Denoising Diffusion Models for Time Series Forecast

Denoising Diffusion Probabilistic Model

approximates the distribution by learning how data can be recovered after it has been diffused to pure noise with two processes:

- the forward process (add noise)
- the reverse denoising process (recover data from diffused noise)

The forward process

introduces noise in T-steps from x0 to xT through a Markov chain, until the data distribution resembles a Gaussian distribution

- single step: normal PDF conditioned on the previous step

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

- the full process: joint PDF, product of each step 1...T

$$q(x_{0:T}) = q(x_0)\, q(x_{1:T}|x_0)$$
$$= q(x_0)\prod_{t=1}^{T} q(x_t|x_{t-1})$$

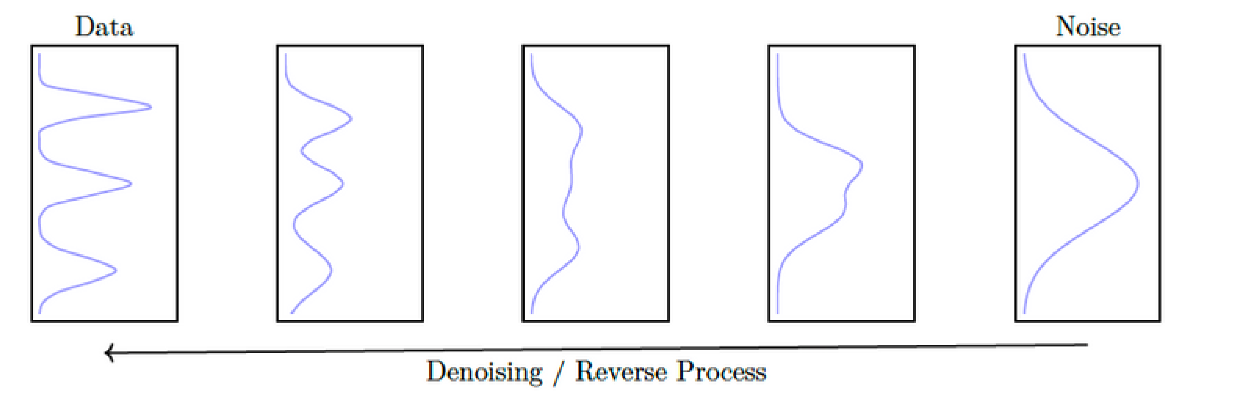


Figure 4: Probability Distribution Evolution: From Noise to Data

The reverse denoising process

transforms the noise data xT back to the input data x0

- reverse of the forward process (derived with Bayes rues): normal distribution with fixed mean vector weighted from x0 to xt and covariance matrix

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1};\ \mu_t(x_t, x_0),\ \tilde{\beta}_t \mathbf{I}\right)$$

$$\alpha_k = \prod_{i=1}^{k}(1-\beta_i)$$

$$\mu_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

- the reverse process (approximated with NN): normal PDF with the mean vector $\mu_p(x_t, t)$ and the covariance matrix $\Sigma_p(x_t, t)$ are two deep **neural networks** that predict the d-dimensional mean and the d×d dimensional covariance matrix for the multivariate Gaussian distribution.

$$p(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p(x_{t-1}|x_t) \qquad \text{(1) joint}$$

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1};\ \mu_p(x_t, t),\ \Sigma_p(x_t, t)) \qquad \text{(2) inductive}$$

$$p(x_T) = \mathcal{N}(x_T;\ \mathbf{0},\ \mathbf{1}) \qquad \text{(3) base}$$

Parameterization for Training

Goal: minimize the *KL divergence* between the derived reverse of the forward process and the reverse process with the loss function:

$$\mathcal{L}_k = D_{KL}\left(q(x^{k-1}|x^k)||p_\theta(x^{k-1}|x^k)\right)$$

Reparameterized in a more stable form: by replacing the forward process $q(x^{k-1}|x^k)$ with:

$$q(x^{k-1}|x^k, x^0) = \mathcal{N}(x^{k-1}; \tilde{\mu}_k(x^k, x^0), \tilde{\beta}_k(x^k, k))$$

$$\text{where } \tilde{\mu}_k(x^k, x^0) = \frac{\sqrt{\alpha_{k-1}}\beta_k}{1-\alpha_t}x^0 + \frac{\sqrt{(1-\beta_k)}(1-\alpha_{k-1})}{1-\alpha_k}x^k$$

$$\text{and } \tilde{\beta}_k = \frac{1-\alpha_{k-1}}{1-\alpha_k}\beta_k$$

$$\mathcal{L}_k = \frac{1}{2\sigma_k^2}||\tilde{\mu}_k(x^k, x^0) - \mu_\theta(x^k, k)||^2$$

