

BRL Seminar

(2024. 01. 03. Wed)

Diffusion Model with Time Series Data

통합과정 7학기 이승한

Contents

1. Preliminaries: Diffusion Model
2. **Conditional** Time-series Diffusion Model: **TimeGrad, CSDI**
3. **Unconditional** Time-series Diffusion Model: **TSDiff**

Papers

Conditional / Unconditional diffusion model

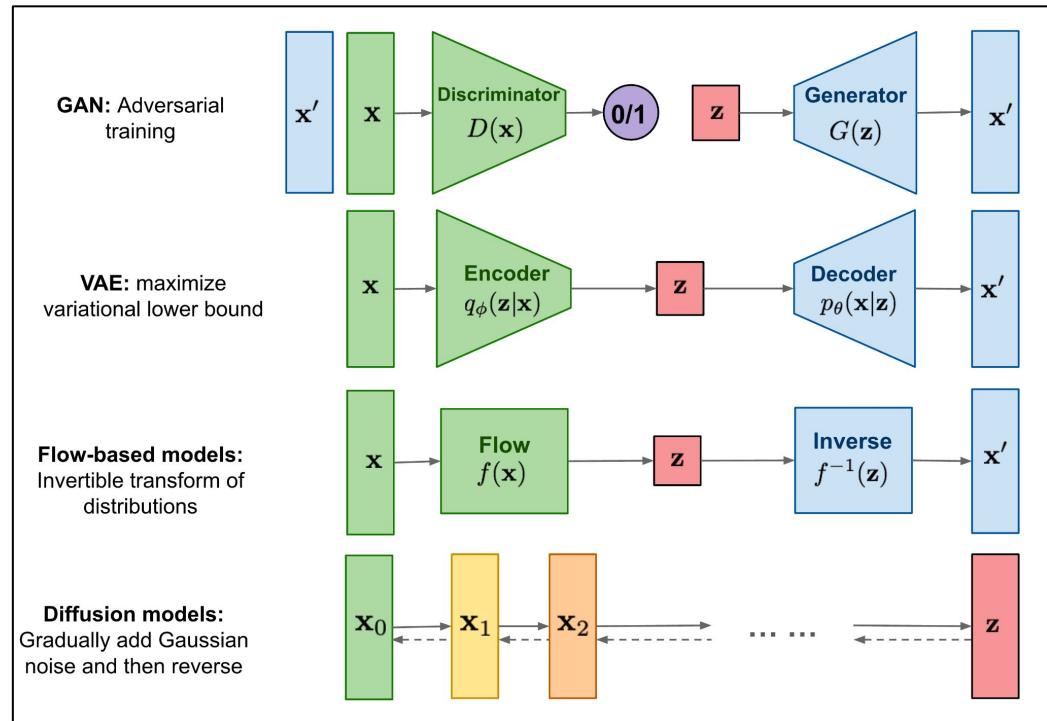
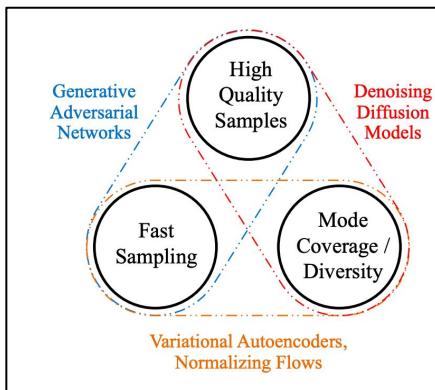
1. **[TimeGrad]** Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting (ICML, 2021)
 - <https://arxiv.org/pdf/2101.12072.pdf>
2. **[CSDI]** Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation (NeurIPS 2021)
 - <https://arxiv.org/pdf/2107.03502.pdf>
3. **[TSDiff]** Non-autoregressive Conditional Diffusion Models for Time Series Prediction (NeurIPS 2023) 

1. Preliminaries: Diffusion Model

1. Preliminaries: Diffusion Model

1-1. Generative Models

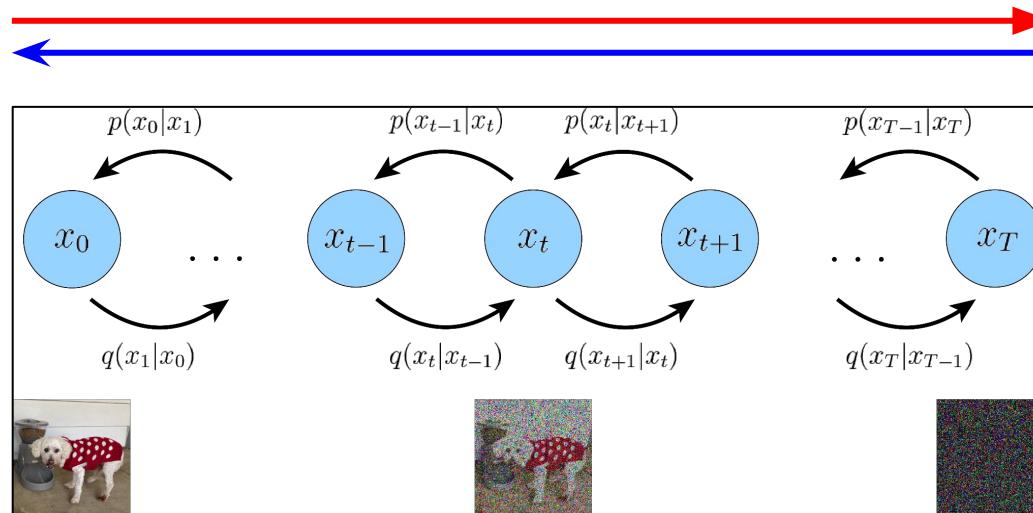
- (1) GAN
- (2) Diffusion
- (3) VAE, Normalizing Flows



1. Preliminaries: Diffusion Model

1-2. Diffusion Model

- **Forward Process:** **Add Noise**
- **Backward Process:** **Remove Noise**



1. Preliminaries: Diffusion Model

1-2. Diffusion Model

- **Forward Process:** **Add** Noise

$$q(x_t | x_{t-1}) = N\left(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I\right)$$

- **Backward Process:** **Remove** Noise

$$p_\theta(x_{t-1} | x_t) = N\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right)$$

β_t controls the strength of the noise → **Noise Scheduling**

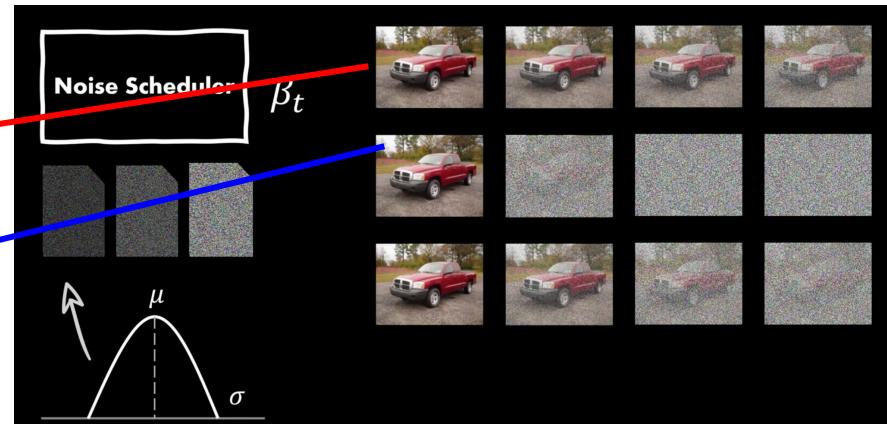
ex) DDPM: Linear Scheduling $T = 1000, \beta_0 = 0.0001, \beta_{1000} = 0.02$

1. Preliminaries: Diffusion Model

1-3. Noise Scheduling

Too **SMALL** noise!

Too **BIG** noise!



Linear scheduler

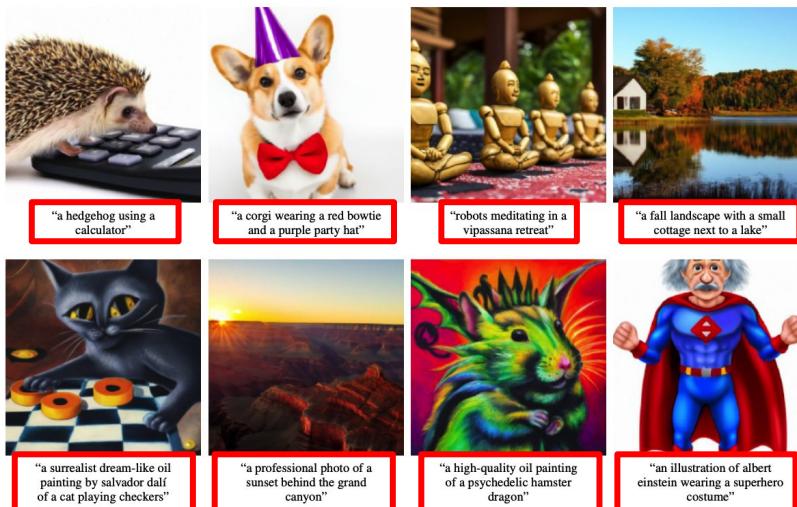


Linear (top) vs Cosine (bottom)

1. Preliminaries: Diffusion Model

1-4. Conditional Diffusion Model

- Unconditional Diffusion Model: “*Draw a picture*”
- **Conditional** Diffusion Model: “*Draw a picture **of a dog***”



Anything can be a condition!

1. Preliminaries: Diffusion Model

1-4. Conditional Diffusion Model

How to guide the diffusion model under certain condition?

