A Appendix: Derivation of loss function

Recall that we specify a sequence of noisy samples $\boldsymbol{x}^g_{N^g_*+1}, \dots, \boldsymbol{x}^g_N$ by applying the forward process on $\boldsymbol{x}^g$. The superscript in N^g_* is suppressed for notation brevity. Suppose the coarse-grained data $\boldsymbol{x}^g_{N_*} \sim q(\boldsymbol{x}^g)$, where the subscript notation N_* indicates that the observed $\boldsymbol{x}^g$ is treated as a sample from the distribution. (In the diffusion model, the subscript is typically denoted as 0, but we start with N_* to simplify the derivation).

$$
\begin{aligned}
\log p_\theta(\boldsymbol{x}^g_{N_*}) &\leq -\log p_\theta(\boldsymbol{x}^g_{N_*}) + D_{\mathrm{KL}}(q(\boldsymbol{x}^g_{(N_*+1):N}|\boldsymbol{x}^g_{N_*}) \| p_\theta(\boldsymbol{x}^g_{(N_*+1):N}|\boldsymbol{x}^g_{N_*})) \\
&= -\log p_\theta(\boldsymbol{x}^g_{N_*}) + \mathbb{E}_{\boldsymbol{x}^g_{(N_*+1):N} \sim q(\boldsymbol{x}^g_{(N_*+1):N}|\boldsymbol{x}^g_{N_*})}\Big[\log \frac{q(\boldsymbol{x}^g_{(N_*+1):N}|\boldsymbol{x}^g_{N_*})}{p_\theta(\boldsymbol{x}^g_{N_*:N})/p_\theta(\boldsymbol{x}^g_{N_*})}\Big] \\
&= -\log p_\theta(\boldsymbol{x}^g_{N_*}) + \mathbb{E}_q\Big[\log \frac{q(\boldsymbol{x}^g_{(N_*+1):N}|\boldsymbol{x}^g_{N_*})}{p_\theta(\boldsymbol{x}^g_{N_*:N})} + \log p_\theta(\boldsymbol{x}^g_{N_*})\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\boldsymbol{x}^g_{(N_*+1):N}|\boldsymbol{x}^g_{N_*})}{p_\theta(\boldsymbol{x}^g_{N_*:N})}\Big] \qquad (12)
\end{aligned}
$$

Then, the training objective can be performed by optimizing the usual variational lower bound shown below:

$$
L_{\mathrm{VLB}} = \mathbb{E}_{q(\boldsymbol{x}^g_{N_*:N})}\Big[\log \frac{q(\boldsymbol{x}^g_{(N_*+1):N}|\boldsymbol{x}^g_{N_*})}{p_\theta(\boldsymbol{x}^g_{N_*:N})}\Big] \geq -\mathbb{E}_{q(\boldsymbol{x}^g_{N_*})} \log p_\theta(\boldsymbol{x}^g_{N_*}) \qquad (13)
$$

It is obvious that the objective L_{VLB} is equivalent to the that of diffusion model in Ho et al. (2020) when employing diffusion models on $\boldsymbol{x}^g$ with $N - N_*$ steps. The forward process is defined as $q(\boldsymbol{x}^g_{(N_*+1):N}|\boldsymbol{x}^g_{N_*}) = \prod_{n=N_*}^{N} q(\boldsymbol{x}^g_n|\boldsymbol{x}^g_{n-1})$, where $q(\boldsymbol{x}^g_n|\boldsymbol{x}^g_{n-1}) := \mathcal{N}(\sqrt{1-\beta^g_n}\boldsymbol{x}^g_{n-1}, \beta^g_n \boldsymbol{I})$. The $\{\beta^g_n\}_{n=N_*}^N$ share values with the variance schedule $\{\beta^1_n\}_{n=1}^N$ of the finest-grained data from index N_*. And, the reverse process is defined by the θ-parameterized trajectory. Then following the same technique in Ho et al. (2020), the L_{VLB} can reduce to the usual loss of diffusion models.

