Adversarial Sparse Transformer for Time Series Forecasting

Sifan Wu[*]
Tsinghua University
wusf18@mails.tsinghua.edu.cn

Xi Xiao
Tsinghua University/Peng Cheng Laboratory
xiaox@sz.tsinghua.edu.cn

Qianggang Ding[*]
Tsinghua University
dqg18@mails.tsinghua.edu.cn

Peilin Zhao[†]
Tencent AI Lab
masonzhao@tencent.com

Ying Wei
Tencent AI Lab
judywei@tencent.com

Junzhou Huang
University of Texas at Arlington/Tencent AI Lab
jzhuang@uta.edu

Abstract

Many approaches have been proposed for time series forecasting, in light of its significance in wide applications including business demand prediction. However, the existing methods suffer from two key limitations. Firstly, most point prediction models only predict an exact value of each time step without flexibility, which can hardly capture the stochasticity of data. Even probabilistic prediction using the likelihood estimation suffers these problems in the same way. Besides, most of them use the auto-regressive generative mode, where ground-truth is provided during training and replaced by the network's own one-step ahead output during inference, causing the error accumulation in inference. Thus they may fail to forecast time series for long time horizon due to the error accumulation. To solve these issues, in this paper, we propose a new time series forecasting model – Adversarial Sparse Transformer (AST), based on Generative Adversarial Networks (GANs). Specifically, AST adopts a Sparse Transformer as the generator to learn a sparse attention map for time series forecasting, and uses a discriminator to improve the prediction performance at a sequence level. Extensive experiments on several real-world datasets show the effectiveness and efficiency of our method.

1 Introduction

Time series forecasting has demonstrated its wide applications in business and industrial decision-making. For example, demand forecasting of energy consumption helps optimize the resource allocation and dispatch the power generation. There are many classical approaches to solve time series forecasting problems, such as Auto Regressive Integrated Moving Average(ARIMA) [3] models or exponential smoothing [7]. They incorporate prior knowledge about time series structures such as trend, seasonality and so on, and can achieve good performance for single linear time series prediction. But they are ineffective in predicting complex time series data, partly because of their inability to utilize the time-related features.

[*] This work is done when Sifan Wu and Qianggang Ding work as interns at Tencent AI Lab.

[†] Corresponding author

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Recent years, to solve the modern large-scale multiple time series forecasting problems, deep neural networks [11, 20, 24, 25, 30, 14] have been applied to model complicated sequential data. Naturally, Recurrent Neural Network(RNN)-based [11, 24] and attention-based [14, 30] models are utilized to mine complex patterns for time series trends. However, all these models only optimize one specific objective such as the likelihood loss, MSE loss or other losses, while the real-world time series datasets have some tend of stochasticity which can hardly be modeled with a specific non-flexible objective. Therefore, it is inappropriate to only optimize a single forecasting objective for time series forecasting models.

Another key issue of existing methods is the error accumulation. Most auto-regressive generator models adopt the teacher forcing strategy [12], where the previous target values are known during training. While during inference, real previous target values are replaced by previously generated values, which causes discrepancy between training and inference. The discrepancy leads to error accumulation, since the model can hardly handle the errors which never occur in the training process. Recently, some non-autoregressive forecasting models have been proposed to resolve the error accumulation. However these models neglect the position information between steps, which is essential for time series forecasting, causing an inferior performance. Therefore, it has become one of the most important issues to alleviate the error accumulation and meanwhile improve the performance via training the time series forecasting model appropriately.

Generative adversarial networks (GANs) [6] use an adversarial training procedure to directly shape the output distribution of the network via back-propagation. Motivated by GANs, in this paper, we propose Adversarial Sparse Transformer (AST) for multiple time series forecasting, which is a framework that combines a modified Transformer and Generative Adversarial Networks (GANs). The discriminator can regularize the modified Transformer at the sequence level and make it learn a better representation for time series, thereby eliminating the error accumulation and remedying the shortcomings of single forecasting objective. Specifically, the Vanilla Transformer is based on the multi-head attention mechanism where the representation of each time step is represented by multiple different weighted average of samples of its relevant time steps. The attention distribution of each head is typically computed by the softmax normalizing transformation, allocating non-zero attention weights to all samples. Considering only a few historical steps have strong correlations with the forecasting time step, we use sparse normalizing transforms like α-entmax [21], which can yield exactly zero probability for irrelevant time steps.

The main contributions of our paper are as follows:

- We propose an effective time series forecasting model – Adversarial Sparse Transformer based on sparse Transformer and Generative Adversarial Networks. Extensive experiments on different real-world time series datasets show the effectiveness of our model.
- We design a Generative Adversarial Encoder-Decoder framework to regularize the forecasting model which can improve the performance at the sequence level. The experiments show that adversarial training improves the robustness and generalization of the model.

The rest of this paper is organized as follows. Section 2 reviews related works on time series forecasting briefly. Section 3 proposes the background of this model. Section 4 describes the model we propose. In Section 5, we demonstrate the effectiveness of AST empirically. Finally we conclude in Section 6.