- (1) **Classifier** guidance (**CG**): **w/** classifier
- (2) **Classifier-free** guidance (**CFG**): **w/o** classifier

1. Preliminaries: Diffusion Model

1-4. Conditional Diffusion Model

How to guide the diffusion model under certain condition?

- (1) **Classifier** guidance (CG): w/ classifier

$$\begin{aligned} \frac{\nabla \log p(\mathbf{x}_t | y)}{\text{Conditional Score}} &= \nabla \log \left(\frac{p(\mathbf{x}_t) p(y | \mathbf{x}_t)}{p(y)} \right) \\ (\text{Score of } \mathbf{x}_t, \text{ under condition } y) &= \nabla \log p(\mathbf{x}_t) + \nabla \log p(y | \mathbf{x}_t) - \nabla \log p(y) \\ &= \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}} + \gamma \underbrace{\nabla \log p(y | \mathbf{x}_t)}_{\text{adversarial gradient}} \xrightarrow{\text{Classifier}} \text{Classifier} \end{aligned}$$

1. Preliminaries: Diffusion Model

1-4. Conditional Diffusion Model

How to guide the diffusion model under certain condition?

$$\begin{aligned}\nabla \log p(\mathbf{x}_t | y) &= \nabla \log \left(\frac{p(\mathbf{x}_t) p(y | \mathbf{x}_t)}{p(y)} \right) \\ &= \nabla \log p(\mathbf{x}_t) + \nabla \log p(y | \mathbf{x}_t) - \nabla \log p(y) \\ &= \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}} + \gamma \underbrace{\nabla \log p(y | \mathbf{x}_t)}_{\text{adversarial gradient}}\end{aligned}$$

- (1) **Classifier** guidance (CG): w/ classifier

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s .

Input: class label y , gradient scale s

$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$

for all t from T to 1 **do**

$\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$

$x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s \Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$

end for

return x_0

Guidance



1. Preliminaries: Diffusion Model

1-4. Conditional Diffusion Model

How to guide the diffusion model under certain condition?

- (2) **Classifier-free** guidance (**CFG**): **w/o** classifier

$$\nabla \log p(\mathbf{x}_t | y) = \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}} + \gamma \underbrace{\nabla \log p(y | \mathbf{x}_t)}_{\text{adversarial gradient}}$$

1. Preliminaries: Diffusion Model

1-4. Conditional Diffusion Model

How to guide the diffusion model under certain condition?

- (2) **Classifier-free** guidance (**CFG**): **w/o** classifier

$$\nabla \log p(\mathbf{x}_t | y) = \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}} + \gamma \underbrace{\nabla \log p(y | \mathbf{x}_t)}_{\text{adversarial gradient}} \rightarrow \gamma \underbrace{\nabla \log p(y | \mathbf{x}_t)}_{\text{adversarial gradient}} = \nabla \log p(\mathbf{x}_t | y) - \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}}$$

1. Preliminaries: Diffusion Model

1-4. Conditional Diffusion Model

How to guide the diffusion model under certain condition?

- (2) **Classifier-free** guidance (CFG): w/o classifier

$$\nabla \log p(\mathbf{x}_t | y) = \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}} + \gamma \underbrace{\nabla \log p(y | \mathbf{x}_t)}_{\text{adversarial gradient}}$$



$$\gamma \underbrace{\nabla \log p(y | \mathbf{x}_t)}_{\text{adversarial gradient}} = \nabla \log p(\mathbf{x}_t | y) - \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}}$$



1. Preliminaries: Diffusion Model

1-4. Conditional Diffusion Model

How to guide the diffusion model under certain condition?

- (2) **Classifier-free** guidance (CFG): w/o classifier

$$\nabla \log p(\mathbf{x}_t | y) = \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}} + \gamma \underbrace{\nabla \log p(y | \mathbf{x}_t)}_{\text{adversarial gradient}} \quad \xrightarrow{\hspace{1cm}} \quad \gamma \underbrace{\nabla \log p(y | \mathbf{x}_t)}_{\text{adversarial gradient}} = \nabla \log p(\mathbf{x}_t | y) - \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}}$$

$$\begin{aligned} \nabla \log p(\mathbf{x}_t | y) &= \nabla \log p(\mathbf{x}_t) + \gamma (\nabla \log p(\mathbf{x}_t | y) - \nabla \log p(\mathbf{x}_t)) \\ &= \nabla \log p(\mathbf{x}_t) + \gamma \nabla \log p(\mathbf{x}_t | y) - \gamma \nabla \log p(\mathbf{x}_t) \\ &= \underbrace{\gamma \nabla \log p(\mathbf{x}_t | y)}_{\text{conditional score}} + \underbrace{(1 - \gamma) \nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}} \end{aligned}$$

No need for an extra classifier!

1. Preliminaries: Diffusion Model

1-4. Conditional Diffusion Model

How to guide the diffusion model under certain condition?

- (2) **Classifier-free** guidance (**CFG**): w/o classifier

Algorithm 1 Joint training a diffusion model with classifier-free guidance

Require: p_{uncond} : probability of unconditional training

- ```

1: repeat
2: $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ \triangleright Sample data with conditioning from the dataset
3: $\mathbf{c} \leftarrow \emptyset$ with probability p_{uncond} \triangleright Randomly discard conditioning to train unconditionally
4: $\lambda \sim p(\lambda)$ \triangleright Sample log SNR value
5: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
6: $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \epsilon$ \triangleright Corrupt data to the sampled log SNR value
7: Take gradient step on $\nabla_\theta \| \epsilon_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \epsilon \|^2$ \triangleright Optimization of denoising model
8: until converged

```

## Integrate condition inside the diffusion model!!

## 2. Conditional Time-series Diffusion Model: TimeGrad, CSDI

## 2. Conditional Time-series Diffusion Model

### Conditional Time-series Diffusion Model

- **TimeGrad (ICML 2021)**: TS Forecasting
- **CSDI (NeurIPS 2021)**: TS Imputation

## 2. Conditional Time-series Diffusion Model

### 2-1. TimeGrad (ICML 2021)

- Diffusion model for **TS forecasting**
- Conditional diffusion model
  - “condition = past information”

$$\prod_{t=t_0}^T p_\theta(\mathbf{x}_t^0 \mid \mathbf{h}_{t-1})$$

=

$$\mathbf{h}_t = \text{RNN}_\theta(\text{concat}(\mathbf{x}_t^0, \mathbf{c}_t), \mathbf{h}_{t-1})$$

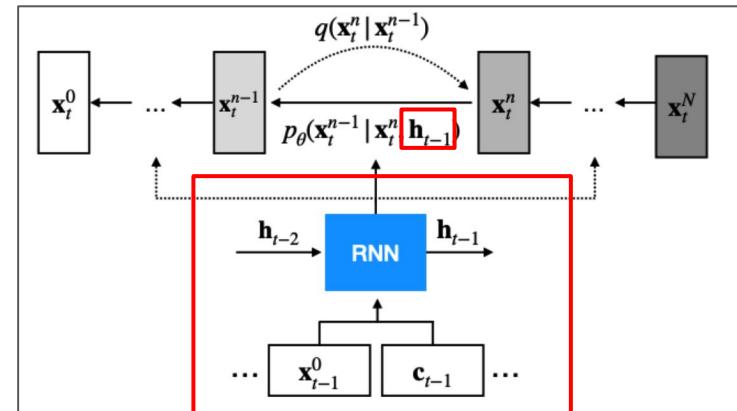
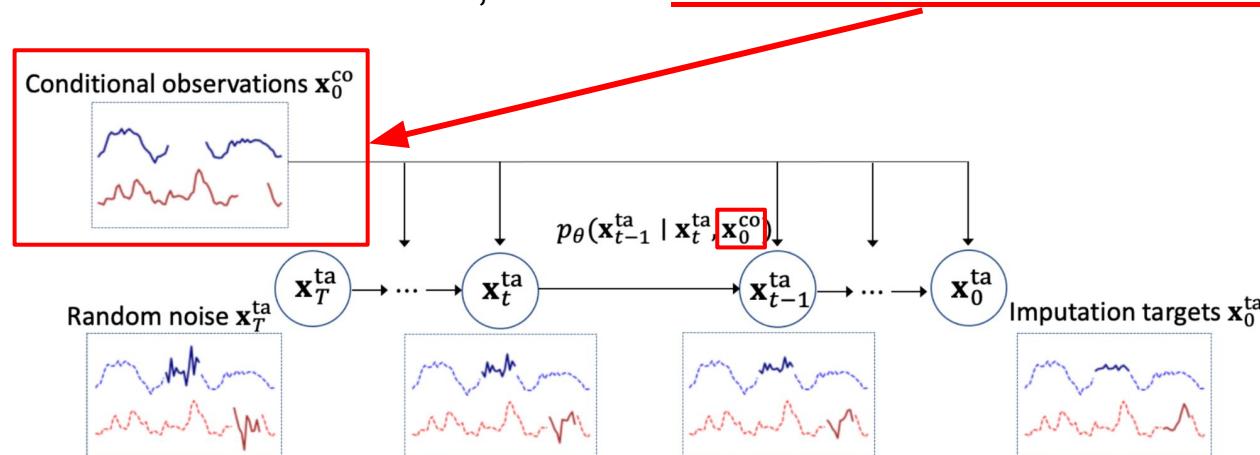


Figure 1. TimeGrad schematic: an RNN *conditioned* diffusion probabilistic model at some time  $t - 1$  depicting the fixed forward process that adds Gaussian noise and the learned reverse processes.