B Appendix: Experiments

B.1 Benchmark experiments

The results of the benchmark experiments, evaluated based on the metrics $\mathrm{NRMSE}_{\mathrm{sum}}$ and $\mathrm{NMAE}_{\mathrm{sum}}$, are presented in Table 4 and Table 5 respectively. In the experiments, we include four extra baseline models for a more comprehensive comparison: TimeDiff (Shen & Kwok, 2023), D^3VAE (Li et al., 2022), PatchTST (Nie et al., 2022), and AutoFormer (Wu et al., 2021).

Table 4: Comparison of $\mathrm{NRMSE}_{\mathrm{sum}}$ (smaller is better) of models on six real-world datasets. The reported mean and standard error are obtained from 10 re-training and evaluation independent runs.

Method	Solar	Electricity	Traffic	KDD-cup	Taxi	Wikipedia
Vec-LSTM ind-scaling	$0.9952_{\pm 0.0077}$	$0.1439_{\pm 0.0228}$	$0.1451_{\pm 0.0248}$	$0.4461_{\pm 0.1833}$	$0.6398_{\pm 0.0390}$	$0.1618_{\pm 0.0162}$
GP-Scaling	$0.9004_{\pm 0.0095}$	$0.0811_{\pm 0.0062}$	$0.1469_{\pm 0.0181}$	$0.3445_{\pm 0.0621}$	$0.3598_{\pm 0.0285}$	$0.1710_{\pm 0.1006}$
GP-Copula	$0.8279_{\pm 0.0053}$	$0.0512_{\pm 0.0009}$	$0.1282_{\pm 0.0033}$	$\mathbf{0.2605}_{\pm 0.0227}$	$0.3125_{\pm 0.0113}$	$0.0930_{\pm 0.0076}$
Autoformer	$0.7046_{\pm 0.0000}$	$0.0475_{\pm 0.0000}$	$0.0951_{\pm 0.0000}$	$0.8984_{\pm 0.0000}$	$0.3498_{\pm 0.0000}$	$0.1052_{\pm 0.0000}$
PatchTST	$0.7270_{\pm 0.0000}$	$0.0474_{\pm 0.0000}$	$0.1897_{\pm 0.0000}$	$0.5137_{\pm 0.0000}$	$0.3690_{\pm 0.0000}$	$0.0915_{\pm 0.0000}$
D^3VAE	$0.7472_{\pm 0.0508}$	$0.1640_{\pm 0.0928}$	$0.4722_{\pm 0.1197}$	$0.5628_{\pm 0.0419}$	$0.7624_{\pm 0.5598}$	$2.2094_{\pm 2.1646}$
TimeDiff	$1.5985_{\pm 0.0359}$	$0.3714_{\pm 0.0073}$	$0.5520_{\pm 0.0087}$	$0.4955_{\pm 0.0147}$	$0.5479_{\pm 0.0084}$	$0.1412_{\pm 0.0099}$
TimeGrad	$0.6953_{\pm 0.0845}$	$0.0348_{\pm 0.0057}$	$0.0653_{\pm 0.0244}$	$0.4092_{\pm 0.1332}$	$0.2365_{\pm 0.0386}$	$0.0870_{\pm 0.0106}$
TACTiS	$0.8532_{\pm 0.0851}$	$0.0427_{\pm 0.0023}$	$0.2270_{\pm 0.0159}$	$0.6513_{\pm 0.1767}$	$0.3387_{\pm 0.0097}$	-
MG-TSD	$\mathbf{0.6178}_{\pm \mathbf{0.0418}}$	$\mathbf{0.0241}_{\pm \mathbf{0.0030}}$	$\mathbf{0.0563}_{\pm \mathbf{0.0230}}$	$0.3001_{\pm 0.0997}$	$\mathbf{0.2334}_{\pm \mathbf{0.0313}}$	$\mathbf{0.0810}_{\pm \mathbf{0.0057}}$

Table 5: Comparison of NMAE_{sum} (smaller is better) of models on six real-world datasets. The reported mean and standard error are obtained from 10 re-training and evaluation independent runs.

