# E-PROTRAN: Efficient Probabilistic Transformers for Forecasting

Batuhan Koyuncu [*1] Tim Nico Bauerschmidt [*1] Isabel Valera [1]

## Abstract

Time series forecasting involves predicting future data points based on historical patterns and is critical for applications in fields such as health-care, financial markets, and weather forecasting, where scalability and efficiency, particularly in training and inference times, are paramount. Transformers, known for their ability to handle long-range dependencies in sequential data, have shown promise in time series analysis. However, the complexity of transformer models can lead to overparameterization, extended training times, and scalability challenges, which can become even more problematic if the assumptions of the underlying generative model are overly complicated. In this paper, we introduce E-PROTRAN by re-designing a state-of-the-art transformer for probabilistic time series forecasting. We empirically demonstrate that E-PROTRAN maintains high performance while significantly enhancing efficiency without necessarily reconstructing the conditioned history. Our model incorporates simplified attention layers and design adjustments that reduce computational overhead without compromising accuracy, offering a more efficient and scalable solution for time series forecasting.

## 1. Introduction

A time series consist of data points with values that can be modeled as a function of time – in other words, data points are ordered in time. Time series analysis has been used for understanding, classification, or forecasting of time series data. In particular, time series forecasting has important applications in fields such as health-care (Ghassemi et al., 2015), financial markets (Sonkavde et al., 2023), weather forecasting (Zaytar & Amrani, 2016), and many more.

[*]Equal contribution [1]Saarland University, Saarbrücken, Germany. Correspondence to: Batuhan Koyuncu <koyuncu@cs.uni-saarland.de>.

*Accepted by the Structured Probabilistic Inference & Generative Modeling workshop* of ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

Transformers (Vaswani et al., 2017) are promising architectures for natural language processing (Radford et al., 2018; Devlin et al., 2019), speech processing (Ao et al., 2021), and computer vision (Carion et al., 2020; Dosovitskiy et al., 2020). One of the main strengths of transformers is their ability to handle long-range dependencies and relationships in sequential data, making them highly preferable for time series analysis and modeling. Moreover, Elsayed et al. (2021) showed that transformer-based models are one of the few architectures that can outperform traditional ML models for time series forecasting tasks.

A transformer learns weights on the fly, i.e., weights are conditioned on the data in each layer using an attention mechanism. This brings greater expressivity to the model; however, it also leads to more complex models as the number of attention blocks increases. In the end, the higher complexity may create overparameterized models, deteriorate training times, reduce scalability, and cause converging problems.

In this paper, we empirically demonstrate that our proposed model, E-PROTRAN, a simplified version of the state-of-the-art attention-based architecture Probabilistic Transformer (PROTRAN) (Tang & Matteson, 2021), offers strongly improved inference speed and scalability with a comparable performance.

## 2. PROTRAN: Probabilistic Transformer

### 2.1. Preliminaries

In this paper, we work with multivariate time series $x_{1:T} \in \mathbb{R}^{T\times d}$, where $x_t \in \mathbb{R}^d$ is a $d$-dimensional data vector at the discrete time step $t \in \mathbb{N}^+$. Given conditioning samples $x_{1:t_0}$ up to time step $t_0$, the goal is to predict the forecasting distribution $p(x_{t_0+1:T}|x_{1:t_0})$. Often, additional covariates that are known for each time step, such as the date, are included in the model. Following a latent variable approach, we decompose the full generative distribution

$$p_{\psi,\theta}(x_{1:T}|x_{1:t_0}, c_{1:T}) = \int p_\theta(x_{1:T}|z_{1:T})p_\psi(z_{1:T}|x_{1:t_0}, c_{1:T})dz_{1:T}. \quad (1)$$

Here, $z_{1:T}$ with $z_t \in \mathbb{R}^{d_z}$ is the corresponding latent time series, $p_\psi(z_{1:T}|x_{1:t_0}, c_{1:T})$ is a conditional prior distribution denoting the transition model with parameters $\psi$, and $p_\theta(x_{1:T}|z_{1:T})$ is a likelihood distribution denoting the emission model with paramters $\theta$.

Our objective is to learn both distributions, parameterizing them via neural networks to capture non-linear temporal patterns. As the full posterior is intractable, we approximate it with amortized variational inference (Rezende et al., 2014; Kingma & Welling, 2013).

In this work, we rely on transformers (Vaswani et al., 2017) to parameterize the conditional prior distribution $p_\psi$. Transformers use multihead attention which enables them to extract different representations by attending to various parts of the input. To this end, query $Q \in \mathbb{R}^{\ell_q \times d}$, key $K \in \mathbb{R}^{\ell_k \times d}$, and value $V \in \mathbb{R}^{\ell_k \times d}$ matrices are used, which typically get linearly transformed. Attention ($\mathcal{A}$) can then be denoted as

$$\begin{aligned}\mathcal{A}(Q,K,V) &= \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V\\ \text{MultiHead}(Q,K,V) &= \text{Concat}(\text{head}_1,\dots,\text{head}_h)W^O\\ \text{where head}_i &= \mathcal{A}(QW_i^Q, KW_i^K, VW_i^V).\end{aligned}$$

$W_i^Q, W_i^K, W_i^V$ correspond to the weight parameters of head $i$, and it is called self-attention if $Q = K = V$.

### 2.2. ProTran's Formulation

Our work is closely related to ProTran for multivariate time series forecasting (Tang & Matteson, 2021). ProTran has the transition and emission models[1]

$$p_\psi(z_{1:T}|x_{1:t_0}, c_{1:T}) = \prod_{t=1}^{T} p_\psi(z_t|z_{1:t-1}, x_{1:t_0}, c_{1:T}) \quad (2)$$

$$p_\theta(x_{1:T}|z_{1:T}) = \prod_{t=1}^{T} p_\theta(x_t|z_t). \quad (3)$$

A latent variable $z_t$ depends on the conditioning data $x_{1:t_0}$ and preceding latent variables $z_{1:t-1}$, while the observed variable $x_t$ depends only on $z_t$. Hence, $z_t$ needs to capture all information to generate $x_t$. The approximate posterior is conditioned on the whole data and factorizes as

$$q_\phi(z_{1:T}|x_{1:T}, c_{1:T}) = \prod_{t=1}^{T} q_\phi(z_t|z_{1:t-1}, x_{1:T}, c_{1:T}). \quad (4)$$

The corresponding evidence lower bound (ELBO) becomes

$$\begin{aligned}&\log(p_{\psi,\theta}(x_{1:T}|x_{1:t_0}, c_{1:T})) \geq \\ &\sum_{t=1}^{T} \left(\mathbb{E}_{z_t \sim q_\phi}\left[\log(p_\theta(x_t|z_t))\right] - D_{\text{KL}}(q_\phi \parallel p_\psi)\right),\end{aligned} \quad (5)$$

where $D_{\text{KL}}$ is the KL divergence, $q_\phi$ and $p_\psi$ denote $q_\phi(z_t|z_{1:t-1}, x_{1:T}, c_{1:T})$ and $p_\psi(z_t|z_{1:t-1}, x_{1:t_0}, c_{1:T})$, which are parameterized by the posterior and prior networks, respectively. The ELBO is tractable, and thus its maximization is used as objective for training the networks.