## 2. Conditional Time-series Diffusion Model

### 2-2. CSDI (NeurIPS 2021)

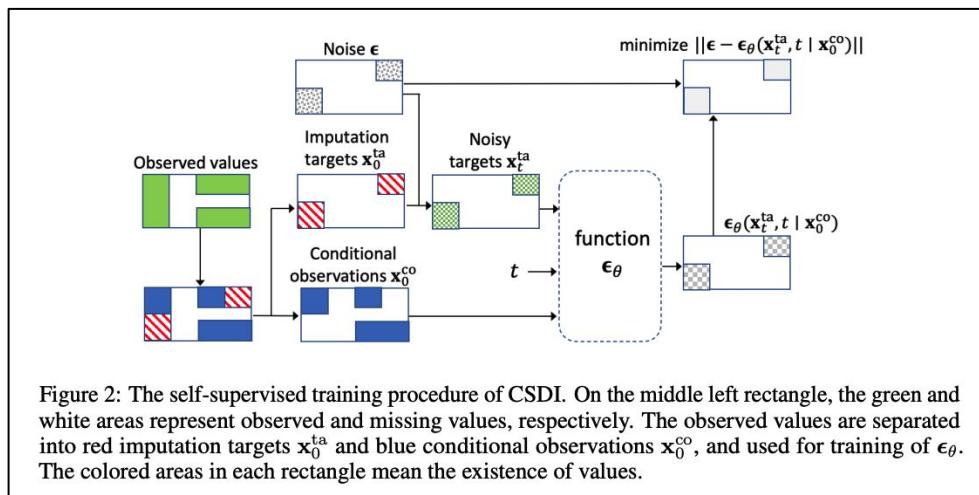
- Diffusion model for **TS imputation**
- Conditional diffusion model, where “condition = observed values”



## 2. Conditional Time-series Diffusion Model

### 2-2. CSDI (NeurIPS 2021)

- Inspired by **masked modeling**



**Masked certain portion of data for training!**

- (1) **Observed (= Train)**
  - (1-1) unmasked (= Train X)
  - (1-2) masked (= Train Y)
- (2) **Unobserved (= Test)**

### 3. Unconditional Time-series Diffusion Model: **TSDiff**

### 3. Unconditional Time-series Diffusion Model

#### TSDiff (NeurIPS 2023)

- (Previous works) **Conditional model**
  - condition depends on the task! ( i.e. imputation, forecasting ... )
- **Task-agnostic & Unconditional** diffusion model for TS
- **Self-guidance mechanism**
  - enables conditioning for downstream tasks “**during inference**”
  - does not require auxiliary network or altering the training procedure

### 3. Unconditional Time-series Diffusion Model

#### TSDiff (NeurIPS 2023)

- Three tasks
  - (1) **Forecasting**
  - (2) **Refinement**
    - refine the predictions of base forecasters efficiently
  - (3) **Synthetic data generation**
    - train downstream forecasters using synthetic samples generated from TSDiff

### 3. Unconditional Time-series Diffusion Model

#### TSDiff (NeurIPS 2023)

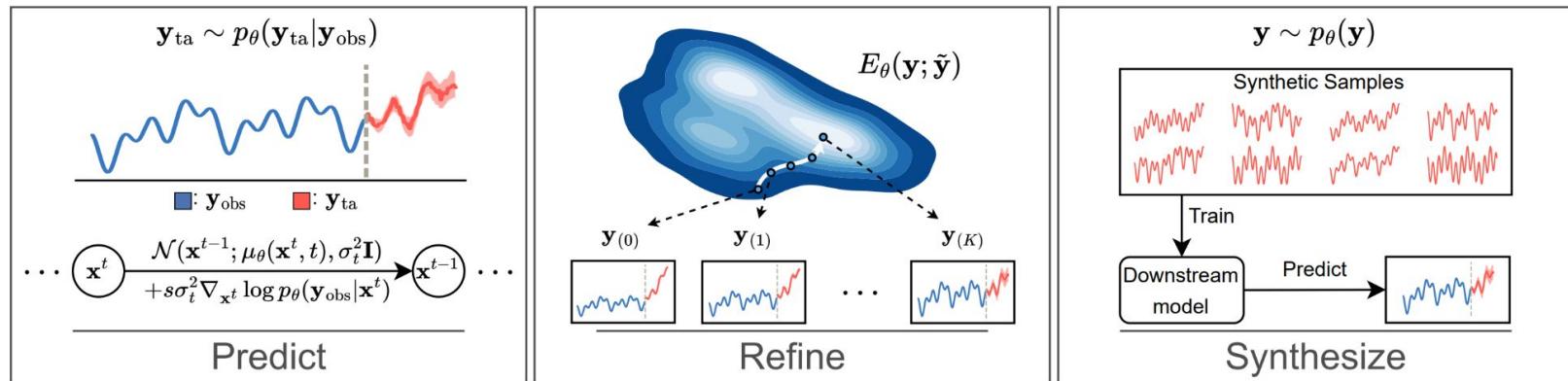


Figure 1: An overview of TSDiff's use cases. **Predict:** By utilizing observation self-guidance, TSDiff can be conditioned during inference to perform predictive tasks such as forecasting (see Sec. 3.1). **Refine:** Predictions of base forecasters can be improved by leveraging the implicit probability density of TSDiff (see Sec. 3.2). **Synthesize:** Realistic samples generated by TSDiff can be used to train downstream forecasters achieving good performance on real test data (see Sec. 4.3).

$x_0$  = 시계열

$x_0 \rightarrow t$ 번 노이즈를 부과  $\rightarrow x_t$

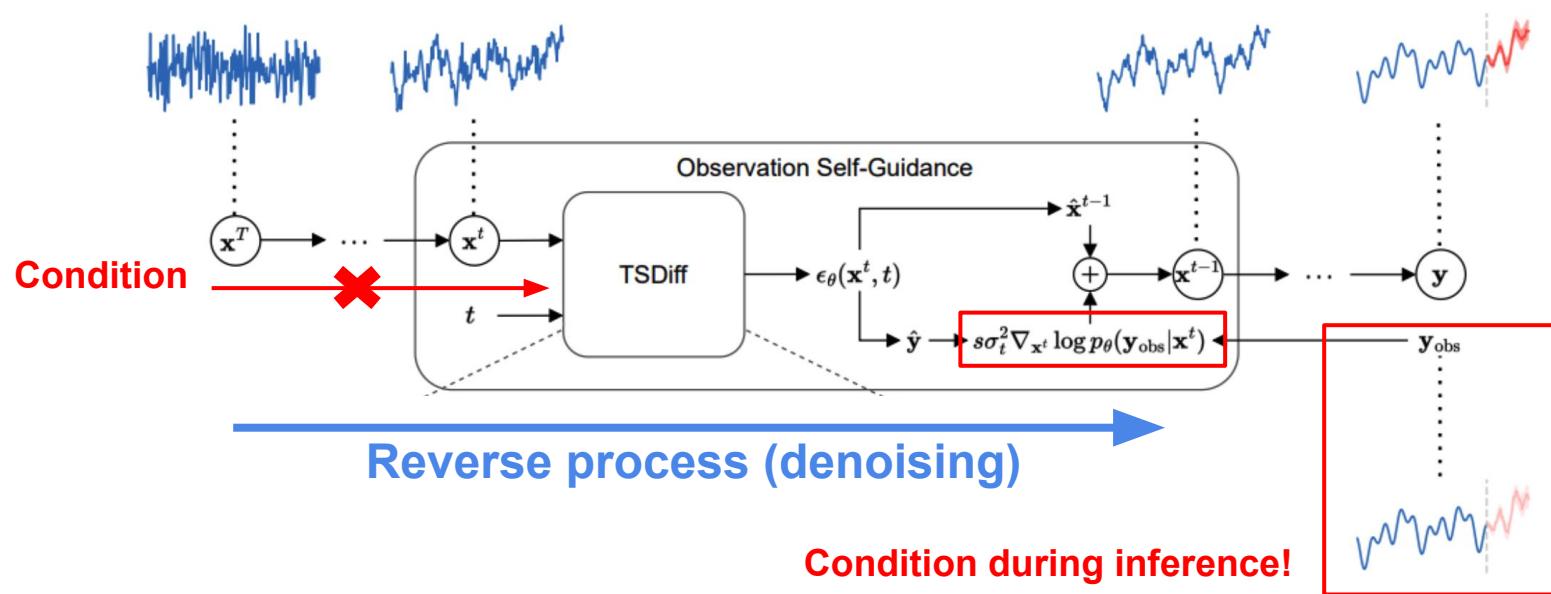
$x_t \rightarrow 1$ 번 노이즈를 부과 (=eps)  $\rightarrow x_{\{t+1\}}$

$f(x_{\{t+1\}}, t+1) = \text{eps}$

### 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

#### 1) Self-Guidance



# 3. Unconditional Time-series Diffusion Model

## TSDiff (NeurIPS 2023)

### 1) Self-Guidance

Seek to train a single unconditional generative model,  $p_\theta(\mathbf{y})$   
& Condition it during inference to draw samples  $p_\theta(\mathbf{y}_{\text{ta}} \mid \mathbf{y}_{\text{obs}})$ .

#### Algorithm 1 Observation Self-Guidance

**Input:** observation  $\mathbf{y}_{\text{obs}}$ , scale  $s$   
 $\mathbf{x}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$   
**for**  $t = T$  to 1 **do**  
     $\mathbf{x}^{t-1} \sim \mathcal{N}(\mu_\theta(\mathbf{x}^t, t), \mathbf{I}) + s\sigma_t^2 \nabla_{\mathbf{x}^t} \log p(\mathbf{y}_{\text{obs}} | \mathbf{x}^t)$   
**end for**

### 1. Preliminaries: Diffusion Model

#### 1-4. Conditional Diffusion Model

How to guide the diffusion model under certain condition?

