediction



(a) 400015 (b) 400071 (c) 400041

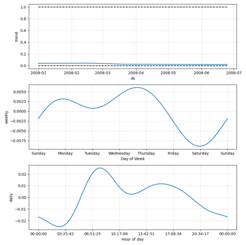
Figure 26: Predictions of Prophet on the Traffic dataset.

Traffic CV

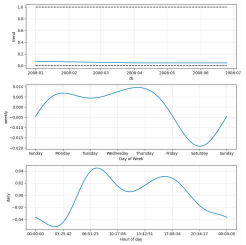


(a) 400015 (b) 400071 (c) 400041

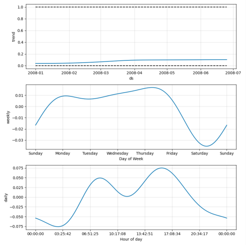
Figure 27: Cross-validation SMAPE of Prophet on the Traffic dataset. The grey dots represent SMAPEs on that specific time index, one from each simulated historical forecast, and the blue line represents the average SMAPE on that time index.



(a) 400015



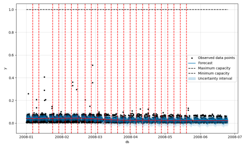
(b) 400071



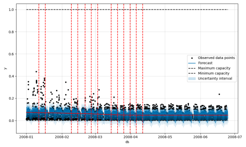
(c) 400041

Figure 28: Trend and seasonality components of Prophet on the Traffic dataset.

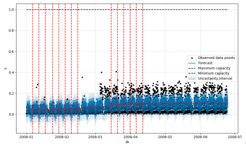
Traffic Models



(a) 400015



(b) 400071



(c) 400041

Figure 29: Model overview of Prophet on the Traffic dataset.

Results

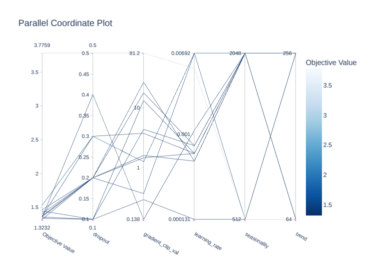
N-BEATS

Best Params

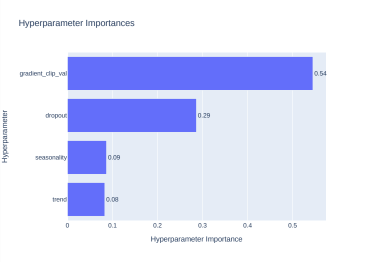
	Clip	Dropout	LR	Trend	Seas
electricity	1.60	0.2	0.0005	256	2048
traffic	1.25	0.3	0.0008	256	2048

Table 3: Optuna-found hyperparameters for N-BEATS.

Electricity Optuna



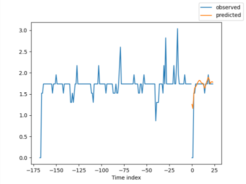
(a) Parallel Coordinates



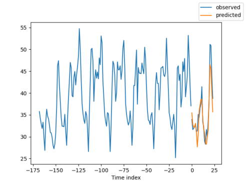
(b) Hyperparameter Importances

Figure 30: Optuna results for the N-BEATS model on the Electricity dataset.

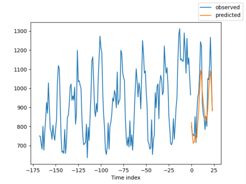
Electricity Prediction



(a) MT003



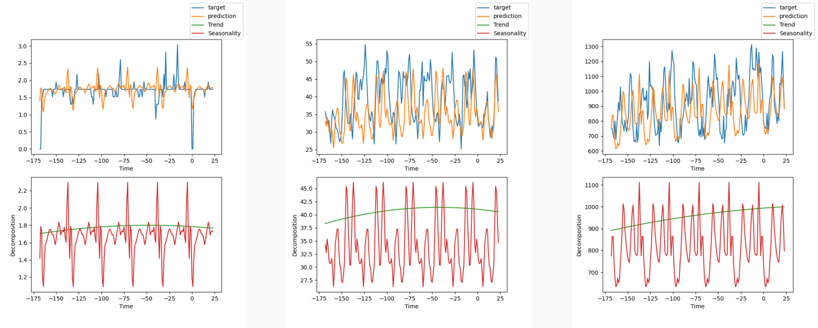
(b) MT015



(c) MT043

Figure 31: Predictions of N-BEATS on the Electricity dataset. The model captures the pattern, but not the magnitudes. This is possibly due to the varying scales challenge.