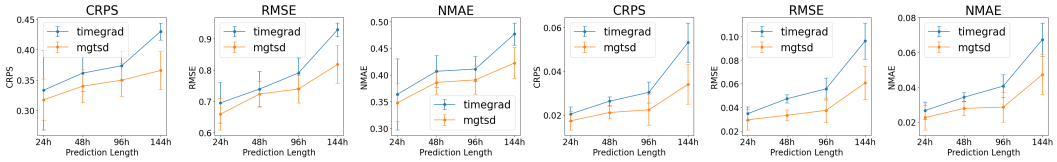
Method	Solar	Electricity	Traffic	KDD-cup	Taxi	Wikipedia
Vec-LSTM ind-scaling	$0.5091_{\pm 0.0027}$	$0.1261_{\pm 0.0211}$	$0.1042_{\pm 0.0228}$	$0.4193_{\pm 0.1902}$	$0.4974_{\pm 0.0351}$	$0.1416_{\pm 0.0180}$
GP-Scaling	$0.4945_{\pm 0.0065}$	$0.0648_{\pm 0.0046}$	$0.0975_{\pm 0.0163}$	$0.2892_{\pm 0.0550}$	$0.2867_{\pm 0.0264}$	$0.1452_{\pm 0.1029}$
GP-Copula	$0.4302_{\pm 0.0046}$	$0.0312_{\pm 0.0007}$	$0.0769_{\pm 0.0022}$	$\mathbf{0.2140}_{\pm 0.0124}$	$0.2390_{\pm 0.0098}$	$0.0659_{\pm 0.0061}$
Autoformer	$0.6368_{\pm 0.0000}$	$0.0388_{\pm 0.0000}$	$0.0684_{\pm 0.0000}$	$0.7658_{\pm 0.0000}$	$0.2652_{\pm 0.0000}$	$0.1239_{\pm 0.0000}$
PatchTST	$0.4351_{\pm 0.0000}$	$0.0350_{\pm 0.0000}$	$0.1219_{\pm 0.0000}$	$0.4497_{\pm 0.0000}$	$0.2887_{\pm 0.0000}$	$0.0625_{\pm 0.0000}$
D^3VAE	$0.4457_{\pm 0.0377}$	$0.1434_{\pm 0.0892}$	$0.3992_{\pm 0.1177}$	$0.4874_{\pm 0.0520}$	$0.6080_{\pm 0.5061}$	$2.0151_{\pm 2.0005}$
TimeDiff	$1.3343_{\pm 0.0305}$	$0.3519_{\pm 0.0075}$	$0.4782_{\pm 0.0058}$	$0.3630_{\pm 0.0127}$	$0.4521_{\pm 0.0102}$	$0.1146_{\pm 0.0106}$
TimeGrad	$0.3694_{\pm 0.0400}$	$0.0266_{\pm 0.0049}$	$0.0410_{\pm 0.0089}$	$0.3614_{\pm 0.1334}$	$0.1365_{\pm 0.0193}$	$0.0631_{\pm 0.008}$
TACTiS	$0.4448_{\pm 0.0313}$	$0.0310_{\pm 0.0015}$	$0.1352_{\pm 0.0159}$	$0.6078_{\pm 0.1718}$	$0.2244_{\pm 0.0036}$	-
MG-TSD	$\mathbf{0.3347}_{\pm \mathbf{0.0220}}$	$\mathbf{0.0178}_{\pm \mathbf{0.0018}}$	$\mathbf{0.0370}_{\pm \mathbf{0.0140}}$	$0.2463_{\pm 0.0865}$	$\mathbf{0.1300}_{\pm \mathbf{0.0150}}$	$\mathbf{0.0601}_{\pm \mathbf{0.0057}}$

B.2 More experiment settings

B.2.1 Performance for long-term forecasting

To evaluate the performance of MG-TSD for long-term forecasting, we maintain a fixed context length of 24 and extend the prediction length to 24, 48, 96, and 144. The results of the datasets `Solar` and `Eelectrity` are displayed in Figure 5.

The results in Figure 5 indicate that MG-TSD performs well for long-time forecasting. The results indicate that as the prediction length increases, the performance of our proposed method stays robust, exhibiting no sudden decline. Furthermore, our method consistently outperforms the competitive baseline. This performance advantage is anticipated to persist in future trends, with no indication of convergence between the approaches.