The forward pass of ProTran starts with projecting the inputs via an MLP, adding a fixed positional encoding, and applying layer normalization (LN) (Ba et al., 2016), i.e.,

$$h_t^{(0)} = \text{LN}(\text{MLP}([x_t, c_t]) + \text{Position}(t)). \quad (6)$$

For forecasting, $x_{t_0+1:T}$ are masked out by setting them to 0, while the covariates $c_{t_0+1:T}$ are known and given to the model. Then, for each layer $l \in [1,\dots,L]$, they compute

$$\begin{aligned}\tilde{w}_t^{(l)} &= \text{LN}(w_{t-1}^{(l)} + \mathcal{A}(w_{t-1}^{(l)}, w_{1:T}^{(l-1)}, w_{1:T}^{(l-1)})) \quad (\textbf{Layer Att.})\\ \bar{w}_t^{(l)} &= \text{LN}(\tilde{w}_t^{(l)} + \mathcal{A}(\tilde{w}_t^{(l)}, w_{1:t-1}^{(l)}, w_{1:t-1}^{(l)})) \quad (\textbf{Autoreg. Att.})\\ \hat{w}_t^{(l)} &= \text{LN}(\bar{w}_t^{(l)} + \mathcal{A}(\bar{w}_t^{(l)}, h_{1:t_0}^{(0)}, h_{1:t_0}^{(0)})) \quad (\textbf{Input Att.})\\ z_t^{(l)} &\sim \mathcal{N}\left(\text{MLP}(\hat{w}_t^{(l)}), \text{diag}(\text{SP}(\text{MLP}(\hat{w}_t^{(l)})))\right) \quad (7)\\ w_t^{(l)} &= \text{LN}(\hat{w}_t^{(l)} + \text{MLP}(z_t^{(l)}) + \text{Position}(t)) \quad (8)\end{aligned}$$

with assumptions on edge cases[2], $\mathcal{N}(\cdot,\cdot)$ is a Gaussian, SP denotes the Softplus function, and diag denotes a diagonal covariance matrix. These equations characterize the conditional prior $p_\psi$. The parameters of $q_\phi$ are mostly shared with $p_\psi$, except that Equation 7 is replaced with

$$k_t = \mathcal{A}(h_t^{(0)}, h_{1:T}^{(0)}, h_{1:T}^{(0)}); \qquad \hat{w}k_t^{(l)} = [\hat{w}_t^{(l)}, k_t] \quad (9)$$

$$z_t^{(l)} \sim \mathcal{N}\left(\text{MLP}(\hat{w}k_t^{(l)}), \text{diag}(\text{SP}(\text{MLP}(\hat{w}k_t^{(l)})))\right). \quad (10)$$

Finally, the emission model parameterizes the likelihood as $p_\theta(x_t|z_t^{(L)})$ = $\text{Laplace}(x_t|\text{MLP}(w_t^{(L)}), \beta)$, where throughout this paper we fix $\beta = 1$. With multiple layers of latent variables, the ELBO in Equation 5 has a sum over the layers for the KL divergence. The underlying assumption is that prior and approximate posterior factorize over the time steps and layers. We provide the exact details in Appendix A.

### 2.3. Weaknesses of ProTran

We argue that ProTran is overly complex while introducing no benefits in terms of predictive power. In the following section, we discuss our arguments in detail.

**Autoregressive Attention (I)** removes the computational benefit that transformers possess over RNNs by leveraging parallel computation. As Vaswani et al. (2017) argue, computing one layer of attention incurs a complexity of $\mathcal{O}(T^2 d)$,

[1]Here $p_\psi(z_1|z_{1:0}, x_{1:t_0}, c_{1:T}) = p_\psi(z_1|x_{1:t_0}, c_{1:T})$.

[2]Layer attention is omitted at $l = 1$, autoregressive attention is omitted at $t = 1$, and $w_0^{(l)}$ are context-independent learnable parameters.

where $T$ is the sequence length and $d$ is the dimensionality. However, attention needs $\mathcal{O}(1)$ *sequential* operations to compute the $T$ output embeddings. In contrast, RNNs require $\mathcal{O}(T)$ sequential operations, making them much slower in practice. ProTran loses this parallelization, and it resembles an RNN where the hidden states have to be computed sequentially for each time step.

**Sampling $z_t$ at Each Layer (II)** and connecting these samples throughout the layers (Equations 7-8) has no benefits for the predictive performance compared to sampling them only in the last layer. On the other hand, it introduces noise in the forward pass and thus to the gradients during training.

**Non-Causal Layer Attention (III)** violates the temporal assumption that $x_{t+1}$ cannot influence $x_t$. Although not a problem for a strictly predictive task, predictions of $z_t$ from the conditional prior can change depending on the forecasting window which influences the length of $w_{1:T}^{(l-1)}$ and, therefore, the output of the layer attention. This is is undesirable. Besides, uncertainty about the future can negatively impact the predictions at previous points.

## 3. E-ProTran: Efficient Probabilistic Transformer for Forecasting

We propose the Efficient Probabilistic Transformer for Forecasting (**E-ProTran**), which effectively addresses and overcomes the aforementioned aspects. We mention the changes we incorporate with E-ProTran and their motivation in the following part.

**(I)** In order to increase computational efficiency, we omit Equation (Autoreg. Att.). This re-enables parallel processing of attention for all time steps, while still allowing information to flow over time through the layer attention. **(II)** We remove the sampling in every layer except the last one thus making our architecture deterministic until the latent bottleneck. Furthermore, we break the explicit dependency between the latent variables $z_t$ at different time steps. **(III)** In the conditional prior, we only employ causal attention to lower layers up until time step $t$. This approach ensures that our predictions are independent of the chosen forecasting window and are unaffected by potential errors and uncertainties at later time steps.

### 3.1. E-ProTran's Formulation

We first project inputs and covariates as in Equation 6 to get $h_{1:T}^{(0)}$. Then, for the conditional prior, we perform[3]

$$\hat{h}_t^{(l)} = \mathrm{LN}(h_t^{(l-1)} + \mathcal{A}(h_t^{(l-1)}, h_{1:t}^{(l-1)}, h_{1:t}^{(l-1)})) \tag{11}$$
$$h_t^{(l)} = \mathrm{MLP}(\hat{h}_t^{(l)}). \tag{12}$$

[3]For $l = 1$, we use the input projections but allow attention over the whole context $h_{1:t_0}^{(0)}$, like in Equation (Input Att.).