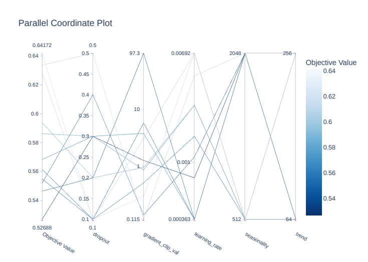
Electricity Interpretation



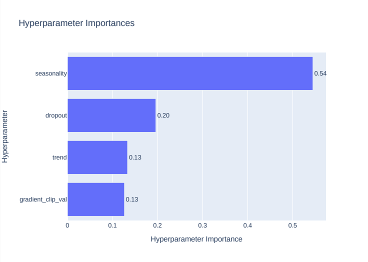
(a) MT003 (b) MT015 (c) MT043

Figure 32: Trend-Seasonality interpretation of N-BEATS on the Electricity dataset. The model's learned seasonality aligns with the prediction range but diverges from the conditioning range.

Traffic Optuna



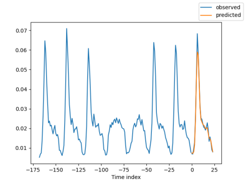
(a) Parallel Coordinates



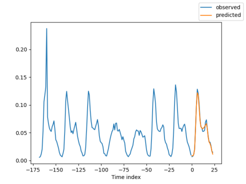
(b) Hyperparameter Importances

Figure 33: Optuna results for the N-BEATS model on the Traffic dataset.

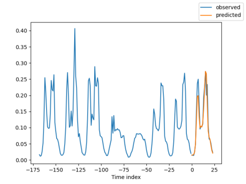
Traffic Prediction



(a) 400015



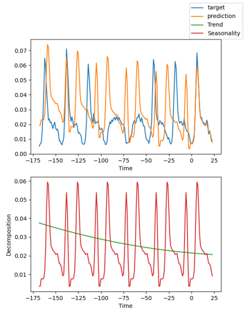
(b) 400071



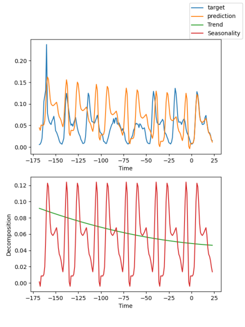
(c) 400041

Figure 34: Predictions of N-BEATS on the Traffic dataset. The model performs well here, possibly due to the traffic dataset not having varying scales.

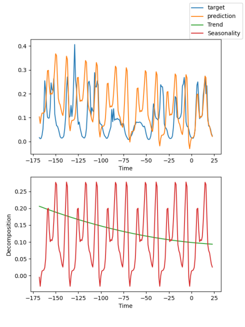
Traffic Interpretation



(a) 400015



(b) 400071



(c) 400041

Figure 35: Trend-Seasonality interpretation of N-BEATS on the Traffic dataset.

Results

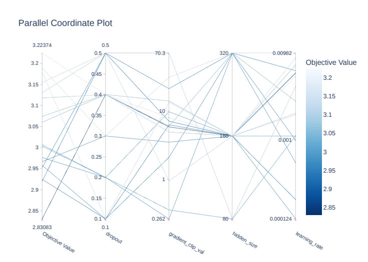
DeepAR

Best Params

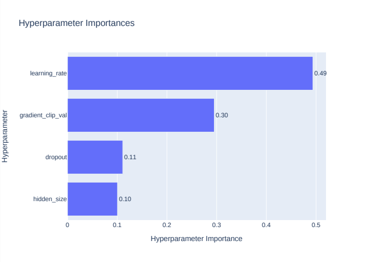
	LSTM	Clip	Dropout	LR
electricity	160	5.83	0.4	0.006
traffic	160	44.44	0.4	0.004

Table 4: Optuna-found hyperparameters for DeepAR.