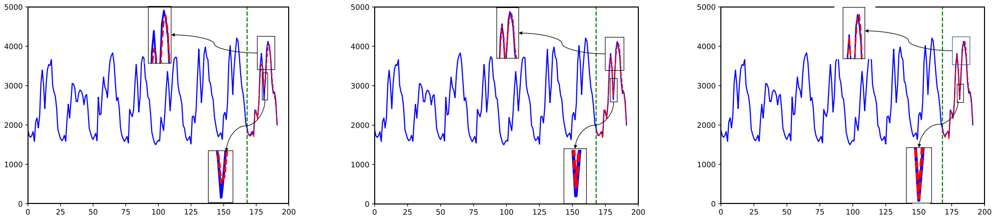
2 Related Work

Time series forecasting Early literature on time series forecasting mostly relies on statistical models. The Box-Jenkins ARIMA [15] family of methods develop a model where the prediction is a weighted linear sum of recent past observations or lags. Liu et al. [15] applied online learning to ARIMA models for time series forecasting. Matrix factorization methods [8, 32] model related series data as a matrix and attempt to learn information across time series. However, it is difficult to predict the modern time series by traditional statistical models because of the complex structure and interdependence between groups of series. Recent years, many researchers have also applied neural networks solving time series forecasting [1, 13, 25]. Langkvist et al. [13] provided an overview of the methods modeling time series forecasting by deep learning and unsupervised feature learning. Bian et al. [1] compared five different architectures of recurrent neural networks for time series forecasting. DeepAR [25] presents an encoder-decoder structure by an auto-regression RNNs modeling probabilistic distributions in the future. The dual-attention model in [22] can only predict one step ahead, which is not suitable for mid-term and long-term time series forecasting. For LSTNet [11], the model is based on RNN architecture and Auto-Regressive to catch long- and short-term dependencies. However, RNN-based models have been proved to be inefficient in dealing with long-term dependencies [27]. However, these models only optimize single objective such as the likelihood loss function with auto-regressive generative mode, which suffer the discrepancy between training and inference as well as the inflexible objective.

Generative Adversarial Networks Generative Adversarial Networks (GANs) have enjoyed great success in computer vision and natural language processing. Several researches use GANs for the generation of sequential data. C-RNN-GAN [19] applies the GAN architecture to generate sequential melody data, using LSTM networks as the generator and the discriminator. The recent timeGAN [31] first utilizes GANs to generate time series. However, this model aims to generate realistic-like time series which can't be classified with the true time series. The generated time series can only model the history time series distribution rather than forecast the future steps. As a result, none of these researches use GANs for time series forecasting problems. In contrast to all these models, we first introduce the GANs into time series forecasting networks. Attached to a discriminator, we regularize the basic Seq2Seq time series forecasting network to improve the prediction performance of models.

Attention Mechanism Researchers have also invited attention mechanisms into sequential problems[4, 26, 28]. Especially, several recent works [5, 18] have developed sparse attention mechanisms aiming to learn sparse mapping, mostly applied to NMT. Transformer [27] is a novel encoder-decoder model based on the attention mechanism and totally removes recurrent neural networks, which can compute the sequence effectively. [14] aims to solve time series forecasting by a transformer [23] with a decoder only model based on convolutional attention. However, this model only optimizes the step-level maximum likelihood and suffers from error accumulation.

In this work, we introduce adversarial training as a regularization for the sequence-level forecasting of time series, which can help remedy the above issues of time series forecasting models.



(a) Vanilla Transformer prediction (b) Sparse Transformer prediction (c) AST prediction

Figure 1: (a) The exemplar of `electricity` time series prediction results of Vanilla Transformer when conditioning range length (blue line) is 168 and prediction range(red line) length is 24, where we can see the Vanilla Transfomer can hardly predict the shark peaks. (b) The prediction results of Sparse Transformer of the same time series, where the Sparse Transformer improves the performance of Vanilla Transformer but still struggles to predict the sharp peaks. (c) The prediction result of AST of the same time series. AST enjoys the best performance in all shark peaks.

3 Background

Problem Definition Interval prediction is useful in many scenarios such as business decisions and risk management. Quantile regression estimates the value of specific quantile, which is the most direct method to predict intervals. Thus we perform quantile regression for our model (e.g. outputting the $50^{th}, 90^{th}$ percentiles at each time step).