- (1) **Classifier** guidance (**CG**): w/ classifier

$$\begin{aligned}\nabla \log p(\mathbf{x}_t \mid y) &= \nabla \log \left( \frac{p(\mathbf{x}_t) p(y \mid \mathbf{x}_t)}{p(y)} \right) \\ &= \nabla \log p(\mathbf{x}_t) + \nabla \log p(y \mid \mathbf{x}_t) - \nabla \log p(y) \\ &= \underbrace{\nabla \log p(\mathbf{x}_t)}_{\text{unconditional score}} + \gamma \underbrace{\nabla \log p(y \mid \mathbf{x}_t)}_{\text{adversarial gradient}}\end{aligned}$$

**Algorithm 1** Classifier guided diffusion sampling, given a diffusion model  $(\mu_\theta(x_t), \Sigma_\theta(x_t))$ , classifier  $p_\phi(y|x_t)$ , and gradient scale  $s$ .

Input: class label  $y$ , gradient scale  $s$   
 $x_T \leftarrow$  sample from  $\mathcal{N}(0, \mathbf{I})$   
**for all**  $t$  from  $T$  to 1 **do**  
     $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$   
     $x_{t-1} \leftarrow$  sample from  $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$   
**end for**  
**return**  $x_0$

Guidance

# 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

## 1) Self-Guidance

Seek to train a single unconditional generative model,  $p_\theta(\mathbf{y})$

& Condition it during inference to draw samples  $p_\theta(\mathbf{y}_{\text{ta}} \mid \mathbf{y}_{\text{obs}})$ .

---

### Algorithm 1 Observation Self-Guidance

---

**Input:** observation  $\mathbf{y}_{\text{obs}}$ , scale  $s$   
 $\mathbf{x}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$   
**for**  $t = T$  to 1 **do**  
     $\mathbf{x}^{t-1} \sim \mathcal{N}(\mu_\theta(\mathbf{x}^t, t), \mathbf{I}) + s\sigma_t^2 \nabla_{\mathbf{x}^t} \log p(\mathbf{y}_{\text{obs}} | \mathbf{x}^t)$   
**end for**

---

### (1) Mean Square guidance

$$p_\theta(\mathbf{y}_{\text{obs}} \mid \mathbf{x}^t) = \mathcal{N}(\mathbf{y}_{\text{obs}} \mid f_\theta(\mathbf{x}^t, t), \mathbf{I})$$

### (2) Quantile guidance

$$p_\theta(\mathbf{y}_{\text{obs}} \mid \mathbf{x}^t) = \frac{1}{Z} \cdot \exp \left( -\frac{1}{b} \max \{ \kappa \cdot (\mathbf{y}_{\text{obs}} - [f_\theta(\mathbf{x}^t, t)]), (\kappa - 1) \cdot (\mathbf{y}_{\text{obs}} - [f_\theta(\mathbf{x}^t, t)]) \} \right)$$

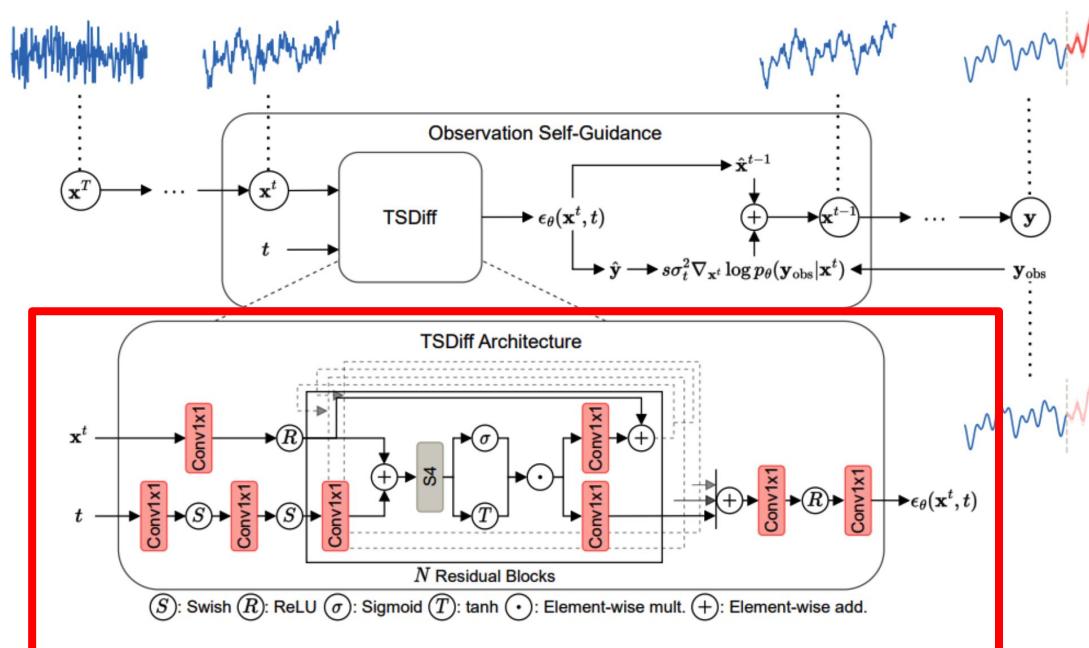
\* Probabilistic forecasts are often evaluated using quantile-based metrics

$$\hat{\mathbf{y}} = f_\theta(\mathbf{x}^t, t) = \frac{\mathbf{x}^t - \sqrt{(1-\bar{\alpha}_t)} \epsilon_\theta(\mathbf{x}^t, t)}{\sqrt{\bar{\alpha}_t}}$$

### 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

#### 2) Backbone

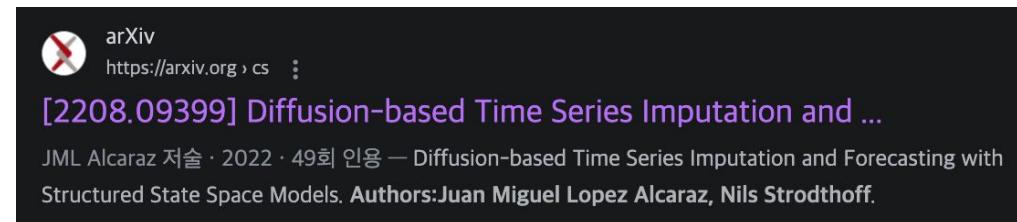
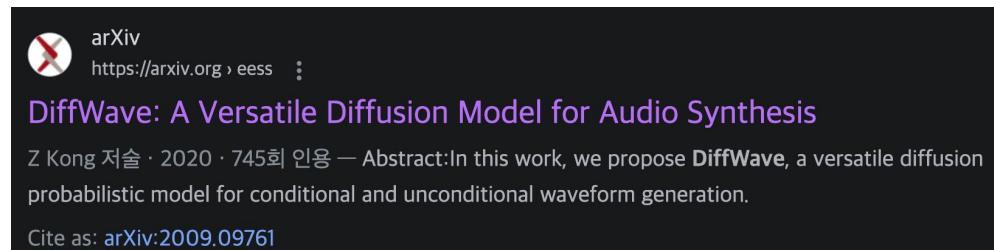


### 3. Unconditional Time-series Diffusion Model

#### TSDiff (NeurIPS 2023)

##### 2) Backbone

- (1) DiffWave →
- (2) SSSD →
- (3) TimeDiff →



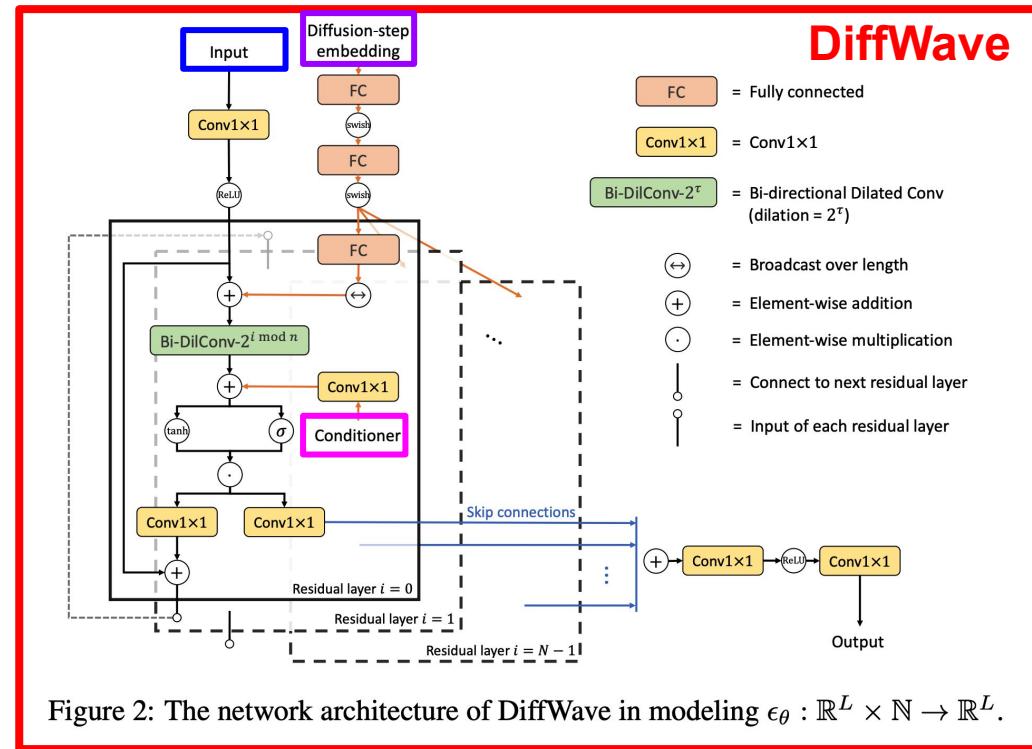
### 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