Papers and proposed models

The Rise of Diffusion Models in Time-Series Forecasting

- survey of SOTA denoising diffusion models for time-series forecasting in chronological order
- models mentioned: TimeGrad (2021), ScoreGrad (2021), CSDI (2021), DSPD & CSPD (2022), D3VAE (2022), TDSTF (2023), SSSD (2023), DiffLoad (2023), TimeDiff (2023), TSDiff (2023)
- main differences between models and improvements from past models:
    - unconditional models vs. conditional
    - architectures for encoding historical data: RNNs -> transformers -> S4 layers (state-space-model)
    - SDE vs ODE (optimized for speed) for the reverse process
    - data prediction vs. noise prediction
    - interpretability by tracking uncertainties (epistemic [model of the relationship] and aleatoric [data and noise]) by replacing the reverse process with encode-decoders

- suggested future venues:
    - improve interpretability with encode-decoders on latent space
    - S4 layers for encoding historical data
    - latent space diffusion
    - state-space models
    - modeling time series as continuous instead of discrete
    - combining conditioning information
    - data vs. noise prediction
    - long-term multivariate TS forecasting

Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement

- Proposed Model:
  - TimeGrad is an autoregressive model for multivariate probabilistic time series forecasting.
  - It uses diffusion probabilistic models, which sample from the data distribution at each time step by *estimating its gradient*.
  - The model learns gradients by **optimizing a variational bound** on the data likelihood and converts white noise into a sample of the distribution of interest through a Markov chain using Langevin sampling. The upper bound is computed with Jensen's inequality

$$\min_\theta \mathbb{E}_{q(\mathbf{x}^0)}[-\log p_\theta(\mathbf{x}^0)] \leq \min_\theta \mathbb{E}_{q(\mathbf{x}^{0:N})}[-\log p_\theta(\mathbf{x}^{0:N}) + \log q(\mathbf{x}^{1:N}|\mathbf{x}^0)].$$

$$\min_\theta \mathbb{E}_{q(\mathbf{x}^{0:N})}\left[-\log p(\mathbf{x}^N) - \sum_{n=1}^{N} \log \frac{p_\theta(\mathbf{x}^{n-1}|\mathbf{x}^n)}{q(\mathbf{x}^n|\mathbf{x}^{n-1})}\right]$$

  - TimeGrad is a general-purpose high-dimensional distribution model that combines the power of autoregressive models with the flexibility of energy-based models (EBMs).
- Implementation:
  - The model is trained using stochastic gradient descent (SGD) with the Adam optimizer.
  - The training process involves randomly sampling context and prediction length-sized windows from the training time series data.

- In training, the model parameters are optimized to minimize the negative log-likelihood of the model.
- Encode the historical data with RNN and minimize the negative log-likelihood based on conditioning

$$q_{\mathcal{X}}(\mathbf{x}^0_{t_0:T} | \mathbf{x}^0_{1:t_0-1}, \mathbf{c}_{1:T}) = \Pi_{t=t_0}^{T} q_{\mathcal{X}}(\mathbf{x}^0_t | \mathbf{x}^0_{1:t-1}, \mathbf{c}_{1:T})$$

$$\mathbf{h}_{t-1} = \mathrm{RNN}_\theta(\mathtt{concat}(\mathbf{x}^0_{t-1}, \mathbf{c}_t), \mathbf{h}_{t-2}),$$

$$\sum_{t=t_0}^{T} -\log p_\theta(\mathbf{x}^0_t | \mathbf{h}_{t-1}),$$

- The use of EBMs allows for better out-of-distribution detection compared to other likelihood models.

Algorithm 1 Training for each time series step $t \in [t_0, T]$

```
Input: data $\mathbf{x}^0_t \sim q_{\mathcal{X}}(\mathbf{x}^0_t)$ and state $\mathbf{h}_{t-1}$
repeat
    Initialize $n \sim \mathrm{Uniform}(1, \dots, N)$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Take gradient step on
        $\nabla_\theta \| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_n}\mathbf{x}^0_t + \sqrt{1-\bar{\alpha}_n}\epsilon, \mathbf{h}_{t-1}, n)\|^2$
until converged
```

Algorithm 2 Sampling $\mathbf{x}^0_t$ via annealed Langevin dynamics

```
Input: noise $\mathbf{x}^N_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and state $\mathbf{h}_{t-1}$
for $n = N$ to 1 do
    if $n > 1$ then
        $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    else
        $\mathbf{z} = \mathbf{0}$
    end if
    $\mathbf{x}^{n-1}_t = \frac{1}{\sqrt{\alpha_n}}(\mathbf{x}^n_t - \frac{\beta_n}{\sqrt{1-\bar{\alpha}_n}}\epsilon_\theta(\mathbf{x}^n_t, \mathbf{h}_{t-1}, n)) + \sqrt{\Sigma_\theta}\mathbf{z}$
end for
Return: $\mathbf{x}^0_t$
```