Specifically, let $\{\mathbf{y}_{i,1:t_0}\}_{i=1}^{S}$ denote S related univariate time series, where $\mathbf{y}_{i,1:t_0} = [y_{i,1}, y_{i,2}, \ldots, y_{i,t_0}]$ and $y_{i,t} \in \mathbb{R}$ is the value of time series i at time t. Further, $\mathbf{X}_{i,1:t_0} \in \mathbb{R}^{t_0 \times k}$ are k-dimensional time-independent (such as car line id) or dynamic time-dependent (such as month of the year, day of the week and so on) covariates. We aim to predict the values of the next τ time steps of each quantile for all time series given the past:

$$\hat{\mathbf{Y}}_{\rho,t_0+1:t_0+\tau} = f_\rho(\mathbf{Y}_{1:t_0}, \mathbf{X}_{1:t_0+\tau}; \Phi) \tag{1}$$

where $\hat{\mathbf{Y}}_{\rho,t}$[3] is the ρ^{th} quantile prediction value in the t time step. f_ρ is a prediction model for ρ^{th} quantile. $\Phi \in \mathbb{R}$ is the learnable parameters of the model learned jointly from all S time series. For each time series , we refer to time series $\{\mathbf{Y}_{1:t_0}\}$ as *target* time series, time ranges $[1, t_0]$ as *conditioning range* and $[t_0 + 1, t_0 + \tau]$ as *prediction range*, as illustrated in Figure 1(a). The time point $t_0 + 1$ is the *forecast start time* and $\tau \in \mathbb{N}$ is the *forecast horizon*. Then our model output forecasts of different quantiles by the corresponding quantile objectives.

The Transformer Encoder-decoder based Transformer is a good candidate for time series forecasting, since the attention mechanisms in multi-head attention layers enable the transformer to capture long-term dependencies of time series. Briefly, the encoder and decoder both consist of N identical layers. Each layer includes two main components: the multi-head self-attention layer and the feed-forward network. The multi-head self-attention sub-layer transforms the input $\mathbf{h}$[4] $\in \mathbb{R}^{n\times d}$ into m distinct query, key and value matrices through linear projections, i.e., $\mathbf{Q}_m = \mathbf{h}\mathbf{W}^Q_m, \mathbf{K}_m = \mathbf{h}\mathbf{W}^K_m, \mathbf{V}_m = \mathbf{h}\mathbf{W}^V_m$, where $d_k = \frac{d}{m}$, $\mathbf{W}^Q_m, \mathbf{W}^K_m \in \mathbb{R}^{d\times d_k}$ and $\mathbf{W}^V_h \in \mathbb{R}^{d\times d_v}$ are learnable parameters. Then each head computes a sequence of scores α_m called *scaled dot-product attention*, and the output of m-th head $\mathbf{O}_m$ is:

$$\mathbf{O}_m = \alpha_m \mathbf{V}_m = \mathrm{softmax}(\frac{\mathbf{Q}_m\mathbf{K}_m^T}{\sqrt{d_k}})\mathbf{V}_m. \tag{2}$$

The output of the multi-head attention layer is the linear projection of the concatenation of $\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_m$. The feed-forward layer is composed of two linear projections with a ReLU activation function, i.e., $FFN(\mathbf{O}) = max(0, \mathbf{O}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2$, here $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable weight metrics and $\mathbf{b}_1, \mathbf{b}_2$ are biases. Similarly, the decoder is also composed of multi-head attention layers and positional-wise feed-forward layers.

Generative Adversarial Networks The Generative Adversarial Networks (GANs) [6] framework establishes a min-max adversarial game between two neural networks – a generative model G, and a discriminative model D. The generator aims to generate fake samples. Concurrently, the discriminator model, $D(x)$, is a neural network to distinguish real samples (positive samples) from samples generated by the generator model (negative samples).

4 Model

Generally speaking, our model is based on the encoder-decoder framework with an auxiliary discriminator, as illustrated in Figure 2. The encoder encodes the input $[\mathbf{Y}_{1:t_0}, \mathbf{X}_{1:t_0}]$ to latent variables $(\mathbf{h}_1, \ldots, \mathbf{h}_{t_0})$, which in the next are fed to the decoder together with the corresponding covariates $\mathbf{X}_{t_0:t_0+\tau}$ to generate the prediction range step by step.