#### 2) Backbone

- (1) DiffWave
- (2) SSSD
- (3) TimeDiff

$$\epsilon_{\theta}(\mathbf{x}^t, t, c)$$



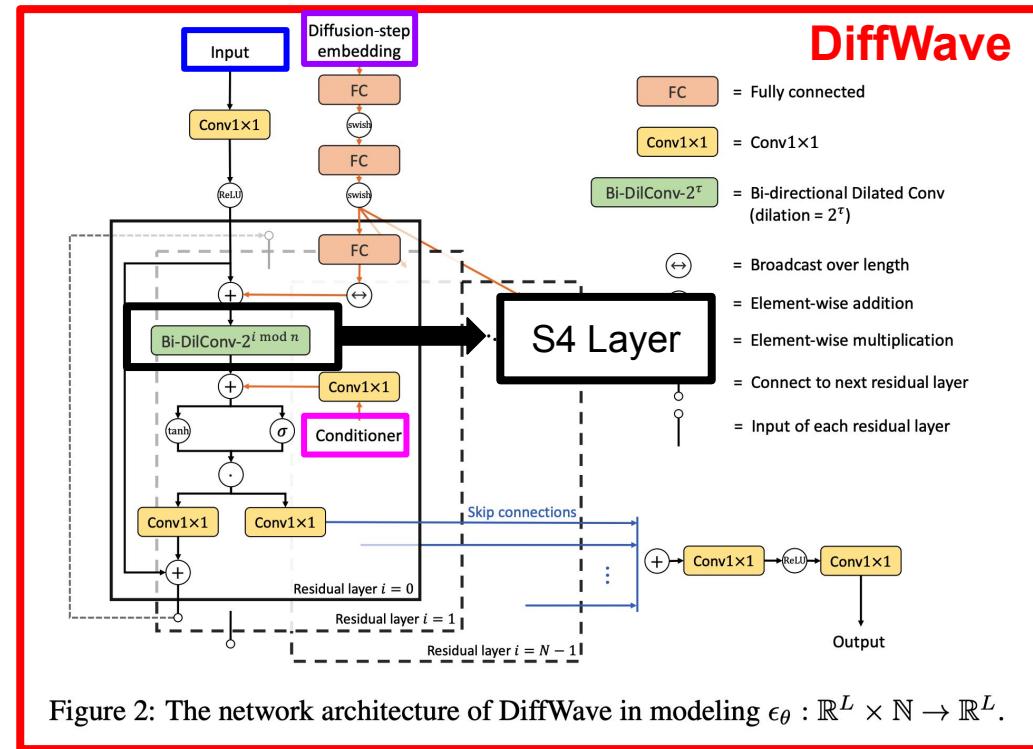
### 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

#### 2) Backbone

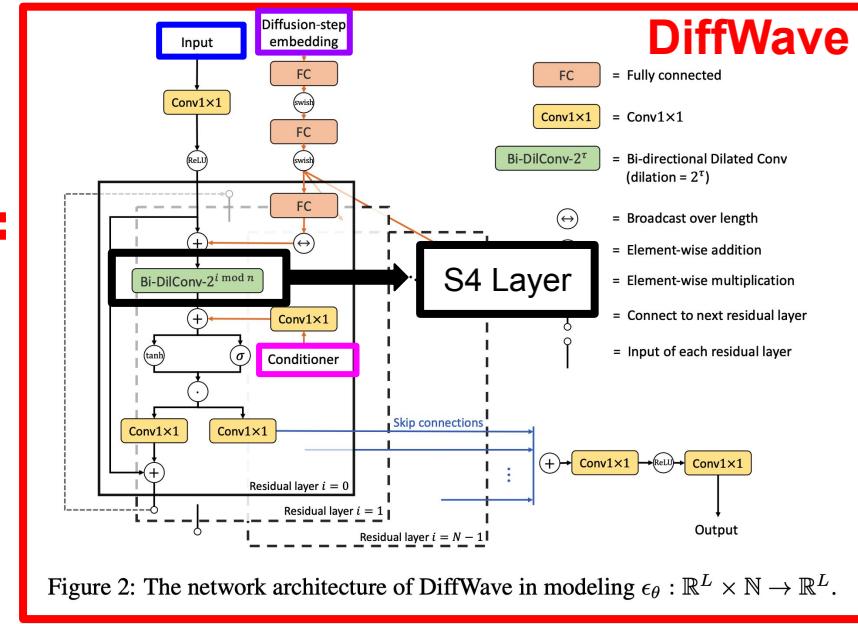
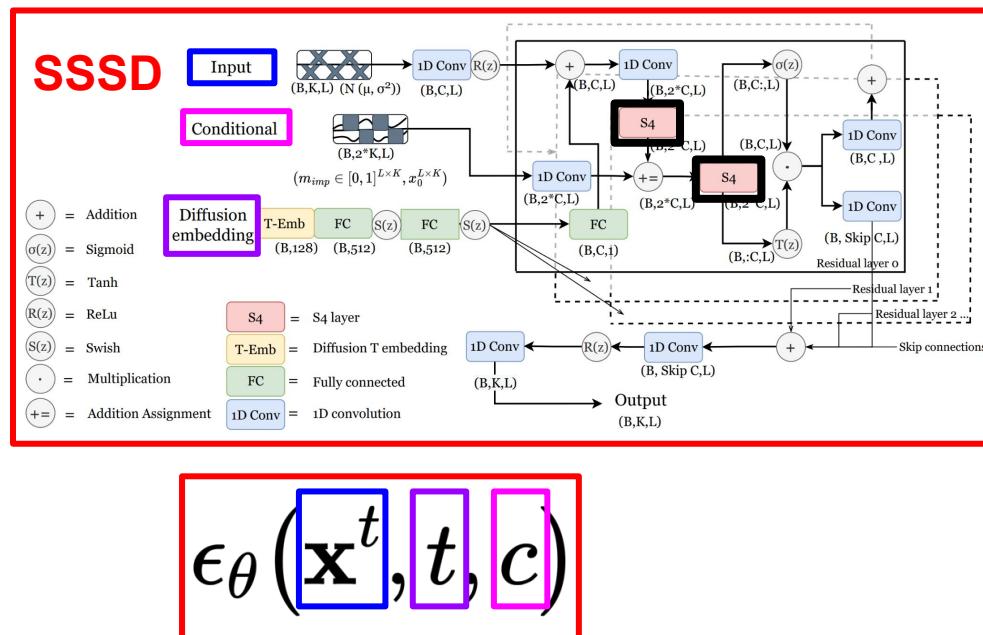
- (1) DiffWave
- (2) SSSD
- (3) TimeDiff

$$\epsilon_{\theta}(\mathbf{x}^t, t, c)$$



# 3. Unconditional Time-series Diffusion Model

## TSDiff (NeurIPS 2023)



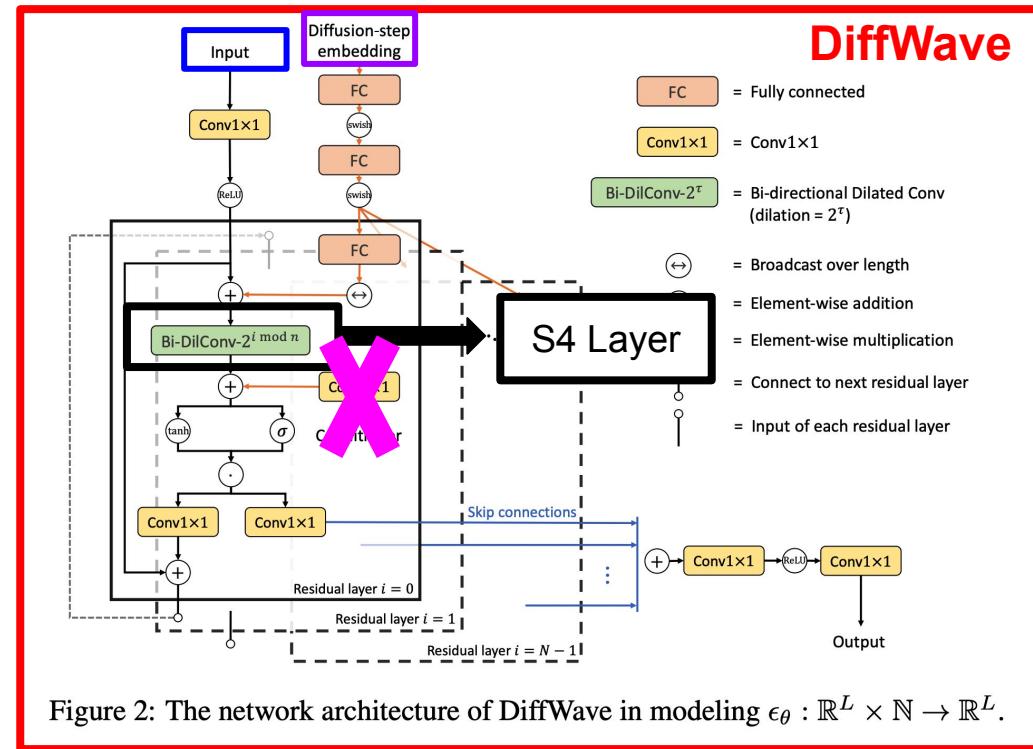
### 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