Electricity Optuna



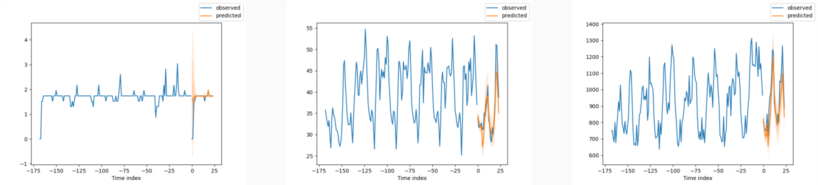
(a) Parallel Coordinates



(b) Hyperparameter Importances

Figure 36: Optuna results for the DeepAR model on the Electricity dataset.

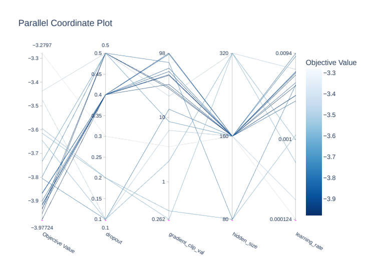
Electricity Prediction



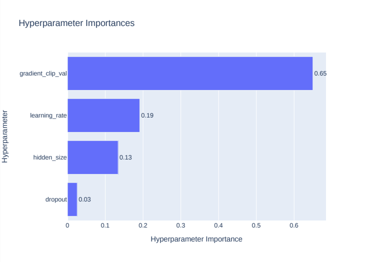
(a) MT003 (b) MT015 (c) MT043

Figure 37: Probabilistic predictions of DeepAR on the Electricity dataset. Monte Carlo samples: 100, Quantiles: 0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98. The model exhibits superior performance, possibly due to its varying scale handling. The prediction intervals appropriately encompass the target values.

Traffic Optuna



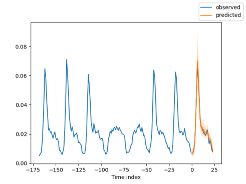
(a) Parallel Coordinates



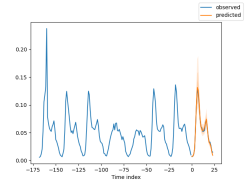
(b) Hyperparameter Importances

Figure 38: Optuna results for the DeepAR model on the Traffic dataset.

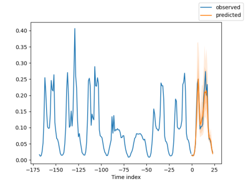
Traffic Prediction



(a) 400015



(b) 400071



(c) 400041

Figure 39: Probabilistic predictions of DeepAR on the Traffic dataset. Monte Carlo samples: 100, Quantiles: 0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98. Wide prediction intervals due to variability in the conditioning range.

Results

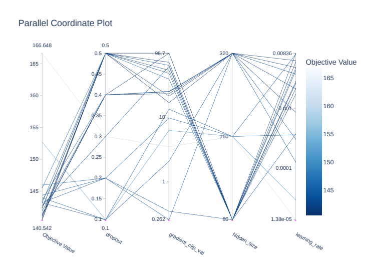
DeepVAR

Best Params

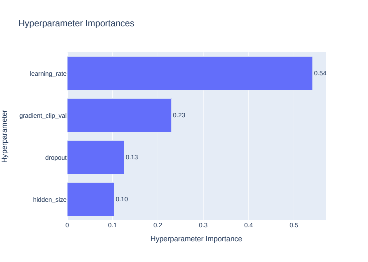
	LSTM	Clip	Dropout	LR
electricity	160	1.23	0.3	0.001
traffic	80	96.73	0.5	0.006

Table 5: Optuna-found hyperparameters for DeepVAR. They were replaced with DeepAR's hyperparameters for the final results.

Electricity Optuna



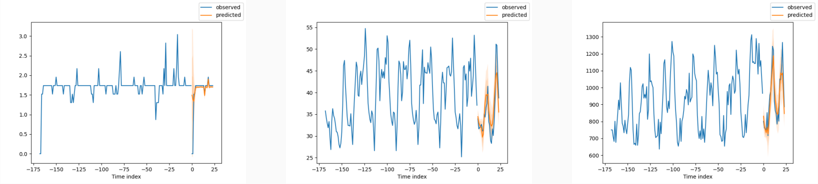
(a) Parallel Coordinates



(b) Hyperparameter Importances

Figure 40: Optuna results for the DeepVAR model on the Electricity dataset. They were replaced with DeepAR's hyperparameters for the final results.