(a) `Solar`: Performance Evaluation Across Different Prediction Lengths

(b) `Electricity`: Performance Evaluation Across Different Prediction Lengths

Figure 5: Performance evaluation across different prediction horizons for MG-TSD with TimeGrad as the baseline Model. The context length is fixed at 24h and the prediction length is tested at 24h, 48h, 96h, and 144h. The average CRPS, NRMSE, and NMAE metrics are computed for both MG-TSD and the baseline over 10 independent runs, with error bars indicating the corresponding standard deviations.

B.2.2 Time and memory usage of the MG-TSD model during training

Experiments have been conducted to evaluate the time and memory usage of the MG-TSD model during training across various granularities. These experiments were executed using a single A6000 card with 48G memory capacity. The Solar dataset was utilized in this context, with a batch size of 128, an input size of 552, 100 diffusion steps, and 30 epochs.

As illustrated in Figure 6, there is a linear increase in memory consumption with an increase in granularity. A slight surge in training time is also observed. These findings are coherent with the architecture of our model. In particular, each additional granularity results in the introduction of an extra RNN in the Temporal Process Module and an increase in computation within the Guided Diffusion Process Module. As per theoretical expectations, these resource consumptions should exhibit linear growth. The slight increase in training time can be ascribed to the design of the Multi-granularity Data Generator Module which enables parallel forward processes across different granularities, thus promoting acceleration. Moreover, it is pertinent to mention that an excessive increase in granularity may not notably boost the final prediction results, hence the granularity will be kept within a certain range. Therefore, the consumption of memory will not rise indefinitely.

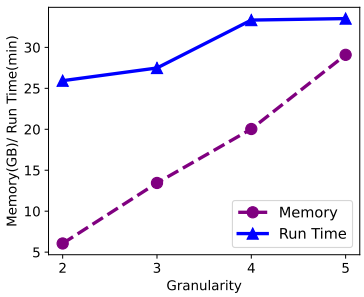
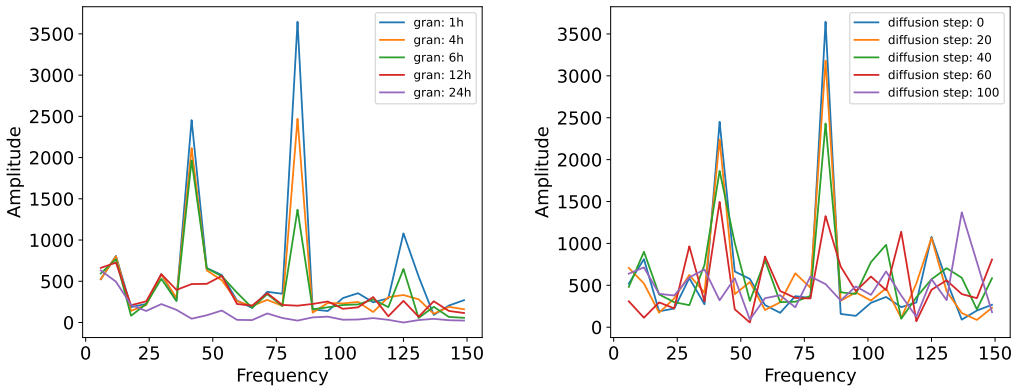


Figure 6: Comparison of Time and Memory Consumption at Different Granularity Levels in MG-TSD Model Training

B.2.3 Variations in the Frequency Domain of Time Series Data: The Impact of Granularity and Denoising Steps

We sampled series from Solar dateset and we conducted a Fast Fourier Transform to extract the seasonality components of the series, as well as the samples of different granularities and corresponding noisy samples along the forward diffusion process.



(a) Different granularities

(b) Different diffusion steps

Figure 7: Variations in the frequency domain of time series data: the impact of granularity and denoising steps.

As depicted in Figure 7(a), as granularity becomes coarser, the components of all outstanding frequencies get lower, while the high-frequency peak (around 125 and 80) diminishes quicker than lower-frequency peak (around 45). Figure 7(b) demonstrates the distribution of frequency components of the same noisy series with gradually ascending forward diffusion steps and the same pattern is observable. This empirical study indicates the connection between the forward diffusion process and the smoothing process from fine-grained data to coarse-grained data, both of which result in losing finer informative features.