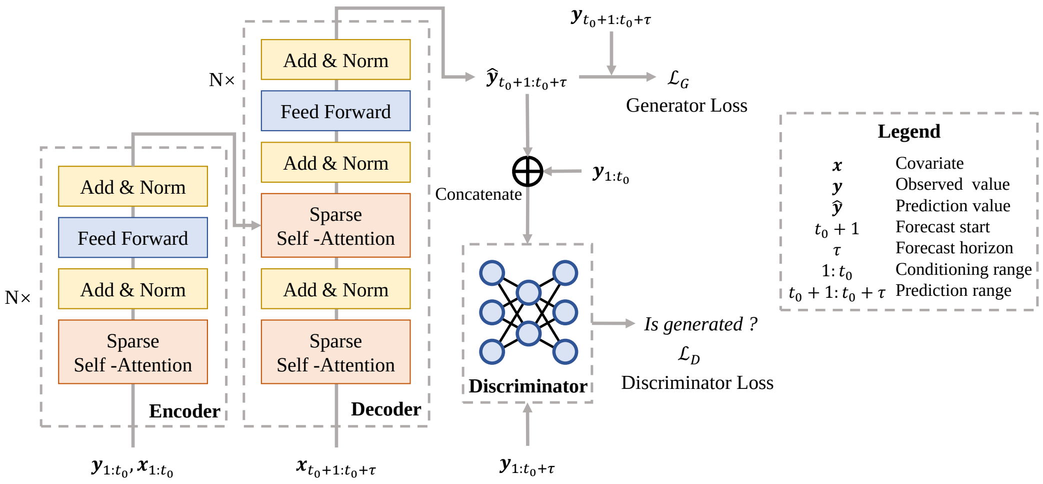
4.1 Sparse Transformer

Considering that only a few historical steps are correlated with the forecasting time step, we should pay no attention to those irrelevant steps. However, in the (multi-head) attention layers of Vanilla Transformer, the attention scores are computed by the softmax mapping defined in Equation 2, which is element-wise proportional to exponent and can never assign an attention score of exactly zero. Since all the scores sum to one, this inevitably means less attention is assigned to the relevant steps, potentially harming performance of the Transformer according to [9]. This has motivated a line of researches on learning networks with sparse mappings [18, 5]. In our paper, we focus on a more recent and flexible family of transformations, α-entmax, [2, 21], defined as,

$$\alpha - entmax(\mathbf{h}) = [(\alpha - 1)\mathbf{h} - \tau\mathbf{1}]_+^{1/\alpha-1}, \tag{3}$$

[3] We omit the ID of time series i for simplicity since all time series would be predicted by the same model

[4] At each time step the same model is applied, so we simplify the formulation with some abuse of notation. Here $\mathbf{h}$ is the intermediate feature vector in encoder or decoder before multi-head attention layers.



$\boldsymbol{y}_{t_0+1:t_0+\tau}$
N×
Add & Norm
Feed Forward
Add & Norm
Sparse Self -Attention
Add & Norm
Sparse Self -Attention
Decoder
$\hat{\boldsymbol{y}}_{t_0+1:t_0+\tau}$
$\mathcal{L}_G$
Generator Loss
Concatenate
$\boldsymbol{y}_{1:t_0}$
N×
Add & Norm
Feed Forward
Add & Norm
Sparse Self -Attention
Encoder
Discriminator
Is generated ?
$\mathcal{L}_D$
Discriminator Loss
Legend
$\boldsymbol{x}$ Covariate
$\boldsymbol{y}$ Observed value
$\hat{\boldsymbol{y}}$ Prediction value
t_0+1 Forecast start
τ Forecast horizon
$1{:}\,t_0$ Conditioning range
$t_0+1{:}\,t_0+\tau$ Prediction range
$\boldsymbol{y}_{1:t_0}, \boldsymbol{x}_{1:t_0}$
$\boldsymbol{x}_{t_0+1:t_0+\tau}$
$\boldsymbol{y}_{1:t_0+\tau}$

Figure 2: Architecture of the Adversarial Sparse Transformer Model

where $[\cdot]_+$ is the ReLU function. $\mathbf{1}$ is the all-one vector, and τ is the Lagrange multiplier. In our proposed sparse Transformer, we simply replace softmax with α-entmax in the attention heads, which can lead to sparse attention weights. When $\alpha = 1$, it equals to the softmax function. When $\alpha = 2$, it recovers sparsemax mapping [18]. Peters et al. [21] claimed that for all $\alpha > 1$, it permits sparse solutions, and $\alpha = 1.5$ is a sensible point. Therefore in our paper, we set the parameter $\alpha = 1.5$ (we further compare the performance of different α in the experiments).

4.2 Adversarial training

Algorithm 1 Adversarial Training for Time Series Forecasting

```
for each training iteration do
    for $k$ steps do
    • Randomly sample $[\mathbf{X}_{1:t_0+\tau}, \mathbf{Y}_{1:t_0+\tau}]$ from training dataset. $\mathbf{Y}_{real} = \mathbf{Y}_{1:t_0+\tau}$.
    • Compute $\mathbf{Y}_{fake}$ by the sparse Transformer $\mathcal{G}$:
          $\mathbf{Y}_{fake} = \mathbf{Y}_{1:t_0} \circ \hat{\mathbf{Y}}_{t_0+1:t_0+\tau} = \mathbf{Y}_{1:t_0} \circ \mathcal{G}(X_{1:t_0+\tau}, Y_{1:t_0})$.
    • Update the Sparse Transformer by stochastic gradient:
          $\bigtriangledown_{\Theta^{\mathcal{G}}}(\mathcal{L}_\rho(\mathbf{Y}_{t_0+1:t_0+\tau}, \mathbf{Y}_{t_0+1:t_0+\tau}) + \lambda\mathbb{E}[log(1 - \mathcal{D}(\mathbf{Y}_{fake})]])$.
    • Update the discriminator by stochastic gradient:
          $\bigtriangledown_{\Theta^{\mathcal{D}}}(\mathbb{E}[-log\mathcal{D}(\mathbf{Y}_{real}) - log(1 - \mathcal{D}(\mathbf{Y}_{fake}))]])$.
    end for
end for
```

Most time series forecasting models optimize a specific objective such as minimizing likelihood loss function or quantile loss function. However, such exact loss function enforcing step level accuracy is incapable of dealing with the real-world stochasticity in time series, thereby leading to inferior performance. Besides, the aforementioned (in Section 1) error accumulation hurts performance badly when forecasting a long horizon of time series. To alleviate these problems, we propose an adversarial training process to regularize the encoder-decoder network and improve the accuracy at the sequence level. Through adversarial training, the encoder-decoder network learns a better representation of time series and forecast multiple future steps of time series with more fidelity at the sequence level.