#### 2) Backbone

- (1) DiffWave
- (2) SSSD
- (3) TimeDiff

$$\epsilon_{\theta}(\mathbf{x}^t, t, \mathbf{X})$$



### 3. Unconditional Time-series Diffusion Model

#### TSDiff (NeurIPS 2023)

TSDiff

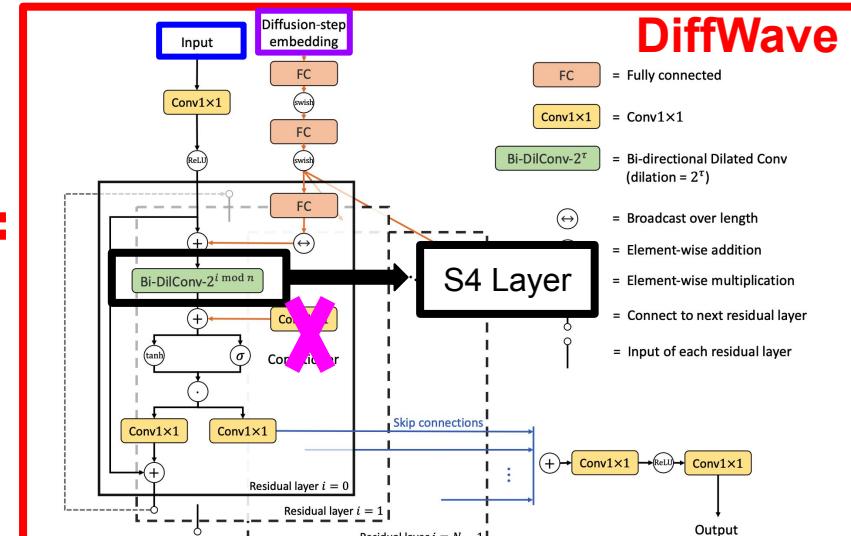
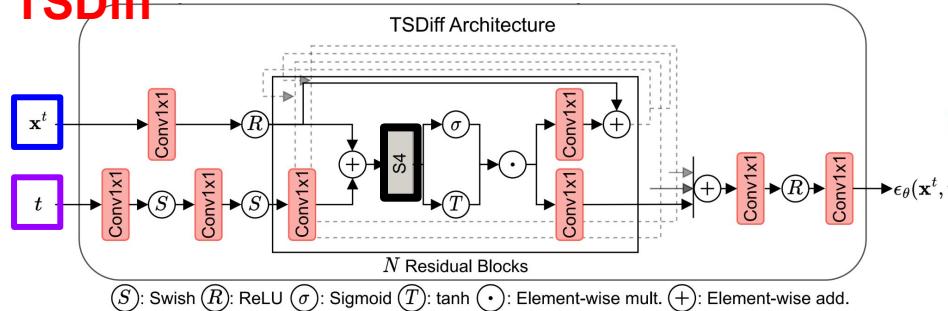
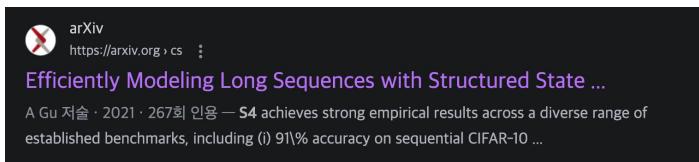


Figure 2: The network architecture of DiffWave in modeling  $\epsilon_\theta : \mathbb{R}^L \times \mathbb{N} \rightarrow \mathbb{R}^L$ .

# 3. Unconditional Time-series Diffusion Model

## TSDiff (NeurIPS 2023)

### 2) Backbone: S4 Layer



#### Algorithm 1 S4 CONVOLUTION KERNEL (SKETCH)

**Input:** S4 parameters  $\Lambda, P, Q, B, C \in \mathbb{C}^N$  and step size  $\Delta$   
**Output:** SSM convolution kernel  $\bar{K} = \mathcal{K}_L(\bar{A}, \bar{B}, \bar{C})$  for  $A = \Lambda - PQ^*$  (equation (5))

- 1:  $\tilde{C} \leftarrow (\bar{I} - \bar{A}^L)^* \bar{C}$  ▷ Truncate SSM generating function (SSMGF) to length  $L$
- 2:  $\begin{bmatrix} k_{00}(\omega) & k_{01}(\omega) \\ k_{10}(\omega) & k_{11}(\omega) \end{bmatrix} \leftarrow [\tilde{C} \; Q]^* \left( \frac{2}{\Delta + \omega} - \Lambda \right)^{-1} [B \; P]$  ▷ Black-box Cauchy kerne
- 3:  $\hat{K}(\omega) \leftarrow \frac{2}{1+\omega} [k_{00}(\omega) - k_{01}(\omega)(1 + k_{11}(\omega))^{-1} k_{10}(\omega)]$  ▷ Woodbury Identity
- 4:  $\hat{K} = \{\hat{K}(\omega) : \omega = \exp(2\pi i \frac{k}{L})\}$  ▷ Evaluate SSMGF at all roots of unity  $\omega \in \Omega_L$
- 5:  $\bar{K} \leftarrow \text{iFFT}(\hat{K})$  ▷ Inverse Fourier Transform

S4 = Model SSM (State-Space Model) with NN

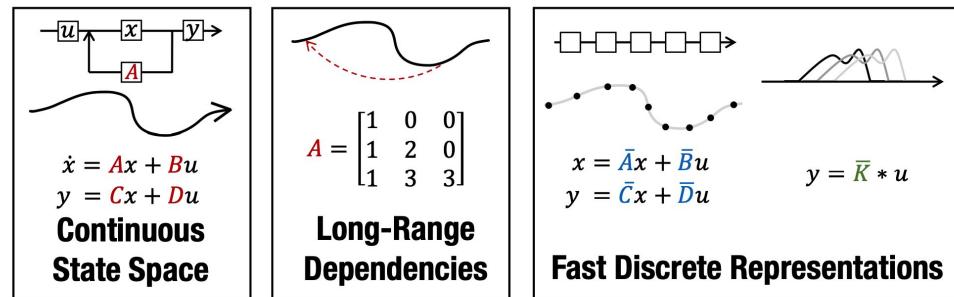


Figure 1: (Left) State Space Models (SSM) parameterized by matrices  $A, B, C, D$  map an input signal  $u(t)$  to output  $y(t)$  through a latent state  $x(t)$ . (Center) Recent theory on continuous-time memorization derives special  $A$  matrices that allow SSMs to capture LRDs mathematically and empirically. (Right) SSMs can be computed either as a recurrence (left) or convolution (right). However, materializing these conceptual views requires utilizing different representations of its parameters (red, blue, green) which are very expensive to compute. S4 introduces a novel parameterization that efficiently swaps between these representations, allowing it to handle a wide range of tasks, be efficient at both training and inference, and excel at long sequences.

# 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

## 3) Prediction Refinement

Goal: Refine the predictions of base forecasters

- Agnostic to the type of base forecaster
- Only assumes access to forecasts generated by them

How?

***Iteratively refine*** the initial forecasts, using the **implicit density** learned by diffusion model  
(serves as a prior)

### 3. Unconditional Time-series Diffusion Model

**TSDiff (NeurIPS 2023)**

#### 3) Prediction Refinement

Two interpretations of refinement

- (a) Sampling from an energy function
- (b) Maximizing the likelihood to find the most likely sequence

### 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

#### 3) Prediction Refinement

##### (a) Sampling from an energy function

- Goal: draw samples from  $p(\mathbf{y}_{\text{ta}} \mid \mathbf{y}_{\text{obs}})$
- **Base forecaster  $g$** 
  - Sample forecast from  $g$ :  $g(\mathbf{y}_{\text{obs}})$  .... **initial guess** of a sample from  $p(\mathbf{y}_{\text{ta}} \mid \mathbf{y}_{\text{obs}})$   
→ **Improve this initial guess!**
- Refinement = Sampling from the regularized **energy-based model (EBM)**

$$E_{\theta}(\mathbf{y}; \tilde{\mathbf{y}}) = -\log p_{\theta}(\mathbf{y}) + \lambda \underbrace{\mathcal{R}(\mathbf{y}, \tilde{\mathbf{y}})}_{\text{Regularizer TS obtained by combining } \mathbf{y}_{\text{obs}} \text{ and } g(\mathbf{y}_{\text{obs}})}$$

Design the energy function s.t. **LOW** energy is assigned to samples that are **LIKELY** under the diffusion model

### 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

#### 3) Prediction Refinement

##### (a) Sampling from an energy function

- Use overdamped **Langevin Monte Carlo (LMC)** to sample from this EBM

$$\mathbf{y}_{(i+1)} = \mathbf{y}_{(i)} - \eta \nabla_{\mathbf{y}_{(i)}} E_\theta(\mathbf{y}_{(i)}; \tilde{\mathbf{y}}) + \sqrt{2\eta\gamma} \xi_i \quad \text{and} \quad \xi_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

### 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

#### 3) Prediction Refinement

##### (b) Maximizing the likelihood to find the most likely sequence

- Refinement = **regularized optimization** of finding the most likely TS that satisfies certain constraints

$$\arg \min_{\mathbf{y}} [-\log p_{\theta}(\mathbf{y}) + \lambda \mathcal{R}(\mathbf{y}, \tilde{\mathbf{y}})]$$

# 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

## 4) Experiments

Goal: Investigate whether “unconditional” TS diffusion model can be employed for downstream tasks

Three tasks

- **(Predict)** Probabilistic forecast (feat. **Self-guidance**)
- **(Refine)** Prediction refinement of base forecasters (feat. **probability density** learned by TSDiff)
- **(Synthesize)** Prediction results of downstream forecasters (feat. **synthetic samples** generated by TSDiff)