To get the latent variables $z_{1:T}$, we use the embeddings $h_{1:T}^{(L)}$ of the last layer, i.e.,

$$z_t \sim \mathcal{N}(\mathrm{MLP}(h_t^{(L)}), \mathrm{diag}(\mathrm{SP}(\mathrm{MLP}(h_t^{(L)})))). \tag{13}$$

We can extend the updates in Equation 11 to get a more similar process to that of ProTran, in other words, we add the same attention computations without introducing random variables. This results in the update equations

$$\tilde{h}_t^{(l)} = \mathrm{LN}(h_t^{(l-1)} + \mathcal{A}(h_t^{(l-1)}, h_{1:t}^{(l-1)}, h_{1:t}^{(l-1)})) \tag{14}$$
$$\bar{h}_t^{(l)} = \mathrm{LN}(\tilde{h}_t^{(l)} + \mathcal{A}(\tilde{h}_t^{(l)}, h_{1:t-1}^{(l)}, h_{1:t-1}^{(l)})) \tag{15}$$
$$\hat{h}_t^{(l)} = \mathrm{LN}(\bar{h}_t^{(l)} + \mathcal{A}(\bar{h}_t^{(l)}, h_{1:t_0}^{(0)}, h_{1:t_0}^{(0)})). \tag{16}$$

We emphasize that the inclusion of autoregressive attention (Equation 15) and input attention (Equation 16) are purely optional, and autoregressive attention hinders parallelization. Note that we do not have to integrate both equations at once, but we can add them individually.

The network for the approximate posterior can now be parameterized arbitrarily, but we usually fix the architecture such that the posterior network shares its parameters with the prior network, i.e., $\psi = \phi$. The difference is that the posterior network can attend the whole input projection $h_{1:T}^{(0)}$ during training, and it is not restricted to time step $t$ in the lower layer attention. For the emission model, we use the exact same approach as ProTran with the only exception that we decode $z_t$ instead of $w_t^{(L)}$. We only have one layer of latent variables that do not depend on each other, so we get a more simplistic decomposition, where the conditional prior distribution $p_\psi$ and the posterior distribution $q_\phi$ only depend on the inputs and the covariates. We provide this decomposition in Appendix A.3.

We can use the ELBO objective in Equation 5 to train the model end-to-end. Importantly, this objective consists of two parts,

$$\begin{aligned}&\log(p_{\psi,\theta}(x_{1:T}|x_{1:t_0}, c_{1:T})) \geq \\ &\sum_{t=1}^{t_0} \left(\mathbb{E}^* \left[\log(p_\theta(x_t|z_t))\right] - D_{\mathrm{KL}}(q_\phi \parallel p_\psi)\right) \\ &+ \sum_{t=t_0+1}^{T} \left(\mathbb{E}^* \left[\log(p_\theta(x_t|z_t))\right] - D_{\mathrm{KL}}(q_\phi \parallel p_\psi)\right),\end{aligned} \tag{17}$$

where $\mathbb{E}^* = \mathbb{E}_{z_t \sim q_\phi}$. The first term enforces the model to reconstruct the conditioning part $x_{1:t_0}$ of a sequence. The second term optimizes the model to forecast. Tang & Matteson (2021) opt to train ProTran on the full ELBO over all time steps. However, to learn forecasting, it is sufficient to only use the forecasting part of the ELBO. For E-ProTran, we experiment with both versions, maximizing the ELBO over the entire sequence (i.e., with reconstruction) or just over the forecasting part (i.e., without reconstruction).

*Table 1.* **Performance of ProTran and E-ProTran on Electricity and Wikipedia.** The CRPS and CRPS$_{\text{sum}}$ are normalized. **L**ayer, **I**nput and **A**utoreg. attentions are abbreviated by their first letter. For all metrics, lower is better.

| Model | | | Electricity | | | | | Wikipedia | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Type | Att. | Rec. | CRPS$_{\text{sum}}$ | CRPS | RMSE | #Params | Forw. Pass (sec) | CRPS$_{\text{sum}}$ | CRPS | RMSE | #Params | Forw. Pass (sec) |
| ProTran | L I A | ✓ | 0.030 | **0.069** | **439.2** | 501,924 | 0.064 ± 0.001 | 0.066 | 0.320 | 5909.2 | 1,522,080 | 0.187 ± 0.003 |
| E-ProTran | L I A | ✓ | **0.024** | 0.075 | 501.7 | 382,084 | 0.044 ± 0.001 | 0.075 | 0.353 | 6020.0 | 1,259,584 | 0.141 ± 0.005 |
| E-ProTran | L I | ✓ | – | – | – | – | – | 0.063 | 0.328 | 5932.2 | 1,126,720 | **0.007 ± 0.001** |
| E-ProTran | L | ✓ | **0.024** | 0.072 | 530.2 | **315,652** | **0.002 ± 0.000** | 0.063 | **0.311** | 5936.8 | **1,060,416** | **0.007 ± 0.002** |
| E-ProTran | L I A | ✗ | 0.030 | 0.077 | 582.8 | 382,084 | 0.044 ± 0.001 | 0.081 | 0.354 | 5983.3 | 1,259,584 | 0.139 ± 0.005 |
| E-ProTran | L I | ✗ | – | – | – | – | – | 0.054 | 0.327 | 5945.1 | 1,126,720 | **0.007 ± 0.002** |
| E-ProTran | L | ✗ | 0.029 | 0.079 | 598.7 | **315,652** | **0.003 ± 0.001** | **0.053** | 0.316 | **5906.7** | **1,060,416** | **0.007 ± 0.002** |

## 4. Experiments

For the experiments, we follow the described approach by Tang & Matteson (2021) as closely as possible. Since the code is not published, we re-implemented ProTran and its evaluation. The code for ProTran and E-ProTran is available in `https://github.com/bkoyuncu/eprotan`.