C Appendix: Implementation Details

C.1 Benchmark Datasets

For our experiments, we use `Solar`, `Electricity`, `Traffic`, `Taxi`, `KDD-cup` and `Wikipedia` open-source datasets, with their properties listed in Table 6.

The dataset can be obtained through the links below.

(i) `Solar`: `https://www.nrel.gov/grid/solar-power-data.html`

(ii) Electricity: https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014

(iii) Traffic: https://archive.ics.uci.edu/dataset/204/pems+sf

(iv) Taxi: https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

(v) KDD-cup: https://www.kdd.org/kdd2018/kdd-cup

(vi) Wikipedia: https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release/datasets

Name	Frequency	Number of series	Context length	Prediction length	Multi-granularity dictionary
Solar	1 hour	137	24	24	[1 hour, 4 hour, 12 hour, 24hour, 48 hour]
Electricity	1 hour	370	24	24	[1 hour, 4 hour, 12 hour, 24 hour, 48 hour]
Traffic	1 hour	963	24	24	[1 hour, 4 hour, 12 hour, 24 hour, 48 hour]
Taxi	30 min	1214	24	24	[30 min , 2 hour, 6 hour, 12 hour, 24 hour]
KDD-cup	1 hour	270	48	48	[1 hour, 4 hour, 12 hour, 24hour, 48 hour]
Wikipedia	1 day	2000	30	30	[1 day, 4 day, 7 day, 14 day]

Table 6: Detailed information of the datasets used in our benchmark including data frequency and number of times series (dimension), including the information about context length and prediction length and the multi-granularity dictionary utilized in the multivariate time series forecasting task.

C.2 LIBRARIES USED

The **MG-TSD** code in this study is implemented using PyTorch (Paszke et al., 2019). It utilizes the PytorchTS library (Rasul, 2021), which enables convenient integration of PyTorch models with the GluonTS library (Alexandrov et al., 2020b) on which we heavily rely for data preprocessing, model training, and evaluation in our experiments.

The code for the baseline methods is obtained from the following sources.

(i) Vec-LSTM-ind-scaling: models the dynamics via an RNN and outputs the parameters of an independent Gaussian distribution with mean-scaling.
Code: https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release;

(ii) GP-scaling: a model that unrolls an LSTM with scaling on each individual time series before reconstructing the joint distribution via a low-rank Gaussian.
Code: https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release

(iii) GP-Copula: a model that unrolls an LSTM on each individual time series. The joint emission distribution is then represented by a low-rank plus diagonal covariance Gaussian copula.
Code: https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release;

(iv) LSTM-MAF: a model which utilizes LSTM for modeling the temporal conditioning and employs Masked Autoregressive Flow (Papamakarios et al., 2017) for the distribution emission.
Code: https://github.com/zalandoresearch/pytorch-ts/tree/master/pts/model/tempflow

(v) Transformer-MAF: a model which utilizes Transformer (Vaswani et al., 2017) for modeling the temporal conditioning and employs Masked Autoregressive Flow (Papamakarios et al., 2017) for the distribution emission model.
Code: https://github.com/zalandoresearch/pytorch-ts/tree/master/pts/model/transformer_tempflow

(vi) TimeGrad: an auto-regressive model designed for multivariate probabilistic time series forecasting, assisted by an energy-based model.
Code: https://github.com/zalandoresearch/pytorch-ts

(vii) TACTiS: a non-parametric copula model based on transformer architecture.
Code: https://github.com/servicenow/tactis

(viii) D^3VAE: a bidirectional variational auto-encoder(BVAE) equipped with diffusion, denoise, and disentanglement.
Code: `https://github.com/ramber1836/d3vae`.

(ix) TimeDiff: a predictive framework trained by blending hidden contextual elements with future actual outcomes for sample conditioning.
Code: There is no publicly available code; we obtained the code by emailing the author.