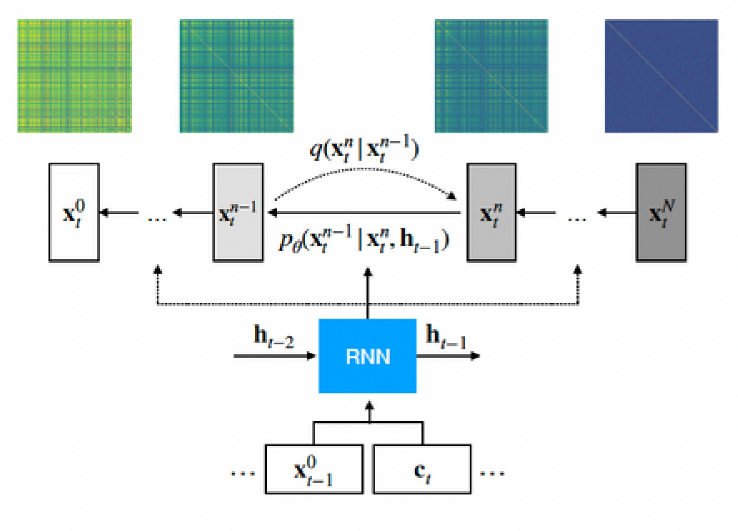


Figure 1. TimeGrad schematic: an RNN *conditioned* diffusion probabilistic model at some time $t-1$ depicting the fixed forward process that adds Gaussian noise to the signal and the learned reverse processes. The cross-correlation of the time series is also depicted during the stages of the process, from white noise (right) to the original signal (left).

- Improvements/Difference Compared to General Diffusion Model:
  - TimeGrad is an autoregressive extension of the general diffusion model.
  - The model learns the conditional distribution of future time steps given the past and covariates.
  - The diffusion process in TimeGrad is conditioned on the hidden state of the RNN.
- Results:
  - TimeGrad achieves the best performance on all benchmark datasets except for the smallest dataset.
  - The model sets a new state-of-the-art in multivariate probabilistic time series forecasting.
  - The results are evaluated using the Continuous Ranked Probability Score (CRPS) metric.

Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement

- Proposed model: D3VAE (Diffusion, Denoise, and Disentanglement Variational Auto-Encoder)
- Implementation: Used a bidirectional VAE as the backbone, coupled diffusion probabilistic model for time series forecasting, scaled denoising network for accuracy, and disentangled latent variables for interoperability.
    - coupled diffusion by diffusing the input series X and target series Y simultaneously

$$X^{(t)} = \sqrt{\bar{\alpha}_t} X^{(0)} + (1-\bar{\alpha}_t)\delta_X := \langle \underbrace{\sqrt{\bar{\alpha}_t} X_r}_{\text{ideal part}}, \underbrace{\sqrt{\bar{\alpha}_t}\epsilon_X + (1-\bar{\alpha}_t)\delta_X}_{\text{noisy part}} \rangle ,$$

$$Y^{(t)} = \sqrt{\bar{\alpha}'_t} Y^{(0)} + (1-\bar{\alpha}'_t)\delta_Y := \langle \underbrace{\sqrt{\bar{\alpha}'_t} Y_r}_{\text{ideal part}}, \underbrace{\sqrt{\bar{\alpha}'_t}\epsilon_Y + (1-\bar{\alpha}'_t)\delta_Y}_{\text{noisy part}} \rangle = \langle \widetilde{Y}_r^{(t)}, \delta_{\widetilde{Y}}^{(t)} \rangle .$$

    - use BVAE as the reverse process the encode-decode diffused data and use DSM (denoising score matching) to clean the generated target series to prevent it to move to Y
    - disentangle latent variables by measuring dependencies among multiple latent variables with Total Correlation (TC).
- Improvements on previous models: D3VAE outperforms competitive algorithms with remarkable margins in terms of MSE and CRPS.

- The difference compared to the general diffusion model: D3VAE augments input and target series simultaneously with a coupled diffusion process, incorporates a denoising score-matching network and disentangles latent variables for better interpretability and stability.
- Results: D3VAE achieves state-of-the-art performance on synthetic and real-world datasets, with significant improvements in MSE and CRPS compared to other models.