### 4.1. Metrics

The most commonly used metric in probabilistic time series forecasting is the continuous ranked probability score (CRPS) (Matheson & Winkler, 1976), which is defined by

$$\mathrm{CRPS}(F, x) = \int_{\mathbb{R}} \left(F(z) - \mathbb{I}_{[x \leq z]}\right)^2 dz. \tag{18}$$

Here, $\mathbb{I}$ is the indicator function, $x \in \mathbb{R}$ is the target, and $F$ is the CDF of the prediction for $x$. For deterministic point forecasts, CRPS is equivalent to the absolute error. To obtain a metric that generalizes to the multivariate case, the mean CRPS or the mean CRPS$_{\text{sum}}$ are reported (e.g., see Tang & Matteson (2021); Salinas et al. (2019)). The latter reduces the data dimension to 1 by summing over all dimensions and then calculating the CRPS. Hence, the mean CRPS is taken over dimensions and time steps, whereas the mean CRPS$_{\text{sum}}$ only needs to consider the different time steps. To estimate the CDF of the forecasting distribution, we follow previous work and generate 100 samples. Furthermore, we calculate the mean forecast from 100 samples to calculate the RMSE over dimensions and time steps. We then average the CRPS$_{\text{sum}}$, CRPS, and RMSE of the forecasting days. The exact computations for the metrics are in Appendix B.

### 4.2. Datasets and Setup

We use three datasets, Electricity, Solar, and Wikipedia, from ProTran's evaluation. These datasets are available in the GluonTS package (Alexandrov et al., 2019) and were pre-processed by Salinas et al. (2019). Electricity and Solar are hourly datasets, while Wikipedia contains daily data. Testing is performed with non-overlapping windows. The test series includes data after a specified cut-off date, using all prior data for training. The conditioning length matches the forecasting horizon: 24 time steps for hourly data and 30 for daily data. More details about the datasets are in Appendix C.

For the model architectures, we adhere to Tang & Matteson (2021): We use 1 layer for Electricity and Solar, and 2 layers for Wikipedia. We use 8-head multihead attention. Hidden representations $(w_t, h_t)$ are in $\mathbb{R}^{128}$, with a latent dimension of $d_z = 16$. All MLPs have two hidden layers of size 128 with ReLU activation and a linear output layer. These configurations apply to ProTran and E-ProTran.

We normalize $x_{1:T}$ and $c_{1:T}$ before feeding them to the models, and we denormalize the predictions before computing performance metrics including CRPS$_{\text{sum}}$, CRPS, and RMSE. We give a more detailed description of the model in Appendix D. All of our experiments are conducted using a single NVIDIA A100 GPU, ensuring a fair comparison between E-ProTran and ProTran.

### 4.3. Results

We present the performance metrics for Electricity and Wikipedia in Table 1 and for Solar in Appendix E.1. We also report the number of parameters and the inference times. The reported inference time is the average time the prior and emission models need for a forward pass on 100 different batches. Note that for the first layer, input attention replaces layer attention, so there are less configurations with 1-layered models. Moreover, in Figures 1 and 2, we present forecasting distributions of the different models on the respective test sets. The results and plots for each model are based on the best training configuration for this architecture. We describe the different configurations in Appendix D.3.

As shown in Table 1, we find that our models perform similarly to or better than ProTran with the same number of

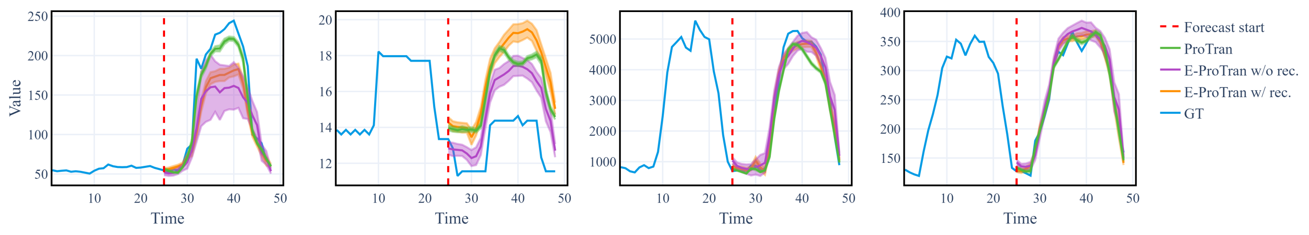

*Figure 1.* **Forecasting predictions on Electricity** for the first test sequence. We show 4 of 370 dimensions. The average predictions are from 100 forward passes; shaded areas show 1 standard deviation. For E-ProTran, only layer attention is used.

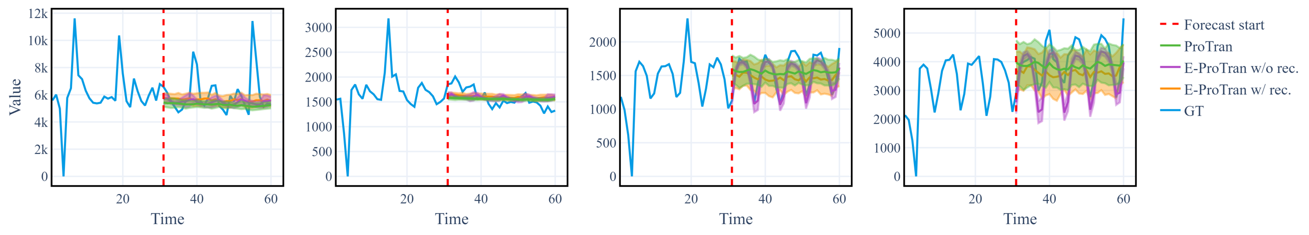

*Figure 2.* **Forecasting predictions on Wikipedia** for the first test sequence. We show 4 of 2000 dimensions. The average predictions are from 100 forward passes; shaded areas show 1 standard deviation. For E-ProTran, only layer attention is used.

layers.[4] We have fewer parameters since E-ProTran does not have random variables between layers, which decreases the amount of MLPs per layer. Moreover, E-ProTran shares all parameters between the prior and posterior networks, except possibly the last layer. Remarkably, enabling parallel processing without autoregressive attention improves the forward pass time by a factor of 20 without harming the predictive performance for all datasets. We observe similar improvements for the training and back propagation times, which we show for Wikipedia in Appendix E.2.

It is not apparent whether including the reconstruction part in the ELBO (Equation 17) leads to better predictive power. Our empirical results show that using the reconstruction loss helps with Electricity but generally worsens the performance on Wikipedia; on Solar, the evidence is inconclusive. We hypothesize that it may be depending on many factors such as the conditioning length, the distribution of the time series in the conditioning and forecasting windows, and hyperparameter tuning. For example, the two rightmost plots in Figure 1 are typical signals for Electricity, where the conditioning and forecasting series behave almost identical. As shown in Figure 2, this is not the case for Wikipedia, where the model without reconstruction loss performs best. For Wikipedia, models with reconstruction closely match the history, but tend to predict only the mean for the forecasting part, whereas models without reconstruction capture temporal patterns in the forecasting window.

[4]We tried to reproduce the results for ProTran; however, they are not the same as reported by Tang & Matteson (2021).