In order to do so, a discriminator network $\mathcal{D}$ is attached on top of the decoder to improve the sequence-level accuracy as illustrated in Figure 2. Similar to [16], we use a network of three fully connected linear layers with LeakReLu [29] as the activation function, to serve as our discriminator. The discriminator then classifies the input series ($\hat{\mathbf{Y}}$ or $\mathbf{Y}$) as either predicted or ground-truth by minimizing the cross-entropy loss function. Adversarially, the Transformer network, as our generator $\mathcal{G}$, attempts to minimize the quantile loss between prediction series and the ground-truth. In this way, the quantile loss and adversarial loss can be complementary each other. On the one hand, the quantile loss helps the generator to capture the overall pattern of time series and aligns the prediction with the ground-truth, which can prevent the discriminator from converging quickly to local optimal. On the other hand, the discriminator regularizes the prediction from a global perspective.

Formally, let $\Theta^{\mathcal{G}}$ and $\Theta^{\mathcal{D}}$ be the parameters of the generator $\mathcal{G}$ and the discriminator $\mathcal{D}$, respectively. The function $\mathcal{G}(\mathbf{X}_{1:t_0+\tau}, \mathbf{Y}_{1:t_0}; \Theta^{\mathcal{G}})$ output the prediction range sequences $\hat{\mathbf{Y}}_{t_0:t_0+\tau}$, which then be input to $\mathcal{G}$. Let $\mathcal{D}(\mathbf{Y})$ be the output of discriminator, which outputs ones if the input is a ground-truth time series or zero otherwise. Consequently the AST solves the minmax optimization problem:

$$\arg\min_{\mathcal{G}}\max_{\mathcal{D}} \lambda\mathcal{L}_{adv}(\Theta^{\mathcal{G}}, \Theta^{\mathcal{D}}) + \mathcal{L}_{\rho}(\Theta^{\mathcal{G}}), \tag{4}$$

$$\mathcal{L}_{adv}(\Theta^{\mathcal{G}}, \Theta^{\mathcal{D}}) = \mathbb{E}[\log(\mathcal{D}(\mathbf{Y}_{real})] + \mathbb{E}[\log(1 - \mathcal{D}(\mathbf{Y}_{fake}))], \tag{5}$$

$$\mathcal{L}_{\rho}(\Theta^{\mathcal{G}}) = 2\sum_{i=0}^{S}\sum_{t=t_0+1}^{t_0+\tau} P_{\rho}(y_{i,t}, \hat{y}_{i,t}), \quad P_{\rho}(y_{i,t}, \hat{y}_{i,t}) = \Delta y_{i,t}(\rho I_{\hat{y}_{i,t} > y_{i,t}} - (1-\rho) I_{\hat{y}_{i,t} \leq y_{i,t}}), \tag{6}$$

$$\Delta y_{i,t} = (\hat{y}_{i,t} - y_{i,t}), \mathbf{Y}_{fake} = (\mathbf{Y}_{1:t_0} \circ \hat{\mathbf{Y}}_{t_0+1:t_0+\tau}), \mathbf{Y}_{real} = (\mathbf{Y}_{1:t_0+\tau}), \tag{7}$$

where λ is the trade-off hyper-parameter that balances $\mathcal{L}_{adv}$ and $\mathcal{L}_{\rho}$, and $\hat{\mathbf{Y}}_{t_0+1:t_0+\tau}$ is the predicted time series by the generator. The discriminator network $\mathcal{D}$ and the generator network $\mathcal{G}$ are trained jointly with Adam [10]. The overall training algorithm is illustrated in Algorithm 1.

5 Experiments

5.1 Datasets And Evaluation Metrics

We use five public datasets: `electricity`[5], `traffic`[6], `wind`[7], `solar`[8], `M4-Hourly` [17] for our evaluation. The `electricity` dataset is an hourly time series of electricity consumption of 370 customers. The `traffic` dataset consists of 963 car lanes hourly occupancy rates (range $[0, 1)$) of San Francisco bay area freeways. The `wind` contains hourly estimates of an area's energy potential for 1986-2015. The `solar` contains solar power production records from January to August in 2006. The `M4-Hourly` contains 414 hourly time series from M4 competition [17], which aims to forecast time series in the testing set. Following [25], we generate multiple training windows by varying the start point from the original time series with fixed history length t_0 and forecasting horizon τ. Table 1 describes the statistics of these datasets. The input covariate x is a combination of time-dependent features (e.g., a set of dummy variables like day-of-the-week, hour-of-the-day, etc) and time-independent features (e.g., car lane id, station id, etc).