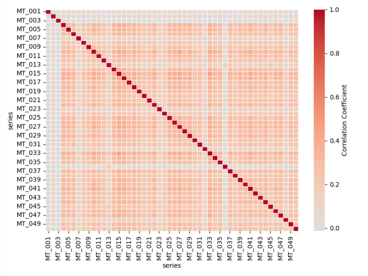
Electricity Prediction



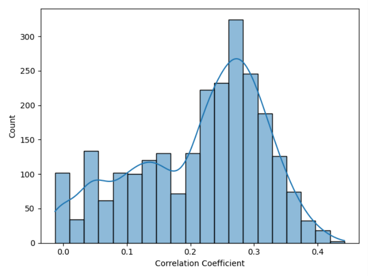
(a) MT003 (b) MT015 (c) MT043

Figure 41: Probabilistic predictions of DeepVAR on the Electricity dataset. Monte Carlo samples: 100, Quantiles: 0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98. Predictions are similar to DeepAR.

Electricity Correlation

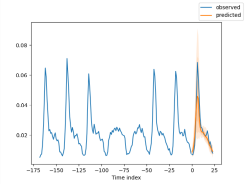


(a) Correlation Heatmap

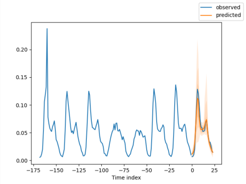


(b) Correlation Histogram (1.0 excluded)

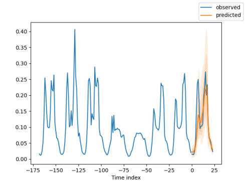
Figure 42: Average Correlation of the DeepVAR model on the Electricity dataset. Result from the average of the covariance matrices across Monte Carlo samples and forecast indices. The learned correlation matrix resembles the original, but not entirely, possibly due to the low-rank-plus-diagonal parametrization.



(a) 400015



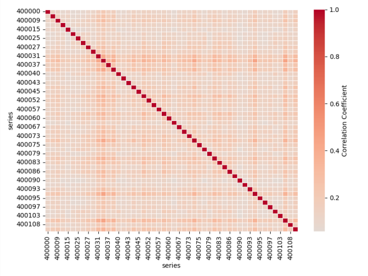
(b) 400071



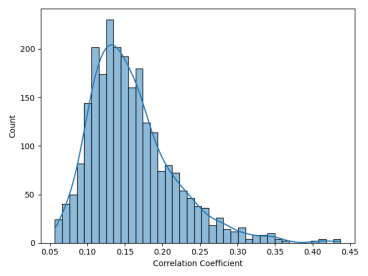
(c) 400041

Figure 43: Probabilistic predictions of DeepVAR on the Traffic dataset. Monte Carlo samples: 100, Quantiles: 0.02, 0.1, 0.25, 0.5, 0.75, 0.9, 0.98. The model underperforms here — modeling with a Gaussian copula might not be appropriate.

Traffic Correlation



(a) Correlation Heatmap



(b) Correlation Histogram (1.0 excluded)

Figure 44: Average Correlation of the DeepVAR model on the Traffic dataset. Result from the average of the covariance matrices across Monte Carlo samples and forecast indices. The estimate is quite different from the true value.

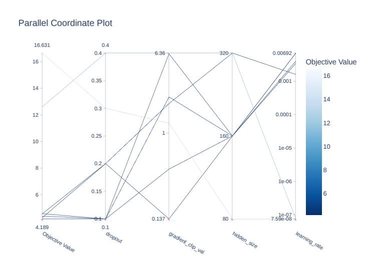
Results

Temporal Fusion Transformer

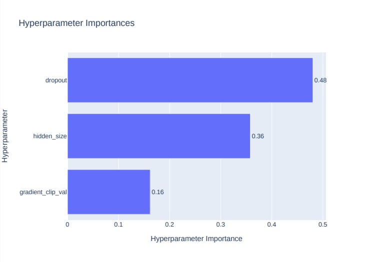
Best Params

	d_{model}	Clip	Dropout	LR
electricity	160	0.43	0.1	0.007
traffic	320	20.69	0.2	0.003

Table 6: Optuna-found hyperparameters for TFT.