## 5. Conclusion

In this paper we showed that ProTran, a model for probabilistic time series forecasting, can be refined to an efficient transformer E-ProTran with a comparable performance. The advantage of our model is its improved speed by forgoing autoregressive attention, making it much more scalable. We also demonstrated that complex sampling processes in prior and posterior networks do not necessarily lead to better performance. Ultimately, E-ProTran offers an efficient alternative for probabilistic time series forecasting, enabling faster, more scalable solutions for real-world applications.

For future work, we plan to dive deeper into the scalability and performance aspects of our model. So far, we have mainly focused on the direct comparison with ProTran, matching it in many aspects such as the number of layers. However, we can easily scale our lightweight model to incorporate more layers and longer conditioning lengths without reaching a problematic runtime. Therefore, we want to test how well E-ProTran performs without these restrictions.

We are also committed to better understanding the different factors that influence the optimization of E-ProTran, such as the ELBO with and without reconstruction, the conditioning length, the differences in frequency components in conditioning and forecasting windows, and their interplay.

## 6. Acknowledgments

Batuhan Koyuncu is supported by the Konrad Zuse School of Excellence in Learning and Intelligent Systems (ELIZA) through the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research.

The authors thank to Pablo M. Olmos and Ignacio Peis Aznarte for their valuable discussions and contributions in the early stages of the project.

## References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. GluonTS: Probabilistic Time Series Modeling in Python. *arXiv preprint arXiv:1906.05264*, 2019.

Ao, J., Wang, R., Zhou, L., Liu, S., Ren, S., Wu, Y., Ko, T., Li, Q., Zhang, Y., Wei, Z., Qian, Y., Li, J., and Wei, F. Speecht5: Unified-modal encoder-decoder pre-training for spoken language processing. *ArXiv*, abs/2110.07205, 2021. URL https://api.semanticscholar.org/CorpusID:238856828.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. *ArXiv*, abs/2005.12872, 2020. URL https://api.semanticscholar.org/CorpusID:218889832.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL https://api.semanticscholar.org/CorpusID:52967399.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020. URL https://api.semanticscholar.org/CorpusID:225039882.

Elsayed, S., Thyssens, D., Rashed, A., Schmidt-Thieme, L., and Jomaa, H. S. Do we really need deep learning models for time series forecasting? *CoRR*, abs/2101.02118, 2021. URL https://arxiv.org/abs/2101.02118.

Ghassemi, M., Pimentel, M. A. F., Naumann, T., Brennan, T., Clifton, D. A., Szolovits, P., and Feng, M. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data. *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, 2015:446–453, 2015. URL https://api.semanticscholar.org/CorpusID:5825525.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. URL https://api.semanticscholar.org/CorpusID:216078090.

Matheson, J. E. and Winkler, R. L. Scoring rules for continuous probability distributions. *Management Science*, 22:1087–1096, 1976. URL https://api.semanticscholar.org/CorpusID:119590882.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U. M., and Vollgraf, R. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=WiGQBFuVRv.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014. URL https://api.semanticscholar.org/CorpusID:16895865.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/0b105cf1504c4e241fcc6d519ea962fb-Paper.pdf.

Sonkavde, G., Dharrao, D. S., Bongale, A. M., Deokate, S. T., Doreswamy, D., and Bhat, S. K. Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications. *International Journal of Financial Studies*, 2023. URL https://api.semanticscholar.org/CorpusID:260264948.

Tang, B. and Matteson, D. S. Probabilistic transformer for time series analysis. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 23592–23608. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/c68bd9055776bf38d8fc43c0ed283678-Paper.pdf.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. 2017. URL https://arxiv.org/pdf/1706.03762.pdf.

Zaytar, A. and Amrani, C. E. Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. *International Journal of Computer Applications*, 143:7–11, 2016. URL https://api.semanticscholar.org/CorpusID:1308550.

## A. Additional Mathematical Details

### A.1. PROTRAN's Generalized ELBO Inequality

With multiple latent layers in PROTRAN the ELBO inequality becomes

$$\log(p_{\psi,\theta}(x_{1:T}|x_{1:t_0}, c_{1:T})) \geq \sum_{t=1}^{T} \left( \mathbb{E}_{z_t^{(L)} \sim q_\phi^{(L)}} \left[ \log(p_\theta(x_t|z_t^{(L)})) \right] - \sum_{l=1}^{L} D_{\mathrm{KL}} \left( q_\phi^{(l)} \parallel p_\psi^{(l)} \right) \right), \tag{19}$$

with $q_\phi^{(l)} = q_\phi(z_t^{(l)}|z_{1:t-1}^{(l)}, z_{1:T}^{(l-1)}, x_{1:T}, c_{1:T})$ and $p_\psi^{(l)} = p_\psi(z_t^{(l)}|z_{1:t-1}^{(l)}, z_{1:T}^{(l-1)}, x_{1:t_0}, c_{1:T})$. Note that $q_\phi^{(l)}$ and $p_\psi^{(l)}$ depend on $t$.

### A.2. PROTRAN's Probabilistic Decomposition

Here we show the mathematical decomposition property that is assumed with the multilayered PROTRAN model and justifies the ELBO inequality in Appendix A.1. The joint, conditional prior, and approximate posterior distribution factorize as follows:

$$p_{\psi,\theta}(x_{1:T}, z_{1:T}^{(1:L)}|x_{1:t_0}, c_{1:T}) = \left( \prod_{t=1}^{T} p_\theta(x_t|z_t^{(L)}) \right) \left( \prod_{l=1}^{L} \prod_{t=1}^{T} p_\psi(z_t^{(l)}|z_{1:t-1}^{(l)}, z_{1:T}^{(l-1)}, x_{1:t_0}, c_{1:T}) \right) \tag{20}$$

$$q_\phi(z_{1:T}^{(1:L)}|x_{1:T}, c_{1:T}) = \prod_{l=1}^{L} \prod_{t=1}^{T} q_\phi(z_t^{(l)}|z_{1:t-1}^{(l)}, z_{1:T}^{(l-1)}, x_{1:T}, c_{1:T}). \tag{21}$$

Again, we treat edge cases with $z_{1:0}^{(l)} = z_{1:T}^{(0)} = \emptyset$ for notational simplicity.