(x) Autoformer: redefines the Transformer with a deep decomposition architecture, including sequence decomposition units, self-correlation mechanisms, and encoder-decoders.
Code: `https://github.com/thuml/Autoformer`

(xi) PatchTST: an efficient design of Transformer-based models for multivariate time series forecasting and self-supervised representation learning.
Code: `https://github.com/yuqinie98/PatchTST`

C.3 HYPER-PARAMETER SETTING FOR EACH MODEL

Dataset	Num gran	Gran dict	Share ratio	Loss weight
`Solar` `Electricity` `Traffic` `KDD-cup`	2	[1h,4h] [1h,12h]	[1,0.9] [1,0.8] [1,0.6]	[0.9,0.1]
	3	[1h,4h,12h] [1h,4h,24h]	[1,0.9,0.8] [1,0.8,0.8] [1,0.8,0.6]	[0.8, 0.1, 0.1] [0.9, 0.05, 0.05] [0.85, 0.10, 0.05]
	4	[1h,4h,12h,24h] [1h,4h,12h,48h]	[1,0.9,0.8,0.8] [1,0.9,0.8,0.6] [1,0.8,0.6,0.6] [1,0.8,0.6,0.4]	[0.8, 0.1, 0.05, 0.05] [0.7,0.1,0.1,0.1]
	5	[1h,4h,8h,12h,24h] [1h,4h,12h,24h,48h]	[1,0.9,0.8,0.6,0.6] [1,0.9,0.8,0.6,0.4] [1,0.8,0.6,0.6,0.6] [1,0.8,0.6,0.6,0.4] [1,0.8,0.6,0.4,0.4]	[0.8,0.1,0.05,0.04,0.01] [0.8,0.05,0.05,0.05,0.05] [0.6,0.1,0.1,0.1,0.1]
`Taxi`	2	[30m,2h] [30m,6h] [30m,12h] [30m,24h]	[1,0.8] [1,0.6]	[0.9,0.1]
`Wikipedia`	3	[1d,4d] [1d,7d] [1d,14d]	[1,0.8] [1,0.6]	[0.9,0.1]

Table 7: Tested hyper-parameter values for the **MG-TSD** Model. The reported results in the paper are based on a parameter search within these choices.

D APPENDIX: METRICS

More details about the metrics we adopt can be found in Gluonts documentation (Alexandrov et al., 2020a). We briefly summarize them as below:

$\mathbf{CRPS_{sum}}$: From de Bézenac et al. (2020), CRPS is a univariate strictly proper scoring rule which measures the compatibility of a cumulative distribution function F with an observation $x \in \mathbb{R}$ as

$$\text{CRPS}(F, x) = \int_{\mathbb{R}} (F(y) - \mathbf{1}(x \leq y))^2 \mathrm{d}y$$

where $\boldsymbol{I}\{x \leq y\}$ is the indicator function, which is 1 if $x \leq y$ and 0 otherwise. The CRPS attains the minimum value when the predictive distribution F same as the data distribution. CRPS_{sum} extends CRPS to multivariate time series with a simple modification.

$$\text{CRPS}_{\text{sum}} = \mathbb{E}_t[\text{CRPS}(F^{-1}_{\text{sum}}, \sum_i x^i_t)],$$

where F_{sum}^{-1} is calculated by summing samples across dimensions and then sorted to get quantiles. A smaller CRPS_{sum} indicates better performance.

NMAE: NMAE is a normalized version of the Mean Absolute Error (MAE) that takes into consideration the scale of the target values. The formula for NMAE is as follows:

$$\text{NMAE} = \frac{\text{mean}(|(\hat{Y} - Y)|)}{\text{mean}(|Y|)}$$

Similarly, in this formula, $\hat{Y}$ represents the predicted time series, and Y represents the true target time series. NMAE calculates the average absolute difference between predictions and true values, normalized by the mean absolute magnitude of the target values. A smaller NMAE implies more accurate predictions.

NRMSE: NRMSE is a normalized adaptation of the Root Mean Squared Error (RMSE) that factors in the scale of the target values. The formula for NRMSE is as follows:

$$\text{NRMSE} = \sqrt{\frac{\text{mean}((\hat{Y} - Y)^2)}{\text{mean}(|Y|)}}$$

Here, $\hat{Y}$ represents the predicted time series, and Y represents the true target time series. NRMSE measures the average squared difference between predictions and true values, normalized by the mean absolute magnitude of the target values. A smaller NRMSE indicates more accurate predictions.