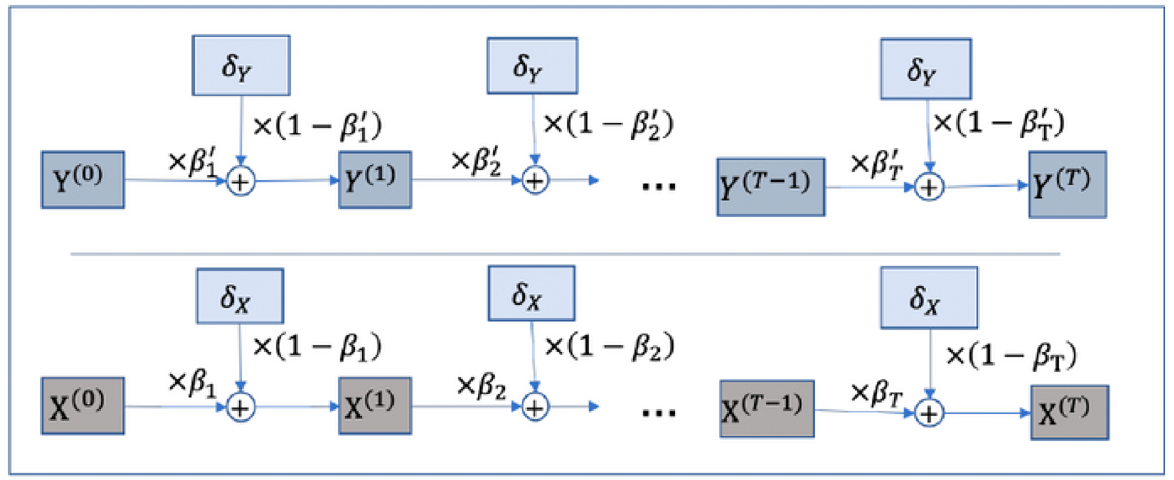
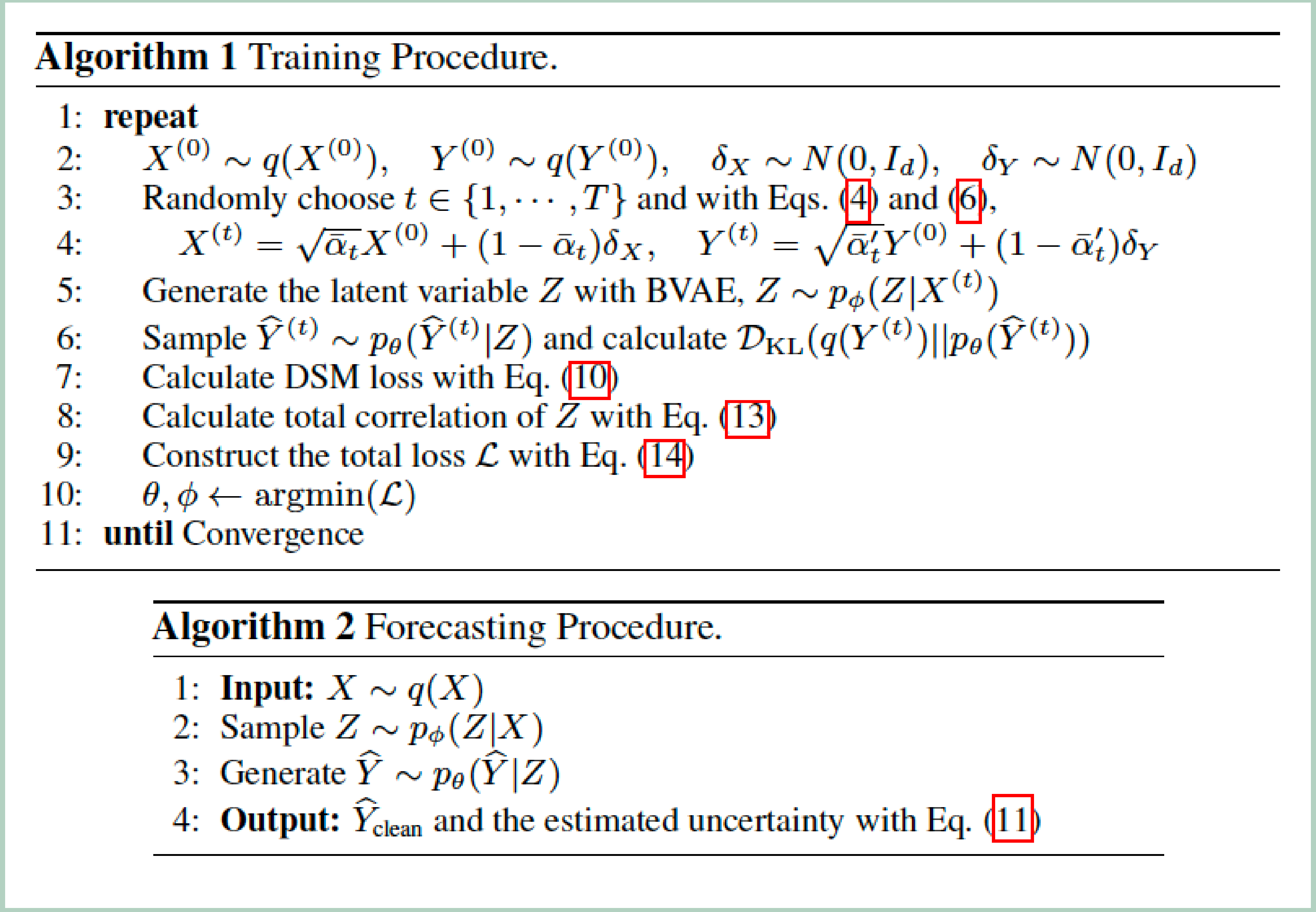