### A.3. E-PROTRAN's Probabilistic Decomposition

For our model, the approximate posterior and the conditional prior have a more simplistic decomposition compared to PROTRAN (Appendix A.2), since we do not introduce multiple layers of latent variables, and our latents are independent of each other. Hence, we get

$$p_{\psi,\theta}(x_{1:T}, z_{1:T}|x_{1:t_0}, c_{1:T}) = \left( \prod_{t=1}^{T} p_\theta(x_t|z_t) \right) \left( \prod_{t=1}^{T} p_\psi(z_t|x_{1:t_0}, c_{1:T}) \right) \tag{22}$$

$$q_\phi(z_{1:T}|x_{1:T}, c_{1:T}) = \prod_{t=1}^{T} q_\phi(z_t|x_{1:T}, c_{1:T}). \tag{23}$$

## B. Exact Computation of the Performance Metrics

**CRPS** To get the mean CRPS over the forecasting time steps and dimensions, we calculate

$$\mathbb{E}_{t,i} \left[ \mathrm{CRPS}(\hat{F}, x) \right] = \frac{1}{d(T - t_0)} \sum_{t=t_0+1}^{T} \sum_{i=1}^{d} \int_{\mathbb{R}} \left( \hat{F}_{t,i}(z) - \mathbb{I}_{[x_{t,i} \leq z]} \right)^2 dz, \tag{24}$$

where $d$ is the number of dimensions and $T - t_0$ is the number of forecasting time steps. $\hat{F}_{t,i}$ is the estimated CDF that the model predicts for $x_t$ in dimension $i$. To estimate the CDF $F$, we generate $S = 100$ samples of the model, i.e. 100 forward passes. Given 100 samples $\hat{x}^{(s)} \in \mathbb{R}$ for a certain dimension and time step, $\hat{F}(z) = \frac{1}{S} \sum_{s=1}^{S} \mathbb{I}_{[\hat{x}^{(s)} \leq z]}$. To calculate the CRPS$_{\text{sum}}$, we use the same formula, with the difference that we first sum over the dimensions of $x$, and then calculate the mean CRPS of the time steps. Note that for the samples we also sum over the dimensions to get an estimated CDF for this sum. Mathematically, let $y_t = \sum_{i=1}^{d} x_{t,i}$, $\hat{y}_t^{(s)} = \sum_{i=1}^{d} \hat{x}_{t,i}^{(s)}$, and $\hat{G}_t(z) = \frac{1}{S} \sum_{s=1}^{S} \mathbb{I}_{\left[\hat{y}_t^{(s)} \leq z\right]}$. The mean CRPS$_{\text{sum}}$ is then

$$\mathbb{E}_t \left[ \mathrm{CRPS}_{\text{sum}}(\hat{G}, x) \right] = \frac{1}{(T - t_0)} \sum_{t=t_0+1}^{T} \int_{\mathbb{R}} \left( \hat{G}_t(z) - \mathbb{I}_{[y_t \leq z]} \right)^2 dz. \tag{25}$$

Often, the *normalized* CRPS$_{\text{sum}}$ and CRPS are reported as they are implemented in the GluonTS (Alexandrov et al., 2019) package. To get the normalized version of both metrics, $\mathbb{E}_{t,i}\left[\text{CRPS}(\hat{F}, x)\right]$ and $\mathbb{E}_t\left[\text{CRPS}_{\text{sum}}(\hat{G}, x)\right]$ are divided by the mean absolute value of the forecasting time series, i.e., $\frac{1}{d(T-t_0)} \cdot \sum_{t=t_0+1}^{T} \sum_{i=1}^{d} |x_{t,i}|$.

**RMSE** Since PROTRAN and E-PROTRAN produce different samples for each forward pass due to the sampling, we take the mean of $S = 100$ samples $\hat{x}_{1:T}^{(s)} \in \mathbb{R}^{T\times d}$, $\bar{x}_{1:T} = \frac{1}{S}\sum_{s=1}^{S} \hat{x}_{1:T}^{(s)}$, to calculate the RMSE to the ground truth. Note that with PROTRAN the different latent samples also depend on one another, so we cannot simply decode the mean of the final latent distribution to get the mean prediction. We report the RMSE, which is defined as

$$\sqrt{\frac{1}{d(T-t_0)} \sum_{t=t_0+1}^{T} \sum_{i=1}^{d} (x_{t,i} - \bar{x}_{t,i})^2}. \tag{26}$$

Since we have multiple test series, we report the mean RMSE over the different test samples.

## C. Datasets

The datasets that we use contain the following data:

- ELECTRICITY: Hourly electricity consumption of 370 customers.
- SOLAR: Hourly photo-voltaic production of 137 stations in Alabama State.
- WIKIPEDIA: Daily page views of 2000 Wikipedia pages.

As in previous work (Tang & Matteson, 2021; Rasul et al., 2021; Salinas et al., 2019), we concatenate the single dimensional time series to get multidimensional ones. The idea is that the model can leverage correlations between the different dimensions/time series for its predictions. Table 2 summarizes the statistics about the datasets.

*Table 2.* **Statistics for the datasets.**

| Data Set | Domain | Freq. | Dimension | #Training Steps | Pred. Length | Cond. Length | #Test Dates |
|---|---|---|---|---|---|---|---|
| ELECTRICITY | $\mathbb{R}^+$ | hourly | 370 | 5,833 | 24 | 24 | 7 |
| SOLAR | $\mathbb{R}^+$ | hourly | 137 | 7,009 | 24 | 24 | 7 |
| WIKIPEDIA | $\mathbb{N}$ | daily | 2,000 | 792 | 30 | 30 | 5 |

## D. Model

### D.1. Input Processing

Before we input the data to the model, we standard scale each dimension with the mean and standard deviation calculated for this dimension across all *training* time steps. As covariates, we use the hour of the day, the day of the week, and the month of the year for hourly data. For daily data, we only use the day of the week and the month of the year. We normalize these covariates like the data, and also concatenate lag values as described by Rasul et al. (2021).

### D.2. Additional Model Details

Here we describe additional details that are not mentioned in Section 4. The MLPs that generate the means and variances of the latent variables share all of their parameters until the output layer. Furthermore, the input and output dimensions of each MLP can be inferred, that is, MLPs that have a latent variable $z_t$ or its distributional parameters as input/ouput have the input/output dimension $d_z = 16$. The same principle applies to MLPs that work on hidden states $(w_t, h_t)$ or the input $(x_t)$. The MLP that gives the initial embedding in Equation 6 is a simple linear layer without an activation function.

The positional embedding is fixed and it is obtained in the usual way from the sine and cosine functions. All layer normalization layers use $\varepsilon = 10^{-5}$ and have learnable parameters. We use a dropout with a probability of $0.1$ on the attention weights as well as on the output of the attention blocks.

### D.3. Training Configurations for the Models

For training, we try 6 different configurations for each architecture that start from the same random seed, so the models have the same initialization for each run. We use the ADAM optimizer (Kingma & Ba, 2017) and try learning rates $0.001$, $0.0003$ (learning rate in Tang & Matteson (2021)), and $0.0001$. We test each of these learning rates with and without learning rate decay. We decay the learning rate by multiplying it with $0.3$ after a certain amount of training steps. For Electricity and Solar, we do it for every quarter of the maximum training steps that is reached; for Wikipedia, we do it for every fifth of the maximum training steps. We report all results on the best models from these 6 training procedures. We determine the best models by their $\text{CRPS}_{\text{sum}}$.