Table 1: Dataset Statistics,where F is the frequency of time series, T is the length of time series, D is number of variables

Datasets	F	T	D
Electricity	hourly	32,304	370
Traffic	hourly	4,049	963
wind	daily	10,957	28
solar	hourly	4,832	137
M4-Hourly	hourly	748/1,008	414

As is in the previous paper [25], we use normalized quantile loss (ρ-risk) to evaluate the quality of the forecasting. For a given collection of time series $\mathbf{y}$ and the corresponding prediction $\hat{\mathbf{y}}$, the ρ-risk for $\rho \in (0, 1)$ is defined as:

$$Q_{\rho}(\mathbf{y}, \hat{\mathbf{y}}) = 2\frac{\sum_{i,t} P_{\rho}(y_{i,t}, \hat{y}_{i,t})}{\sum_{i,t} |y_{i,t}|} \tag{8}$$

To be consistent with previous works [14, 25], we mainly report the results for $\rho = 0.5$ and 0.9 which abbreviated as Q_{50} and Q_{90}, respectively.

[5] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams 20112014

[6] https://archive.ics.uci.edu/ml/datasets/PEMS-SF

[7] https://www.kaggle.com/sohier/30-years-of-european-wind-generation

[8] https://www.nrel.gov/grid/solar-power-data.html

Table 2: Q_{50} loss results for the short-term (24-hour ahead, abbreviated as 1d) forecast and long-term (7d ahead) forecast scenarios.

Dataset	Reported Metrics From [14]								
	ARIMA	**ETS**	**TRMF**	**DeepAR**	**DSSM**	**ConvTrans**	**T**	**ST**	**AST**
$\texttt{elect}_{1d}$	0.154 ± 0.039	0.101 ± 0.022	0.084 ± 0.008	0.075 ± 0.010	0.083 ± 0.009	0.059 ± 0.008	0.064 ± 0.007	0.055 ± 0.005	$\mathbf{0.042} \pm 0.007$
$\texttt{elect}_{7d}$	0.283 ± 0.056	0.121 ± 0.029	0.087 ± 0.011	0.082 ± 0.015	0.085 ± 0.013	0.070 ± 0.011	0.070 ± 0.011	0.058 ± 0.009	$\mathbf{0.057} \pm 0.010$
$\texttt{traffic}_{1d}$	0.223 ± 0.049	0.236 ± 0.036	0.186 ± 0.021	0.161 ± 0.031	0.167 ± 0.033	0.122 ± 0.025	0.120 ± 0.022	0.109 ± 0.019	$\mathbf{0.093} \pm 0.010$
$\texttt{traffic}_{7d}$	0.492 ± 0.079	0.509 ± 0.102	0.202 ± 0.041	0.179 ± 0.035	0.168 ± 0.035	0.139 ± 0.029	0.129 ± 0.021	0.127 ± 0.023	$\mathbf{0.125} \pm 0.019$

Table 3: Q_{90} loss results for the short-term (24-hour ahead, abbreviated as 1d) forecast and long-term (7d ahead) forecast scenarios.TRMF outputs points predictions, so we only report Q_{50} results.

Dataset	Reported Metrics From [14]								
	ARIMA	**ETS**	**TRMF**	**DeepAR**	**DSSM**	**ConvTrans**	**T**	**ST**	**AST**
$\texttt{elect}_{1d}$	0.102 ± 0.019	0.077 ± 0.010	-	0.040 ± 0.007	0.056 ± 0.011	0.028 ± 0.005	0.036 ± 0.007	0.029 ± 0.005	$\mathbf{0.025} \pm 0.006$
$\texttt{elect}_{7d}$	0.109 ± 0.020	0.101 ± 0.019	-	0.053 ± 0.010	0.052 ± 0.012	0.044 ± 0.009	0.039 ± 0.007	0.041 ± 0.008	$\mathbf{0.036} \pm 0.006$
$\texttt{traffic}_{1d}$	0.137 ± 0.024	0.148 ± 0.028	-	0.099 ± 0.019	0.113 ± 0.023	0.081 ± 0.014	0.087 ± 0.015	0.084 ± 0.013	$\mathbf{0.068} \pm 0.010$
$\texttt{traffic}_{7d}$	0.280 ± 0.052	0.529 ± 0.14	-	0.105 ± 0.019	0.114 ± 0.023	0.094 ± 0.021	0.096 ± 0.009	0.088 ± 0.008	$\mathbf{0.086} \pm 0.007$

Table 4: Performance summary (The reported results are in the format of Q_{50}/Q_{90}) of using different α on `electricity` and `traffic`.

	sparsemax	1.5-entmax	softmax		sparsemax	1.5-entmax	softmax
$\texttt{elect}_{1d}$	0.040/0.026	**0.042/0.025**	0.040/0.027	$\texttt{traffic}_{1d}$	0.123/0.090	**0.093/0.068**	0.120/0.086

5.2 Accuracy Comparison

To assess the performance of our model, we compare our model on the `electricity` and `traffic` datasets with two classical forecasting methods, i.e., **ARIMA**, **ETS**, which are effective time series forecasting models, and several latest deep learning based models, including **DeepAR**[25], which is a state-of-the-art RNN-based probability forecasting model, **DSSM**[24], which is an RNN-based state space model, **TRMF**[32], which is a recent matrix factorization method, and **ConvTrans**[14], which is a transformer-based model.