Figure 2: An illustration of the coupled diffusion process. The input $X^{(0)}$ and the corresponding target $Y^{(0)}$ are diffused simultaneously with different variance schedules. $\beta = \{\beta_1, \cdots, \beta_T\}$ is the variance schedule for the input and $\beta' = \{\beta'_1, \cdots, \beta'_T\}$ is for the target.



Algorithm 1 Training Procedure.

```
1: repeat
2:     $X^{(0)} \sim q(X^{(0)}), \quad Y^{(0)} \sim q(Y^{(0)}), \quad \delta_X \sim N(0, I_d), \quad \delta_Y \sim N(0, I_d)$
3:     Randomly choose $t \in \{1, \cdots, T\}$ and with Eqs. (4) and (6),
4:         $X^{(t)} = \sqrt{\bar{\alpha}_t} X^{(0)} + (1 - \bar{\alpha}_t)\delta_X, \quad Y^{(t)} = \sqrt{\bar{\alpha}'_t} Y^{(0)} + (1 - \bar{\alpha}'_t)\delta_Y$
5:     Generate the latent variable $Z$ with BVAE, $Z \sim p_\phi(Z|X^{(t)})$
6:     Sample $\widehat{Y}^{(t)} \sim p_\theta(\widehat{Y}^{(t)}|Z)$ and calculate $\mathcal{D}_{\mathrm{KL}}(q(Y^{(t)})||p_\theta(\widehat{Y}^{(t)}))$
7:     Calculate DSM loss with Eq. (10)
8:     Calculate total correlation of $Z$ with Eq. (13)
9:     Construct the total loss $\mathcal{L}$ with Eq. (14)
10:    $\theta, \phi \leftarrow \mathrm{argmin}(\mathcal{L})$
11: until Convergence
```

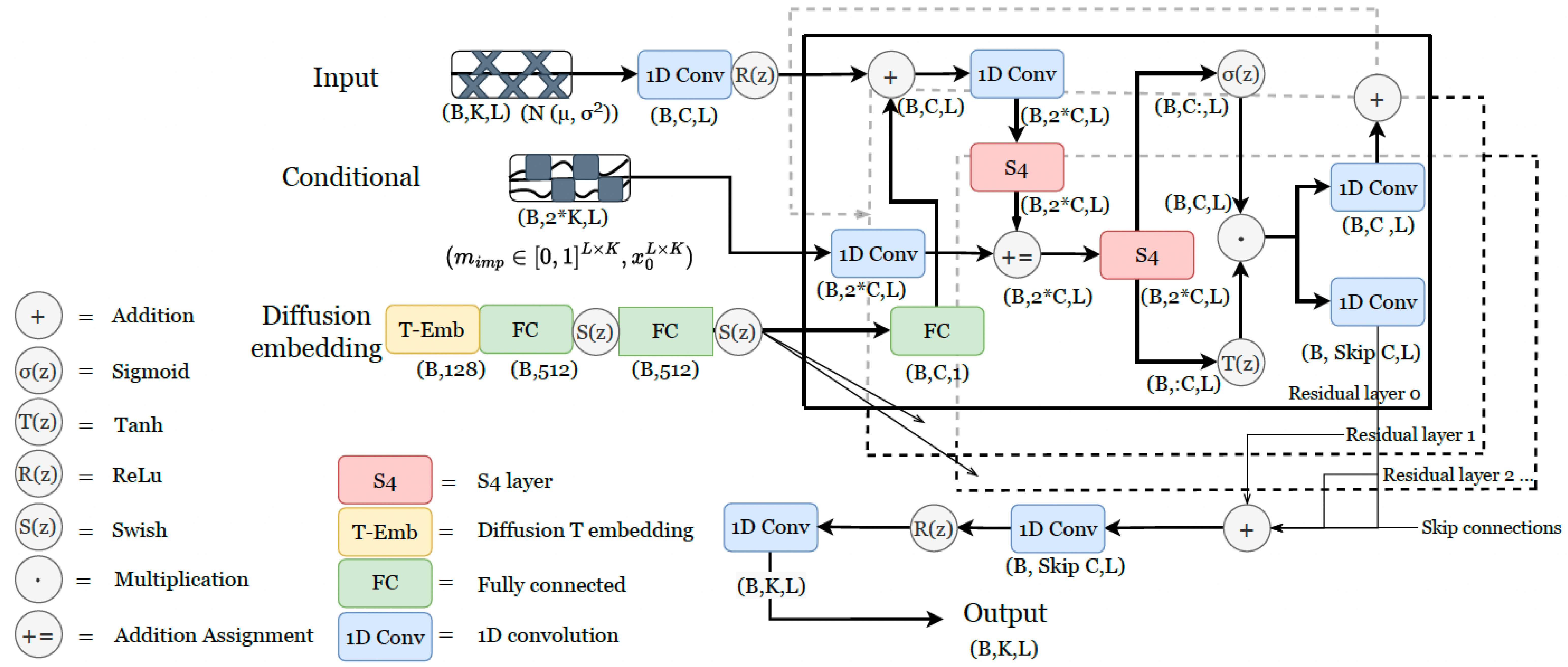
Algorithm 2 Forecasting Procedure.

```
1: Input: $X \sim q(X)$
2: Sample $Z \sim p_\phi(Z|X)$
3: Generate $\widehat{Y} \sim p_\theta(\widehat{Y}|Z)$
4: Output: $\widehat{Y}_{\mathrm{clean}}$ and the estimated uncertainty with Eq. (11)
```

Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models

- Proposed Model: SSSDS4 is a model for time series imputation that combines diffusion models and structured state space models. It uses diffusion embedding, S4 layers, and residual layers to capture long-term dependencies in the data.
- Implementation: SSSDS4 is implemented using a combination of diffusion models and structured state space models. It applies the diffusion process to the regions to be imputed only and uses a conditional diffusion model. The model is trained and evaluated on various datasets with different missingness scenarios and ratios.
    - The structured state-space model connects one-dimensional input sequence u(t) to a one-dimensional output sequence y(t) via a N-dimensional hidden state x(t)
    - A, B, C, D are transitional matrices $x'(t) = Ax(t) + Bu(t) \text{ and } y(t) = Cx(t) + Du(t)\,,$
- Improvements on Previous Models: SSSDS4 improves upon previous models by combining diffusion models and structured state space models, which better capture long-term dependencies in time series data. It also introduces modifications to the DiffWave architecture and introduce noise to the regions to be imputed.

- S4 layers:
  - stacking several copies of the above SSM blocks with appropriate normalization layers, point-wise fully-connected layers in the style of a transformer layer
- Difference Compared to General Diffusion Model: SSSDS4 differs from the general diffusion model by using structured state space models as the main computational building block instead of dilated convolutions or transformer layers. It applies the diffusion process to the regions to be imputed only, instead of the entire sequence. The model also includes additional S4 layers and residual layers. Main contributions:
  a. instead of bidirectional dilated convolutions, use S4 layer as a diffusion layer within each of its residual blocks after adding the diffusion embedding.
  b. include a second S4 layer after the addition assignment
  c. diffuse only on area to be imputed
- Results: SSSDS4 outperforms state-of-the-art approaches in terms of imputation and forecasting performance. It provides qualitatively and quantitatively superior imputations compared to previous models, especially in challenging missingness scenarios. The model achieves significant error reductions and shows robustness to mismatches in missingness ratio and scenarios between train and test time.



Input
(B,K,L)
(N (μ, σ²))
1D Conv
R(z)
(B,C,L)
Conditional
(B,2*K,L)
$(m_{imp} \in [0,1]^{L\times K}, x_0^{L\times K})$
Diffusion embedding
T-Emb
FC
S(z)
(B,128)
(B,512)
(B,512)
+
(B,C,L)
1D Conv
(B,2*C,L)
S4
(B,2*C,L)
1D Conv
(B,2*C,L)
+=
(B,2*C,L)
S4
(B,2*C,L)
FC
(B,C,1)
σ(z)
(B,C:,L)
(B,C,L)
·
T(z)
(B,:C,L)
1D Conv
(B,C ,L)
1D Conv
(B, Skip C,L)
+
Residual layer 0
Residual layer 1
Residual layer 2 ...
Skip connections
+
1D Conv
(B, Skip C,L)
R(z)
1D Conv
(B,K,L)
Output
(B,K,L)
+ = Addition
σ(z) = Sigmoid
T(z) = Tanh
R(z) = ReLu
S(z) = Swish
· = Multiplication
+= = Addition Assignment
S4 = S4 layer
T-Emb = Diffusion T embedding
FC = Fully connected
1D Conv = 1D convolution

Predict, Refine, Synthesize: Self-Guiding Diffusion Models for Probabilistic Time Series Forecasting