To build training batches, we randomly sample starting points in the training part of the time series, and then take the following $T$ time steps to create training samples. We use a batch size of $64$ and train 8,000 update steps for Electricity, 12,000 for Solar, and 1,500 for Wikipedia.

## E. Additional Results

### E.1. Performance on Solar

In Table 3, we present the same statistics as in Table 1, but for the Solar dataset. We also include the results for E-ProTran, where the posterior network does not share all parameters with the prior network. Instead, the posterior network takes the last hidden states $h_{1:T}^{(L)}$ of the conditional prior network as input. It then performs self-attention over the full input projection $h_{1:T}^{(0)}$ and generates the latent distribution parameters via an MLP from the concatenation of the self-attention output and $h_{1:T}^{(L)}$. These computations are the same as in Equations 9 and 10 for ProTran, but only applied in the last layer. While we generally do not find an improvement over the usual E-ProTran architecture with this attention head, for a specific configuration it performs better on the Solar dataset. We include this configuration under E-ProTran(H) in the table.

*Table 3.* **Performance of ProTran and E-ProTran on Solar.** The CRPS and $\text{CRPS}_{\text{sum}}$ are normalized. **L**ayer, **I**nput and **A**utoreg. attentions are abbreviated by their first letter. For all metrics, lower is better.

| Model | | | Solar | | | | |
|---|---|---|---|---|---|---|---|
| Type | Att. | Rec. | $\text{CRPS}_{\text{sum}}$ | CRPS | RMSE | #Params | Forw. Pass (sec) |
| ProTran | L I A | ✓ | 0.240 | **0.279** | **24.43** | 411,986 | $0.064 \pm 0.001$ |
| E-ProTran | L I A | ✓ | 0.306 | 0.364 | 29.00 | 292,146 | $0.044 \pm 0.003$ |
| E-ProTran | L | ✓ | 0.331 | 0.386 | 29.56 | **225,714** | $\mathbf{0.002 \pm 0.000}$ |
| E-ProTran | L I A | ✗ | 0.991 | 0.991 | 58.48 | 292,146 | $0.044 \pm 0.001$ |
| E-ProTran | L | ✗ | 0.328 | 0.380 | 30.48 | **225,714** | $\mathbf{0.002 \pm 0.000}$ |
| E-ProTran(H) | L | ✗ | **0.239** | 0.301 | 26.16 | 345,298 | $\mathbf{0.002 \pm 0.000}$ |

### E.2. Training Times on Wikipedia

In Table 4, we show the the forward pass and backpropagation times (in seconds) on Wikipedia for the different models. The times are tracked over the whole training process for all training batches. The optimization parameters are the same for all models, that is, the learning rate is $0.0003$ and we do not use learning rate decay. Notably, for E-ProTran, we implemented the possibility to not share the architecture between prior and posterior networks, hence, we do not couple their forward passes. For ProTran, we do the forward pass for posterior and prior networks at the same time. Therefore, the times for E-ProTran can be further improved by sharing the forward pass during training when prior and posterior share their parameters.

### E.3. Forecasting Plots

In Figures 3, 4, and 5, we provide more of the forecasting plots. Each of them shows the forecasting distributions of the different models over the first 12 dimensions of the first test series for each dataset. E-ProTran models are the simplest

*Table 4.* **Training times on Wikipedia.** We show the average forward pass and backpropagation times.

| Model | | | Training Times (sec) | |
|---|---|---|---|---|
| Type | Att. | Rec. | Forward Pass | Backpropagation |
| ProTran | L I A | ✓ | $0.393 \pm 0.027$ | $0.507 \pm 0.012$ |
| E-ProTran | L I A | ✓ | $0.373 \pm 0.027$ | $0.500 \pm 0.005$ |
| E-ProTran | L I | ✓ | $0.016 \pm 0.015$ | $0.020 \pm 0.003$ |
| E-ProTran | L | ✓ | $0.011 \pm 0.013$ | $0.009 \pm 0.002$ |
| E-ProTran | L I A | ✗ | $0.381 \pm 0.026$ | $0.510 \pm 0.008$ |
| E-ProTran | L I | ✗ | $0.012 \pm 0.013$ | $0.010 \pm 0.002$ |
| E-ProTran | L | ✗ | $0.014 \pm 0.016$ | $0.016 \pm 0.003$ |

version with only lower layer attention (bottom rows in Tables 1 and 3). For each model, we take the best one out of the different training configurations according to the test $\text{CRPS}_{\text{sum}}$, so the models are the same as in Table 1 and Appendix E.1. For Solar, we also include the E-ProTran(H) model.

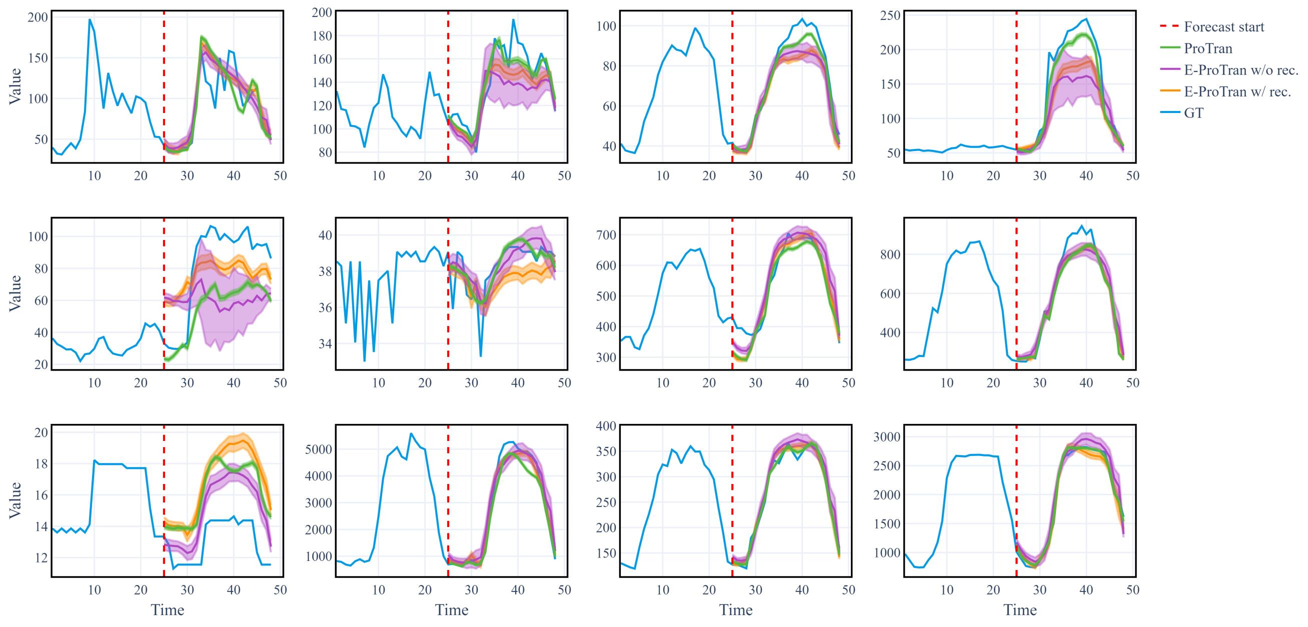

*Figure 3.* **Forecasting predictions on Electricity** for the first test sequence. We show the first 12 of 370 dimensions. The average predictions are from 100 forward passes; shaded areas show 1 standard deviation. For E-ProTran, only layer attention is used.