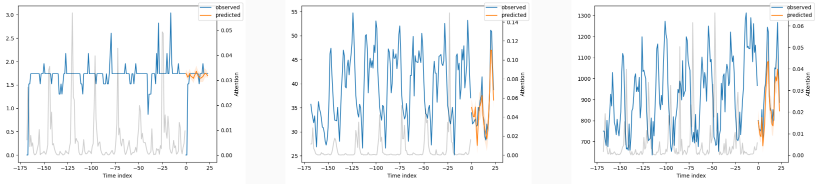
(a) Parallel Coordinates



(b) Hyperparameter Importances

Figure 45: Optuna results for the TFT model on the Electricity dataset.

Electricity Prediction



(a) MT003 (b) MT015 (c) MT043

Figure 46: Quantile predictions of TFT on the Electricity dataset. The attention weights for each time index are also shown. Predictions at quantiles 0.1, 0.5 and 0.9. Underperformance is possibly due to varying scales. Uninformative prediction intervals.

Electricity Attention

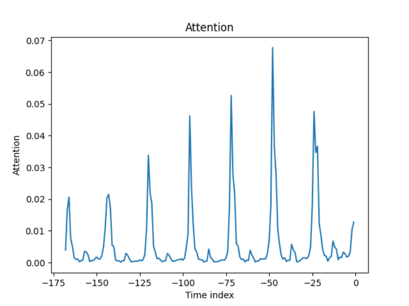
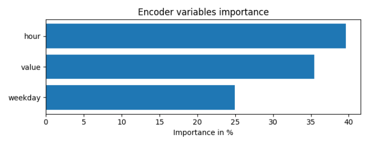
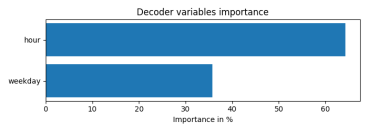


Figure 47: Average attention of TFT on the conditioning range on the Electricity dataset. Peaks at 24-hour multiples. A higher peak at two days in the past instead of one could be related to the model's underperformance on the dataset.

Electricity Variable Importance



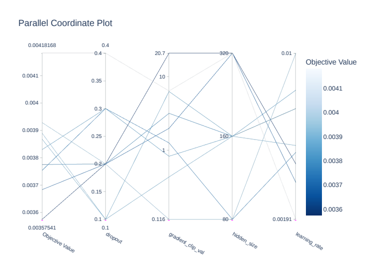
(a) Encoder Variables Importance



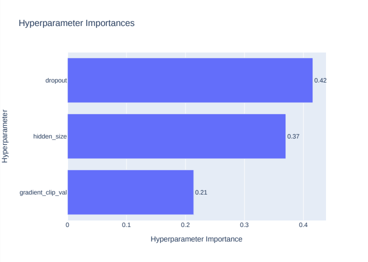
(b) Decoder Variables Importance

Figure 48: Variables importance of the encoder and decoder of the TFT model on the Electricity dataset. Computed using the softmax outputs of the Variable Selection Networks across time. The historical consumption not being the most important possibly relates to the lack of performance.

Traffic Optuna



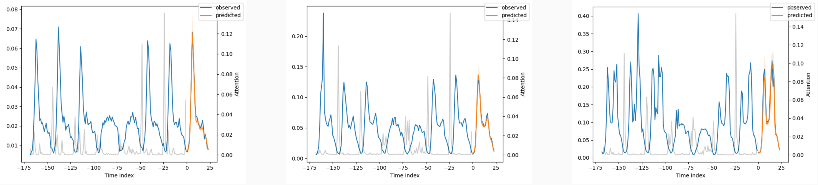
(a) Parallel Coordinates



(b) Hyperparameter Importances

Figure 49: Optuna results for the TFT model on the Traffic dataset.

Traffic Prediction



(a) 400015 (b) 400071 (c) 400041

Figure 50: Quantile predictions of TFT on the Traffic dataset. The attention weights for each time index are also shown. Predictions at quantiles 0.1, 0.5 and 0.9. Remarkable performance.

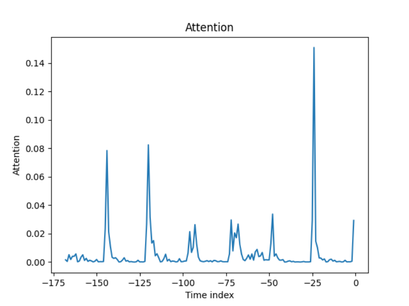
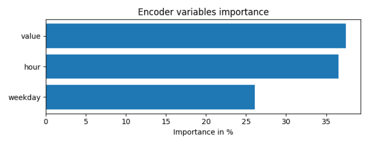
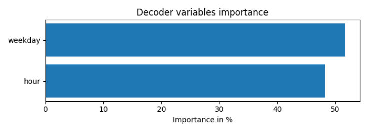


Figure 51: Average attention of TFT on the conditioning range on the Traffic dataset. Possibly using 1 day before as local context, and 5–6 days before as “weekly seasonality” since 7 days before is not available.