- Proposed Model: TSDiff is an unconditional diffusion model for time series forecasting that incorporates a self-guidance mechanism for conditioning during inference. It does not require auxiliary networks or modifications to the training procedure. TSDiff can be used for prediction, refinement, and synthesis tasks.
- Implementation: TSDiff is based on the SSSD architecture and uses lagged time series to capture historical information. It employs mean square self-guidance or quantile self-guidance during inference. TSDiff also uses energy-based sampling or maximizing the likelihood to refine predictions. Trained by an approximation of the evidence lower bound (ELBO) of the
- log-likelihood to predict the sample noise with the simplified objective function.

$$\mu_\theta(\mathbf{x}^t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}^t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}^t, t)\right)$$

$$\mathbb{E}_{\mathbf{y},\boldsymbol{\epsilon},t}\left[\|\boldsymbol{\epsilon}_\theta(\mathbf{x}^t, t) - \boldsymbol{\epsilon}\|^2\right]$$

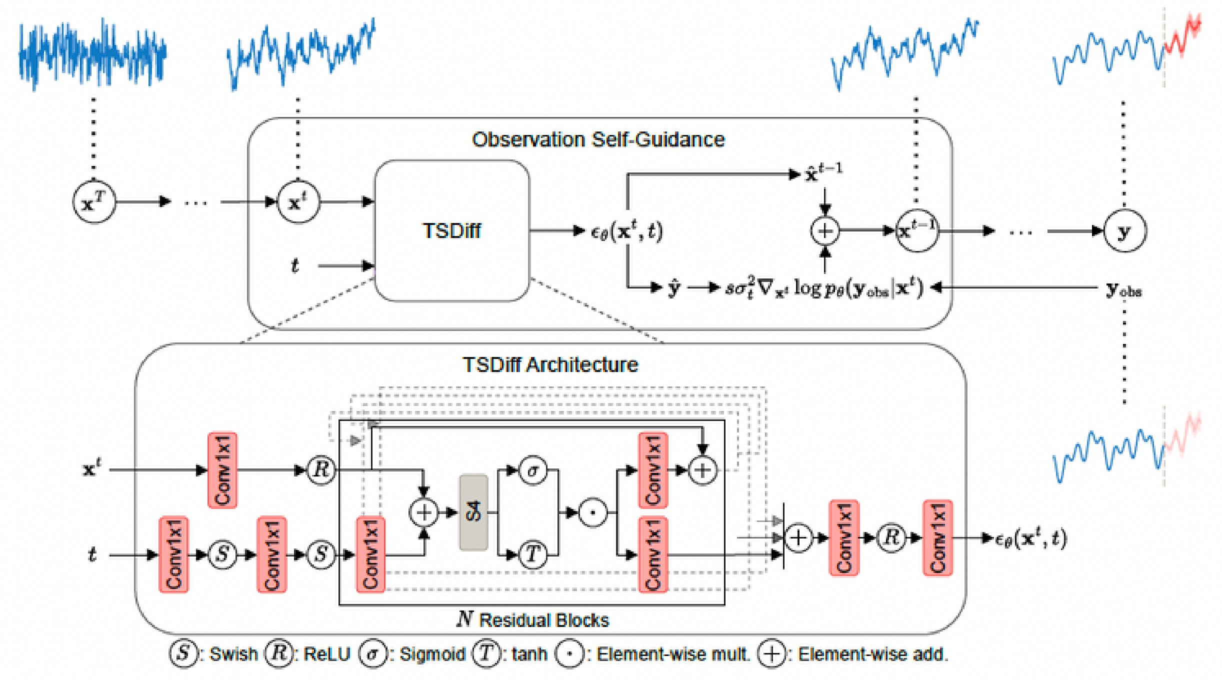
- Improvements on Previous Models: TSDiff outperforms task-specific conditional models and offers a task-agnostic approach to time series forecasting. It achieves competitive performance without requiring conditional training. TSDiff also introduces a method to iteratively refine predictions with reduced computational overhead.

- Difference Compared to General Diffusion Model: TSDiff is specifically designed for time series forecasting and incorporates self-guidance and prediction refinement techniques. It leverages the learned implicit probability density to improve predictions and generate realistic samples. TSDiff can be conditioned during inference for arbitrary forecasting tasks without the need for auxiliary networks.
  - to incorporate the historical information beyond L timesteps, append lagged time series along the channel dimensions to add noise to the inputs
  - resulting input: $\mathbf{x}^t \in \mathbb{R}^{L\times C}$, C-1 is the number of lags
  - classifier guidance decomposes class-conditional score function with log-likelihood into conditional and marginal score functions, which allows drawing samples from the decomposed conditional function

$$p_\theta(\mathbf{x}^t|\mathbf{y}_{\text{obs}}) \propto p_\theta(\mathbf{y}_{\text{obs}}|\mathbf{x}^t)p_\theta(\mathbf{x}^t),$$