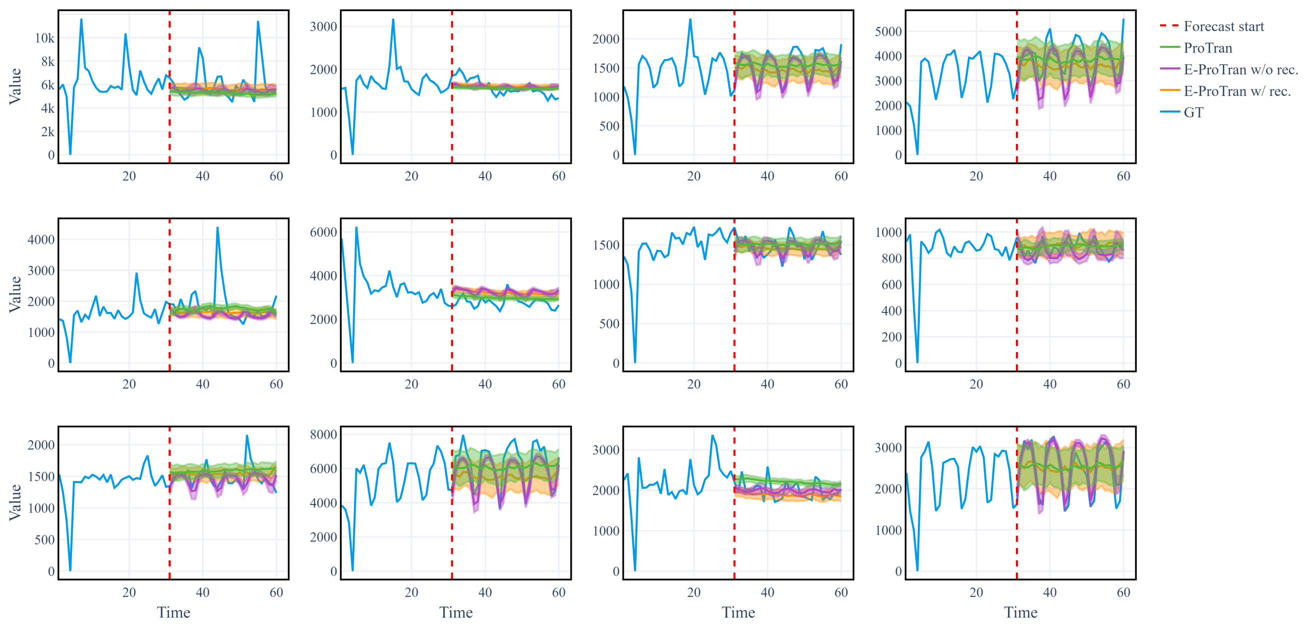

*Figure 4.* **Forecasting predictions on Wikipedia** for the first test sequence. We show the first 12 of 2000 dimensions. The average predictions are from 100 forward passes; shaded areas show 1 standard deviation. For E-ProTran, only layer attention is used.

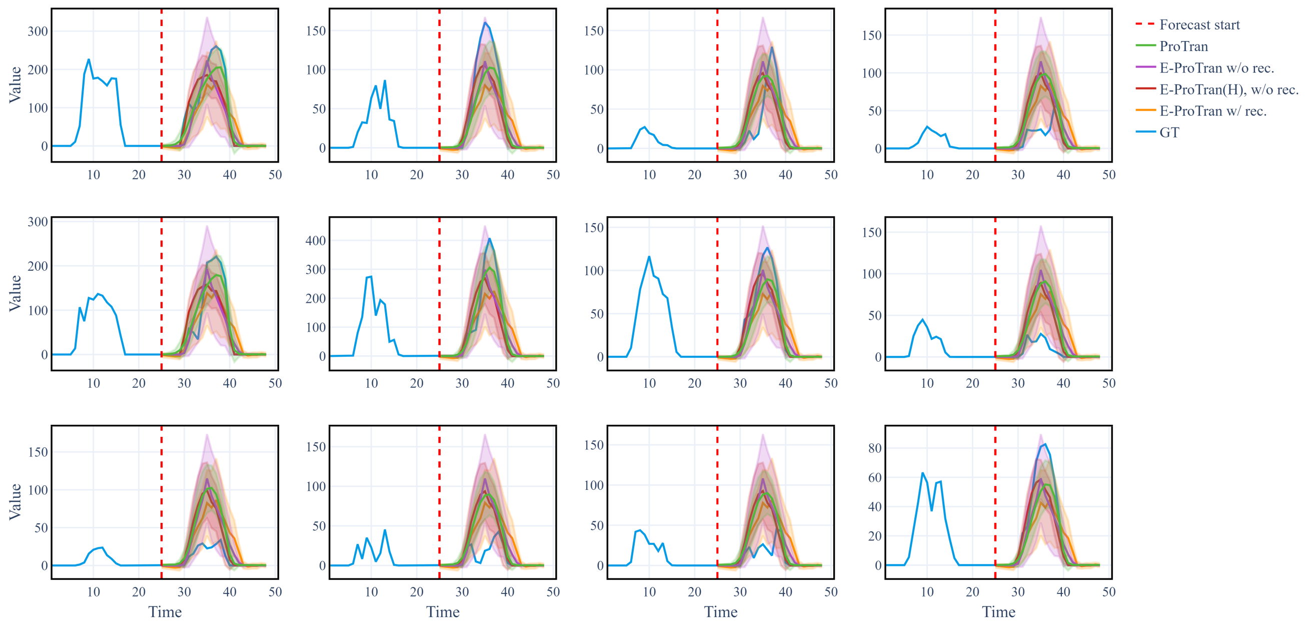

*Figure 5.* **Forecasting predictions on Solar** for the first test sequence. We show the first 12 of 137 dimensions. The average predictions are from 100 forward passes; shaded areas show 1 standard deviation. For E-ProTran, only layer attention is used.