E APPENDIX: MORE ILLUSTRATIVE PLOTS

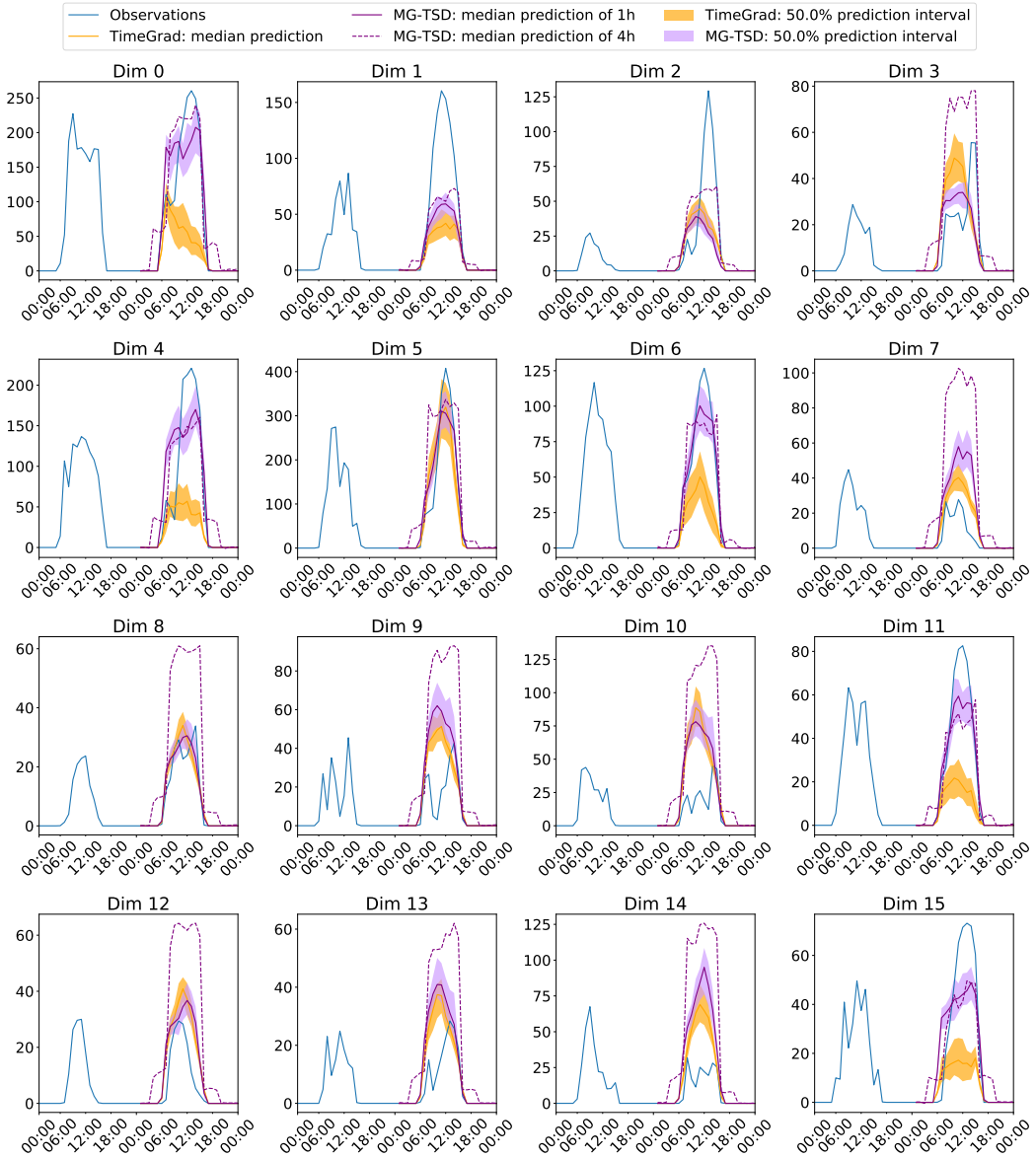


Figure 8: **MG-TSD** and TimeGrad prediction intervals and test set ground-truth for Solar data of some illustrative dimensions of 370 dimensions from first rolling-window.