$$\nabla_{\mathbf{x}^t} \log p_\theta(\mathbf{x}^t|\mathbf{y}_{\text{obs}}) = \nabla_{\mathbf{x}^t} \log p_\theta(\mathbf{y}_{\text{obs}}|\mathbf{x}^t) + \nabla_{\mathbf{x}^t} \log p_\theta(\mathbf{x}^t)$$

$$p_\theta(\mathbf{x}^{t-1}|\mathbf{x}^t, \mathbf{y}_{\text{obs}}) = \mathcal{N}(\mathbf{x}^{t-1}; \mu_\theta(\mathbf{x}^t, t) + s\sigma_t^2\nabla_{\mathbf{x}^t} \log p_\theta(\mathbf{y}_{\text{obs}}|\mathbf{x}^t), \sigma_t^2\mathbf{I}).$$



Observation Self-Guidance
TSDiff
$\epsilon_\theta(\mathbf{x}^t, t)$
$\hat{\mathbf{x}}^{t-1}$
$\hat{\mathbf{y}}$
$s\sigma_t^2\nabla_{\mathbf{x}^t} \log p_\theta(\mathbf{y}_{\text{obs}}|\mathbf{x}^t)$
$\mathbf{y}_{\text{obs}}$
TSDiff Architecture
Conv1x1
S4
N Residual Blocks
S: Swish R: ReLU σ: Sigmoid T: tanh ⊙: Element-wise mult. ⊕: Element-wise add.

Figure 2: An overview of observation self-guidance. The predicted noise, $\boldsymbol{\epsilon}_\theta(\mathbf{x}^t, t)$, first denoises $\mathbf{x}^t$ unconditionally as $\hat{\mathbf{x}}^{t-1}$ and approximates $\mathbf{y}$ as $\hat{\mathbf{y}}$. The reverse diffusion step then guides $\hat{\mathbf{x}}^{t-1}$ via the log-likelihood of the observation $\mathbf{y}_{\text{obs}}$ under a distribution parameterized by $\hat{\mathbf{y}}$.

- **Refinement:** proposed two methods of refining forecasts (from any base forecasters)
    - Energy-based sampling
    - Maximizing the likelihood to find the most likely sequence
- **Synthesis:** evaluate downstream's forecaster's performance on TSDiff synthetic data vs. real data
    - TSDiff's sample quality evaluated via downstream model performance.
    - LPS, measured by ridge regression, used as a reliable metric.
    - TSDiff samples outperformed TimeGAN and TimeVAE.
    - TSDiff showed strong performance against DeepAR and Transformer forecasters.
    - Despite lacking extended time features, TSDiff samples were comparable to real data performance.
    - Results indicate TSDiff effectively captures time series patterns and generates realistic samples.

- Results: TSDiff demonstrates competitive performance against task-specific models and outperforms other generative models in terms of predictive quality. It introduces the Linear Predictive Score (LPS) metric to evaluate generative performance. TSDiff generates realistic samples and achieves good performance on real test data.

Non-autoregressive Conditional Diffusion Models for Time Series Prediction

- Model Implementation:
  - TimeDiff: Proposed for modeling non-stationary time series data, it introduces novel conditioning mechanisms tailored for time series prediction.
  - The model utilizes a forward diffusion process to obtain diffused samples based on the input and noise.
  - Conditioning is applied to the backward denoising process, where past and future time series information is combined to guide the denoising.
  - The denoising network, shown in red in Figure 1, is a crucial part of the model.
- Forward Diffusion Process:
  - The forward diffusion process diffuses input data by combining it with random noise.
  - It involves a straightforward calculation based on a formula that incorporates the input, noise, and diffusion parameters.
- Conditioning the Backward Denoising Process:
  - The model employs mixup to combine past and future time series information, producing a conditioning signal called zmix.
  - An autoregressive model is used to provide an initial guess for the future time series segment, which is concatenated with zmix to form the final conditioning signal.

  - This conditioning signal guides the denoising process to predict the future time series segment accurately.
- Architecture Used:
  - The model architecture includes an encoder, decoder, and denoising network.
  - The denoising network consists of convolutional layers for processing the input data and generating denoised output.
- Improvements/Differences with Previous Models:
  - TimeDiff focuses on capturing complex temporal dependencies in non-stationary time series data, which is crucial for real-world applications.
  - It introduces novel conditioning mechanisms tailored for time series prediction, such as mixup and autoregressive modeling.
  - Compared to previous models like D3VAE, TimeDiff offers better denoising capabilities by incorporating conditioning mechanisms specific to time series prediction.
  - The use of mixup and autoregressive modeling distinguishes TimeDiff from other models and improves its performance in capturing temporal patterns effectively.