# 3. Unconditional Time-series Diffusion Model

## TSDiff (NeurIPS 2023)

### 4) Experiments

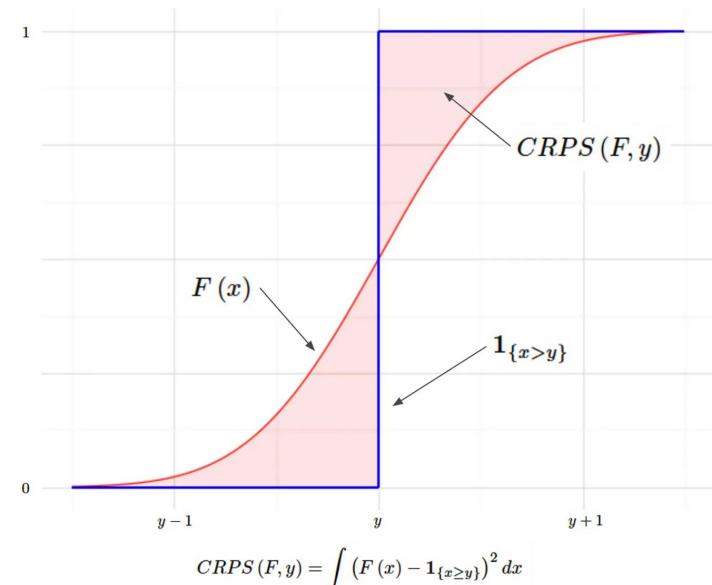
**Dataset:** 8 univariate TS from different domains

**Metric:** CRPS (Continuous ranked probability score)

- approximate CRPS by normalized average quantile loss using 100 sample paths

Table 5: Overview of the benchmark datasets used in our experiments.

| Dataset     | GluonTS Name                 | Train Size | Test Size | Domain         | Freq. | Median Seq. Length | Context Length | Prediction Length |
|-------------|------------------------------|------------|-----------|----------------|-------|--------------------|----------------|-------------------|
| Solar       | solar_nips                   | 137        | 959       | $\mathbb{R}^+$ | H     | 7009               | 336            | 24                |
| Electricity | electricity_nips             | 370        | 2590      | $\mathbb{R}^+$ | H     | 5833               | 336            | 24                |
| Traffic     | traffic_nips                 | 963        | 6741      | (0, 1)         | H     | 4001               | 336            | 24                |
| Exchange    | exchange_rate_nips           | 8          | 40        | $\mathbb{R}^+$ | D     | 6071               | 360            | 30                |
| M4          | m4_hourly                    | 414        | 414       | N              | H     | 960                | 312            | 48                |
| KDDCup      | kdd_cup_2018_without_missing | 270        | 270       | N              | H     | 10850              | 312            | 48                |
| UberTLC     | uber_tlc_hourly              | 262        | 262       | N              | H     | 4320               | 336            | 24                |
| Wikipedia   | wiki2000_nips                | 2000       | 10000     | N              | D     | 792                | 360            | 30                |



Visualization of the CRPS. The predicted distribution is marked in red, and the ground truth's degenerate distribution is marked in blue. The CRPS is the (squared) area trapped between the two CDFs. Image by author.

<https://towardsdatascience.com/crps-a-scoring-function-for-bayesian-machine-learning-models-dd55a7a337a8>

# 3. Unconditional Time-series Diffusion Model

## TSDiff (NeurIPS 2023)

### 4) Experiments

(Predict) Probabilistic forecast (feat. Self-guidance)

#### (a) Standard Forecasting

Table 1: Forecasting results on eight benchmark datasets. The best and second best models have been shown as **bold** and underlined, respectively.

| Method         | Solar              | Electricity        | Traffic            | Exchange           | M4                 | UberTLC            | KDDCup             | Wikipedia          |
|----------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| Seasonal Naive | 0.512±0.000        | 0.069±0.000        | 0.221±0.000        | 0.011±0.000        | 0.048±0.000        | 0.299±0.000        | 0.561±0.000        | 0.410±0.000        |
| ARIMA          | 0.545±0.006        | -                  | -                  | 0.008±0.000        | 0.044±0.001        | 0.284±0.001        | 0.547±0.003        | -                  |
| ETS            | 0.611±0.040        | 0.072±0.004        | 0.433±0.050        | 0.008±0.000        | 0.042±0.001        | 0.422±0.001        | 0.753±0.008        | 0.715±0.002        |
| Linear         | 0.569±0.021        | 0.088±0.008        | 0.179±0.003        | 0.011±0.001        | <u>0.039±0.001</u> | 0.360±0.023        | 0.513±0.011        | 1.624±1.114        |
| DeepAR         | 0.389±0.001        | 0.054±0.000        | 0.099±0.001        | 0.011±0.003        | 0.052±0.006        | <b>0.161±0.002</b> | 0.414±0.027        | 0.231±0.008        |
| MQ-CNN         | 0.790±0.063        | 0.067±0.001        | -                  | 0.019±0.006        | 0.046±0.003        | 0.436±0.020        | 0.516±0.012        | 0.220±0.001        |
| DeepState      | 0.379±0.002        | 0.075±0.004        | 0.146±0.018        | 0.011±0.001        | 0.041±0.002        | 0.288±0.087        | -                  | 0.318±0.019        |
| Transformer    | 0.419±0.008        | 0.076±0.018        | 0.102±0.002        | 0.010±0.000        | 0.040±0.014        | 0.192±0.004        | 0.411±0.021        | <b>0.214±0.001</b> |
| TFT            | 0.417±0.023        | 0.086±0.008        | 0.134±0.007        | <b>0.007±0.000</b> | <u>0.039±0.001</u> | 0.193±0.006        | 0.581±0.053        | 0.229±0.006        |
| CSDI           | 0.352±0.005        | 0.054±0.000        | 0.159±0.002        | 0.033±0.014        | 0.040±0.003        | 0.206±0.002        | 0.318±0.002        | 0.289±0.017        |
| TSDiff-Cond    | <b>0.338±0.014</b> | 0.050±0.002        | <b>0.094±0.003</b> | 0.013±0.002        | <u>0.039±0.006</u> | 0.172±0.008        | 0.754±0.007        | 0.218±0.010        |
| TSDiff-MS      | 0.391±0.003        | 0.062±0.001        | 0.116±0.001        | 0.018±0.003        | 0.045±0.000        | 0.183±0.007        | 0.325±0.028        | 0.257±0.001        |
| TSDiff-Q       | 0.358±0.020        | <b>0.049±0.000</b> | 0.098±0.002        | 0.011±0.001        | <u>0.036±0.001</u> | 0.172±0.005        | <b>0.311±0.026</b> | 0.221±0.001        |

TSDiff-Cond: conditional model version

TSDiff-MS: Self-guidance of MSE

TSDiff-Q: Self-guidance of Quantile Loss

# 3. Unconditional Time-series Diffusion Model

## TSDiff (NeurIPS 2023)

### 4) Experiments

(Predict) Probabilistic forecast (feat. **Self-guidance**)

#### (b) Forecasting with missing values

Table 2: Forecasting with missing values results on six benchmark datasets.

| Method |             | Solar       | Electricity | Traffic     | Exchange    | UberTLC     | KDDCup      |
|--------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| RM     | TSDiff-Cond | 0.357±0.023 | 0.052±0.001 | 0.097±0.003 | 0.012±0.004 | 0.180±0.015 | 0.757±0.024 |
|        | TSDiff-Q    | 0.387±0.015 | 0.052±0.001 | 0.110±0.004 | 0.013±0.000 | 0.183±0.002 | 0.397±0.042 |
| BM-B   | TSDiff-Cond | 0.377±0.017 | 0.049±0.001 | 0.094±0.005 | 0.009±0.000 | 0.181±0.009 | 0.699±0.009 |
|        | TSDiff-Q    | 0.387±0.019 | 0.051±0.000 | 0.110±0.006 | 0.011±0.001 | 0.182±0.004 | 0.441±0.096 |
| BM-E   | TSDiff-Cond | 0.376±0.036 | 0.065±0.003 | 0.123±0.023 | 0.035±0.021 | 0.179±0.013 | 0.819±0.033 |
|        | TSDiff-Q    | 0.435±0.113 | 0.068±0.009 | 0.139±0.013 | 0.020±0.001 | 0.183±0.005 | 0.344±0.012 |

Missingness: masking 50%

Three scenarios

- (1) random missing (RM)
- (2) blackout missing at the beginning of input (BM-B)
- (3) blackout missing at the end of input (BM-E)

TSDiff-Q performs competitively against task-specific conditional models!