Traffic Variable Importance



(a) Encoder Variables Importance



(b) Decoder Variables Importance

Figure 52: Variables importance of the encoder and decoder of the TFT model on the Traffic dataset. Computed using the softmax outputs of the Variable Selection Networks across time. The historical occupancy rates are found to be the most important, unlike in the electricity case.

Conclusion

Deep Learning Supremacy

Deep learning models consistently outperformed Prophet on both datasets. This underscores the power of DL in forecasting when the volume of the data allows it.

Dataset Nuances

Model performance varied significantly based on the dataset. The complexity of the patterns and the distribution of the scales play an important role.

Summary of Findings ii

Probabilistic Advantage

Confidence intervals can offer valuable insight and heavily impact decision-making, but they should be used with caution since they can often be misleading.

The Importance of Manual Tuning

In certain cases, manual hyperparameter tuning outperforms fully automated approaches for deep learning models. This underscores the value of domain knowledge and experimentation within the optimization process.

Interpretability Considerations

While N-BEATS offers a theoretically interpretable architecture, the results suggest that practical interpretability benefits might be limited. TFT, on the other hand, provided valuable insights through its attention mechanism and variable selection. Finally, the interpretability of the Prophet model comes with model simplicity.

Future Work i

Hybrid Approaches

Explore combining the strengths of Prophet's interpretability with the power of DL models to create hybrid forecasting solutions.

Additional Datasets

Evaluate these models on datasets from other domains (e.g., finance, weather) to investigate their generalizability and performance in different contexts.

Future Work ii

Expanding Scope

Incorporate external factors like weather conditions or special events into models to enhance forecasting accuracy and inform proactive decision-making.

Real-World Deployment

Investigate the challenges of deploying forecasting models in real-world operational settings, considering aspects like model maintenance, scalability, and integration with existing systems.

Limitations of Interpretability

Investigate the factors hindering interpretability in theoretically interpretable architectures like N-BEATS.

Power of Attention Mechanisms

Further exploration of attention-based models like TFT could yield insights into the key drivers and dependencies within complex time-series data.

Thank you!

Bibliography

References

[1] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*, 2nd. Melbourne, Australia: OTexts, 2018, Accessed on 2023-01-01. [Online]. Available: `https://otexts.com/fpp2/`.

[2] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: The forecast package for r," *Journal of Statistical Software*, vol. 27, no. 3, pp. 1–22, 2008. DOI: `10.18637/jss.v027.i03`. [Online]. Available: `https://www.jstatsoft.org/index.php/jss/article/view/v027i03`.

[3] S. J. Taylor and B. Letham, “Forecasting at scale,” *PeerJ Preprints*, vol. 5, e3190v2, 2017. DOI: 10.7287/peerj.preprints.3190v2.

[4] V. Flunkert, D. Salinas, and J. Gasthaus, “Deepar: Probabilistic forecasting with autoregressive recurrent networks,” *CoRR*, vol. abs/1704.04110, 2017. arXiv: 1704.04110. [Online]. Available: http://arxiv.org/abs/1704.04110.

[5] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus, *High-dimensional multivariate forecasting with low-rank gaussian copula processes*, 2019. arXiv: 1910.03002 [cs.LG].

[6] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, *N-beats: Neural basis expansion analysis for interpretable time series forecasting*, 2020. arXiv: `1905.10437 [cs.LG]`.

[7] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, *Temporal fusion transformers for interpretable multi-horizon time series forecasting*, 2020. arXiv: `1912.09363 [stat.ML]`.

[8] A. Trindade, *ElectricityLoadDiagrams20112014*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C58C86, 2015.

[9] M. Cuturi, *PEMS-SF*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C52G70, 2011.

[10] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[11] S. Watanabe, *Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance*, 2023. arXiv: `2304.11127 [cs.LG]`.

[12] L. N. Smith, *Cyclical learning rates for training neural networks*, 2017. arXiv: `1506.01186 [cs.CV]`.