For the short-term forecasting, we evaluate rolling-day forecasts for seven days after training and the length of conditioning range is set to one week of time series (168 observations per series). For the long-term forecasting, we directly forecast for 7 days and the length of conditioning range is set to two weeks of time series (336 observations per series). The experimental results for `electricity` and `traffic` datasets are summarized in Table 2 and 3 respectively, where **T** and **ST** represent Vanilla Transformer and sparse Transformer respectively, and **AST** is the abbreviation of Adversary Sparse Transformer. Finally, our model archives the best results over all the datasets.

From the results, we can draw the following conclusions. **ARIMA** and **ETS** perform the worst possibly because of inability to use the important covariates feature and model shared patterns across time series. **TRMF** can only estimate the mean and outperform **ARIMA** and **ETS**. Besides, the other four methods based on RNN or Transformer achieve better results than the first two. This indicates deep neural networks can learn shared seasonal patterns from several series and the covariate feature help to improve the performance. **ST** outperform Vanilla Transformer in most experiments (other than the Q_{90} of long-term forecasting), indicating that the dependencies among the steps of time series are sparse. **AST** outperforms **ST** in all experiments indicating that the sequence-level regularization is essential for long-horizon time series forecasting. Specifically, from Figure 1, we can see that AST have a better ability to model the sharp peaks of time series than Vanilla Transformer and Sparse Transformer. Especially, compared with **ConvTrans**, **AST** improves the Q_{50} for 26.4% on average and improves the Q_{90} for 15.6% on average, indicating that adversarial training and sparse Transformer can better model time series. In a word, the results show that the AST can alleviate the limitations of the above methods.

5.3 Ablation Study

The effect of α-entmax To validate the effect of different α of the attention map on the prediction performance, we test AST on `electricity` with $\alpha \in \{1, 1.5, 2\}$ while fixing other settings, and show the results in Figure 3. From the results, we can observe that AST with $1.5 - entmax$ can allocate proper attention to the essential ones, thus helping the model to learn the patterns more easily.

Table 6: Q_{50}/Q_{90}-loss of datasets with various granularities, where $\diamond$ denotes a result from [14].

Model	`wind`	`solar`	`M4-hourly`
DeepAR	$0.288^\diamond$/$\mathbf{0.113}^\diamond$	$0.222^\diamond/0.093^\diamond$	$0.090^\diamond/0.030^\diamond$
DeepState	$0.392^\diamond/0.189^\diamond$	$1.126^\diamond/0.517^\diamond$	$0.044^\diamond$/$\mathbf{0.026}^\diamond$
TRMF	$0.311^\diamond/-$	$0.241^\diamond/-$	-/-
Ours	**0.272**/0.124	**0.155/0.054**	**0.042**/0.028

Table 7: Performance summary (Q_{50}/Q_{90}) of encoder-decoder based Transformer vs auto-regressive decoder-only based Transformer.

		T	ST	AST		T	ST	AST
enc-dec	`elect`$_{1d}$	0.064/0.036	0.055/0.029	**0.042/0.025**	`traffic`$_{1d}$	0.120/0.087	0.109/0.084	**0.093/0.068**
dec-only		0.053/0.055	0.052/0.026	**0.048/0.024**		0.154/0.136	0.092/0.079	**0.096/0.069**

The test performance of different α on the `electricity` and `traffic` is shown in Table 4, which shows that when $\alpha = 1.5$, Q_{50} and Q_{90} achieve the minimal values for both datasets. The results indicate that allocating too sparse attention to learn the underlying relationship will deteriorate the performance, and allocating too dense attention wastes attention on irrelevant steps which cause a poor performance.

Attention Weight Density Analysis To visualize the impact of sparse attention maps, we compare the empirical attention weight density (the average number of tokens receiving non-zero attention) within each module of Transformer. Figure 4 (in supplementary) shows that compared with softmax, entmax tends to be sparse and allocate higher scores to important items which contribute to the interpretability of the model.

DeepAR Equipped with Adversarial training To further explore the effects of adversarial training to time series forecasting networks, we attach the adversarial training to the DeepAR [25] network, which is a state-of-the-art auto-regressive LSTM-based time series forecasting network.

Similar to our Adversarial Sparse Transformer, we add a discriminator on the top of the DeepAR network, and train the DeepAR and the discriminator iteratively. The results in Table 5 show that adversarial training improves the performance of DeepAR, proving that adversarial training can alleviate the error accumulation and the shortcomings of using a specific loss function.