→ **Robustness w.r.t missing values during inference**

# 3. Unconditional Time-series Diffusion Model

## TSDiff (NeurIPS 2023)

### 4) Experiments

(Refine) Prediction refinement of base forecasters (feat. probability density learned by TSDiff)

Table 3: Refinement results on eight benchmark datasets. The best and second best settings have been shown as **bold** and underlined, respectively.

|                | Setting | Solar              | Electricity        | Traffic            | Exchange           | M4                 | UberTLC            | KDDCup             | Wikipedia          |
|----------------|---------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| Seasonal Naive | Base    | 0.512±0.000        | 0.069±0.000        | 0.221±0.000        | 0.011±0.000        | 0.048±0.000        | 0.299±0.000        | 0.561±0.000        | 0.410±0.000        |
|                | LMC-MS  | <b>0.480±0.009</b> | 0.059±0.004        | <u>0.126±0.001</u> | 0.013±0.001        | 0.040±0.002        | <u>0.186±0.005</u> | 0.505±0.027        | <u>0.339±0.001</u> |
|                | LMC-Q   | <b>0.480±0.007</b> | 0.051±0.001        | 0.134±0.004        | <u>0.009±0.000</u> | <b>0.036±0.001</b> | 0.204±0.007        | <u>0.399±0.003</u> | 0.357±0.001        |
|                | ML-MS   | 0.489±0.007        | 0.064±0.006        | 0.130±0.002        | 0.015±0.002        | 0.046±0.003        | 0.202±0.004        | 0.519±0.028        | <u>0.349±0.001</u> |
|                | ML-Q    | <b>0.480±0.007</b> | <u>0.050±0.001</u> | 0.135±0.004        | <u>0.009±0.000</u> | <b>0.036±0.001</b> | 0.215±0.008        | 0.403±0.003        | 0.365±0.001        |
| Linear         | Base    | 0.569±0.021        | 0.088±0.008        | 0.179±0.003        | 0.011±0.001        | 0.039±0.001        | 0.360±0.023        | 0.513±0.011        | 1.624±1.114        |
|                | LMC-MS  | <b>0.494±0.019</b> | 0.059±0.004        | <u>0.113±0.001</u> | 0.013±0.001        | 0.040±0.002        | <u>0.187±0.007</u> | 0.458±0.015        | <u>1.315±0.992</u> |
|                | LMC-Q   | 0.516±0.020        | <u>0.055±0.003</u> | 0.119±0.002        | <u>0.009±0.000</u> | 0.034±0.001        | 0.228±0.010        | <u>0.346±0.010</u> | 1.329±1.002        |
|                | ML-MS   | 0.503±0.016        | 0.063±0.005        | 0.117±0.002        | 0.015±0.002        | 0.045±0.003        | 0.203±0.007        | 0.472±0.015        | 1.327±0.993        |
|                | ML-Q    | 0.523±0.021        | 0.056±0.003        | 0.121±0.003        | 0.010±0.001        | <u>0.032±0.001</u> | 0.240±0.010        | 0.350±0.011        | 1.335±1.002        |
| DeepAR         | Base    | 0.389±0.001        | 0.054±0.000        | <u>0.099±0.001</u> | 0.011±0.003        | 0.052±0.006        | 0.161±0.002        | 0.414±0.027        | 0.231±0.008        |
|                | LMC-MS  | 0.398±0.004        | 0.059±0.004        | 0.111±0.001        | 0.012±0.001        | 0.040±0.002        | 0.184±0.005        | 0.469±0.034        | 0.227±0.002        |
|                | LMC-Q   | <u>0.388±0.002</u> | 0.053±0.001        | 0.101±0.001        | <u>0.010±0.001</u> | <u>0.035±0.001</u> | 0.161±0.002        | <u>0.401±0.021</u> | <u>0.220±0.005</u> |
|                | ML-MS   | 0.402±0.009        | 0.064±0.006        | 0.115±0.002        | 0.014±0.001        | 0.046±0.003        | 0.198±0.005        | 0.477±0.034        | 0.235±0.002        |
|                | ML-Q    | <b>0.386±0.002</b> | <u>0.052±0.001</u> | <u>0.099±0.001</u> | <u>0.010±0.002</u> | <u>0.035±0.001</u> | <u>0.160±0.002</u> | <u>0.401±0.021</u> | <u>0.221±0.006</u> |
| Transformer    | Base    | 0.419±0.008        | 0.076±0.018        | 0.102±0.002        | <b>0.010±0.000</b> | 0.040±0.014        | 0.192±0.004        | 0.411±0.021        | 0.214±0.001        |
|                | LMC-MS  | 0.415±0.009        | 0.059±0.004        | 0.111±0.001        | 0.013±0.001        | 0.040±0.002        | 0.185±0.005        | 0.462±0.014        | 0.229±0.003        |
|                | LMC-Q   | 0.415±0.008        | <b>0.058±0.003</b> | 0.101±0.001        | <u>0.010±0.000</u> | 0.038±0.006        | <u>0.177±0.005</u> | <u>0.384±0.005</u> | <u>0.211±0.002</u> |
|                | ML-MS   | 0.418±0.010        | 0.063±0.005        | 0.115±0.002        | 0.014±0.002        | 0.046±0.003        | 0.198±0.005        | 0.470±0.014        | 0.238±0.003        |
|                | ML-Q    | <b>0.413±0.008</b> | 0.059±0.005        | <u>0.099±0.001</u> | <u>0.010±0.000</u> | <u>0.037±0.006</u> | <u>0.177±0.005</u> | <u>0.384±0.006</u> | <u>0.210±0.002</u> |

**Base:** initial forecast

(method) LMC vs. ML

- **LMC:** Langevin Monte Carlo

- **ML:** Maximum Likelihood

(regularize) MS vs. Q

- **MS:** MSE

- **Q:** Quantile loss

### 3. Unconditional Time-series Diffusion Model

TSDiff (NeurIPS 2023)

#### 4) Experiments

(**Synthesize**) Prediction results of downstream forecasters (feat. **synthetic samples** generated by TSDiff)

Q) *How to evaluate the quality of generated samples?*

A) Several metrics have been proposed ....

But this paper focuses on “predictive metrics”

-> proposes **Linear Predictive Score (LPS)**

Test CRPS of linear (ridge)  
regression model trained on  
synthetic samples.

# 3. Unconditional Time-series Diffusion Model

## TSDiff (NeurIPS 2023)

### 4) Experiments

(**Synthesize**) Prediction results of downstream forecasters (feat. **synthetic samples** generated by TSDiff)

Table 4: Results of forecasters trained on synthetic samples from different generative models on eight benchmark datasets. Best scores are shown in **bold**.

|                 | Generator     | Solar               | Electricity        | Traffic            | Exchange           | M4                 | UberTLC            | KDDCup             | Wikipedia          |
|-----------------|---------------|---------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|
| Linear<br>(LPS) | Real          | 0.569±0.021         | 0.088±0.008        | 0.179±0.003        | 0.011±0.001        | 0.039±0.001        | 0.360±0.023        | 0.513±0.011        | 1.624±1.114        |
|                 | TimeVAE       | 0.933±0.147         | 0.128±0.005        | 0.236±0.010        | 0.024±0.004        | 0.074±0.003        | 0.354±0.020        | 1.020±0.179        | 0.643±0.068        |
|                 | TimeGAN       | 1.140±0.583         | 0.234±0.064        | 0.398±0.092        | <b>0.011±0.000</b> | 0.140±0.053        | 0.665±0.104        | 0.713±0.009        | 0.421±0.023        |
|                 | <b>TSDiff</b> | <b>0.581±0.032</b>  | <b>0.065±0.002</b> | <b>0.164±0.002</b> | 0.012±0.001        | <b>0.045±0.007</b> | <b>0.291±0.084</b> | <b>0.481±0.013</b> | <b>0.392±0.013</b> |
| DeepAR          | Real          | 0.389±0.001         | 0.054±0.000        | 0.099±0.001        | 0.011±0.003        | 0.052±0.006        | 0.161±0.002        | 0.414±0.027        | 0.231±0.008        |
|                 | TimeVAE       | 0.493±0.012         | 0.060±0.001        | 0.155±0.006        | 0.009±0.000        | <b>0.039±0.010</b> | 0.278±0.009        | 0.621±0.003        | 0.440±0.012        |
|                 | TimeGAN       | 0.976±0.739         | 0.183±0.036        | 0.419±0.122        | <b>0.008±0.001</b> | 0.121±0.035        | 0.594±0.125        | 0.690±0.091        | 0.322±0.048        |
|                 | <b>TSDiff</b> | <b>0.478 ±0.007</b> | <b>0.058±0.001</b> | <b>0.129±0.003</b> | 0.017±0.009        | 0.042±0.024        | <b>0.191±0.018</b> | <b>0.378±0.012</b> | <b>0.222±0.005</b> |
| Transf.         | Real          | 0.419±0.008         | 0.076±0.018        | 0.102±0.002        | 0.010±0.000        | 0.040±0.014        | 0.192±0.004        | 0.411±0.021        | 0.214±0.001        |
|                 | TimeVAE       | 0.520±0.030         | 0.071±0.009        | 0.163±0.018        | 0.011±0.001        | 0.035±0.011        | 0.291±0.008        | 0.717±0.181        | 0.451±0.017        |
|                 | TimeGAN       | 0.972±0.687         | 0.182±0.008        | 0.413±0.204        | <b>0.009±0.001</b> | 0.114±0.052        | 0.685±0.448        | 0.632±0.016        | 0.314±0.045        |
|                 | <b>TSDiff</b> | <b>0.457±0.008</b>  | <b>0.056±0.001</b> | <b>0.143±0.020</b> | 0.030±0.021        | <b>0.030±0.008</b> | <b>0.225±0.055</b> | <b>0.356±0.030</b> | <b>0.239±0.010</b> |

Even performs better than the model trained with **REAL** data!

## 4. Conclusion

## 4. Conclusion

- Diffusion model with TS has not been widely explored yet.  
( Most of the work has been done in computer vision domain )
- All the previous works (except for TSDiff) uses “**conditional**” diffusion model
- “**Unconditional**” diffusion model is **task-agnostic**, imposing condition during inference
- Future work: develop unconditional TS diffusion model...
  - with **model architecture** tailored for TS data  
(i.e. Temporal dependency, Time series decomposition)
  - with **noise scheduling** tailored for TS data  
(i.e. exploring ACF(auto-correlation function) )

# References

- Rasul, Kashif, et al. "Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting." ICML. PMLR, 2021.
- Tashiro, Yusuke, et al. "Csdi: Conditional score-based diffusion models for probabilistic time series imputation." NeurIPS (2021): 24804-24816.
- Kollovieg, Marcel, et al. "Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting." NeurIPS (2023).

# Thank You!

(한계) 다양한 TS task 통해 unconditional 모델 강점 부각 X