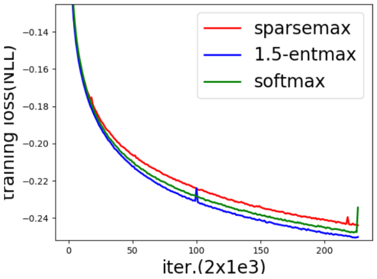


Figure 3: Illustration of performance with different activation functions in multi-head attention.

5.4 Further Exploration

We further explore the performance of our model on the `wind`, `solar` and `M4-Hourly` with various granularities(e.g. different frequencies like daily). The prediction lengths are 8, 30, and 24 respectively. The Q_{50} and Q_{90} test losses are reported in Table 6 for all the methods. AST significantly outperforms other methods on 4/6 tasks except the Q_{90} of `wind` and `M4-Hourly`.

Furthermore, we compare the performance of the auto-regressive decoder-only Transformer [23] and the vanilla encoder-decoder Transformer in Table 7. The results indicate that the encoder-decoder based Transformer outperforms the auto-regressive decoder-only Transformer.

Table 5: Performance of DeepAR compared with DeepAR equipped with adversarial training

Model		Q_{50}	Q_{90}
DeepAR	`elect`$_{1d}$	0.075	0.040
	`traffic`$_{1d}$	0.161	0.099
DeepAR+Adv	`elect`$_{1d}$	**0.067**	**0.035**
	`traffic`$_{1d}$	**0.155**	**0.089**

6 Conclusion

In this work, we present Adversarial Sparse Transformer(AST), a novel Transformer-based model for time series forecasting. By adversarial learning, we improve the contiguous and fidelity at the sequence level. We further propose Sparse Transformer to improve the ability to pay more attention on relevant steps in time series. Extensive experiments on a series of real-world time series datasets have demonstrated the effectiveness of AST for both short-term and long-term time series forecasting. According to the experimental results, we argue that (1) adversarial training can improve the time series forecasting from a global perspective, and (2) the dependencies among steps of time series have some tend of sparsity.

Broader Impact

Our proposed new Time series forecasting model– AST improves the time series forecasting by adversarial training and Sparse Transformer, and it achieves impressive performance. AST can better model time series data and alleviate the error accumulation in inference. We believe our work will inspire the related research of time series forecasting. Our work will benefit the application of time series forecasting such as business and industrial decision-making. And we think there are no one will be disadvantaged by our work. Our model does not take advantage of data bias, it is general and scalable.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (61972219), the National Natural Science Foundation of Guangdong Province (2018A030313422), the RD Program of Shenzhen (JCYJ20190813174403598,SGDX20190918101201696), the National Key Research and Development Program of China (2018YFB1800600, 2018YFB1800204).

Reference

[1] Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of recurrent neural networks for short term load forecasting. *arXiv preprint arXiv:1705.04378*, 2017.

[2] Mathieu Blondel, André FT Martins, and Vlad Niculae. Learning classifiers with fenchel-young losses: Generalized entropies, margins, and algorithms. *arXiv preprint arXiv:1805.09717*, 2018.

[3] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[4] Yagmur Gizem Cinar, Hamid Mirisaee, Parantapa Goswami, Eric Gaussier, Ali Aït-Bachir, and Vadim Strijov. Position-based content attention for time series forecasting with sequence-to-sequence rnns. In *International Conference on Neural Information Processing*, pages 533–544. Springer, 2017.

[5] Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*, 2019.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Rob Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[8] Rob J Hyndman, Roman A Ahmed, George Athanasopoulos, and Han Lin Shang. Optimal combination forecasts for hierarchical time series. *Computational Statistics & Data Analysis*, 55(9):2579–2589, 2011.

[9] Sarthak Jain and Byron C. Wallace. Attention is not explanation, 2019.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104. ACM, 2018.

[12] Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[13] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

[14] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, 2019.

[15] Chenghao Liu, Steven CH Hoi, Peilin Zhao, and Jianling Sun. Online arima algorithms for time series prediction. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[16] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders, 2015.

[17] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

[18] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623, 2016.

[19] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[20] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[21] Ben Peters, Vlad Niculae, and André FT Martins. Sparse sequence-to-sequence models. *arXiv preprint arXiv:1905.05702*, 2019.

[22] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. A dual-stage attention-based recurrent neural network for time series prediction, 2017.

[23] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[24] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794, 2018.

[25] David Salinas, Valentin Flunkert, and Jan Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks, 2017.

[26] Huan Song, Deepta Rajan, Jayaraman J Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[28] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster, 2017.

[29] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[30] Ke Xu, Yifan Zhang, Deheng Ye, Peilin Zhao, and Mingkui Tan. Relation-aware transformer for portfolio policy learning. In *IJCAI*, 2020.

[31] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 5509–5519, 2019.

[32] Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in neural information processing systems*, pages 847–855, 2016.