Diffusion Models for Robust RUL/TTF Prediction under Complex Degradational Process

Donghyun Ko

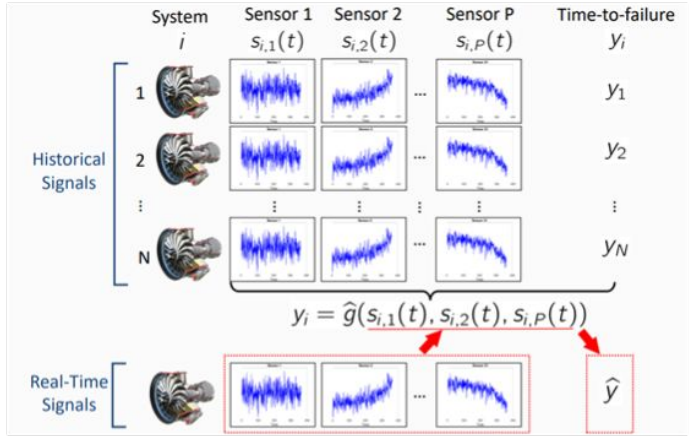
- **Name:** Donghyun Ko
- **Affiliation:** Ph.D. Student, Industrial & Systems Engineering, NC State University
- **Background:** Republic of Korea Army, Major **(“Currently, on an academic leave”)**
  * TRADOC(Training and Doctrine Command), Center for Army Analysis
- **Research Interests:** ML/DL, Deep generative learning, Diffusion model, Optimization, Statistical inference, Predictive maintenance,
- **Current Work:** Conditional diffusion models for robust RUL/TTF prediction,
  Optimizing on-time power system node setting by Diffusion model,
  Optimizing 3D printing parameters with ML-based approach

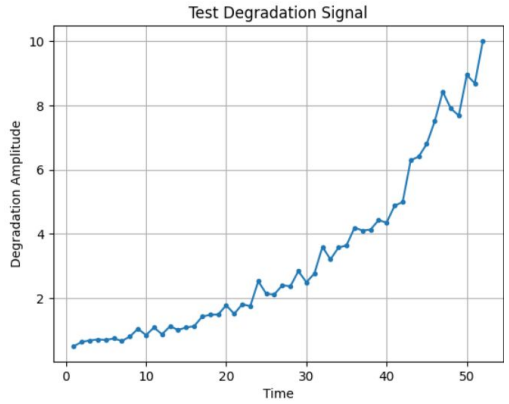
❑ Motivation

❖ **Predicting remaining useful life (RUL) and time-to-failure (TTF) of engineering systems is a key enabler of predictive maintenance and system reliability in manufacturing**

➢ Degradation signal data

- In practice, there is **no dedicated "degradation sensor"**. A degradation signal is a 1D time series engineered from existing sensors so it reflects aging over time!
  - **Step 1. Sensor Selection**: Select only failure-informative sensors
    - Group LASSO, Elastic Net, and other domain knowledge used
  - **Step 2. Feature Extraction**: Reduce dimension of the selected sensors via PCA, or FPCA
    - The selected top 'K' scores are extracted features
  - **Step 3. Fuse the selected features into a 1D time-series**
    - Use the first principal component (Simplest but unstable)
    - Use other DL-based methods to fuse them into a 1D time-series
- **Degradation data typically shows 1D exponential increasing pattern with noise**
  - Complex system shows non-monotonic, or multimodal degradation pattern (e.g., thermal cycling, load/speed changes, maintenance resets, and etc.)





➢ Traditional RUL/TTF methods fall into two categories:

- **Model-based approach:** Explicitly model stochastic degradation processes (Bayesian updating)
- **Data-driven approach:** Apply M/L methods, such as principal component analysis (PCA) followed by lognormal regression

❑ Motivation (Cont')

➢ Recent advances in generative deep learning provide alternatives

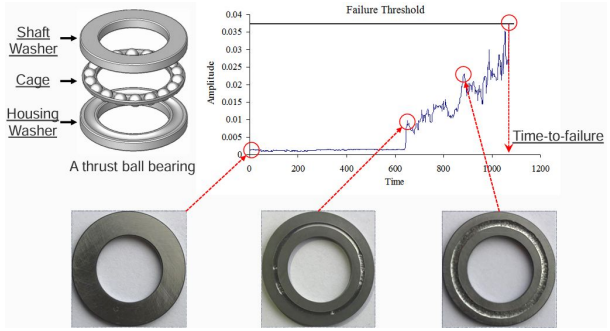
- **Diffusion models offer probabilistic frameworks to reproduce complex and high-dimensional data**
- **Diffusion-based models have shown strong results in image generation, time series imputation,** and recently RUL/TTF prediction
  - RUL Prediction Using Conditional Score-based Diffusion Models (Tang & Che, PHM 2023)
  - Remaining Useful Life Prediction Using Class-Conditional Diffusion Models (Zhang et al., PHM 2023)
  - Bayesian Score-based Diffusion Models for Uncertainty-Aware Remaining Useful Life Prediction (Li et al., 2023)
  - Diffusion Models for Remaining Useful Life Prediction with Uncertainty Quantification (Wen et al., 2024), and etc.
- While these demonstrate **feasibility** of applying diffusion model for RUL/TTF prediction, systematic benchmarking against classical baselines(e.g., Parametric methods; M/L-based methods) remains limited; **how much does diffusion models perform better?**
- They also tend to use **curated datasets** with relatively smooth degradation patterns, leaving open questions regarding the robustness of diffusion models under more realistic or abrupt signal behaviors

❑ Benchmark 1. Bayesian Updating: i.i.d & BM model

❖ Objective: To model the functional form of the degradation process

➢ Candidates: Exponential + i.i.d' vs. 'Exponential + Brownian Motion(BM)'

- Degradation is assumed to follow an exponential form: **Model (with i.i.d. errors):**



$$S(t_i) = \phi + \theta \exp\left(\beta t_i + \varepsilon(t_i) - \frac{\sigma^2}{2}\right)$$

Where:

- ϕ: A constant (typically the baseline level)
- θ: A log-normal random variable representing the initial state of the component
- β: Degradation rate (normally distributed)

- **i.i.d Model:** This model uses a log-transformation to express the degradation process in linear form. Then, linear regression can be applied to estimate the initial condition and degradation rate of each component

$\varepsilon(t_i)$: Observation error, assumed to be i.i.d. normal $N(0, \sigma^2)$ ⇒ $L(t_i) = \ln(S(t_i) - \phi) = \ln\theta + \beta t_i + \varepsilon(t_i) - \frac{\sigma^2}{2}$

- **BM Model:** Same modeling, yet with assumption that errors are from Brownian motion error

It is suitable for modeling degradation processes with time-dependent noise

$\varepsilon(t) = \sigma W(t)$ where $W(t)$ is a standard Brownian motion ⇒ $L(t) = \theta' + \beta' t + \varepsilon(t)$

θ', β': Parameters of the distribution

$L(t)$: Observed degradation signal amplitude at time step 't'

❑ Benchmark 1. Bayesian Updating: i.i.d & BM model (Cont')

❖ Procedure of Bayesian Updating

➢ 3 Steps: Assume prior distribution - Posterior update - Derive RUL distribution

- Step 1. **Assume the prior distribution** of the parameters in the two candidates as 'Gaussian distribution'

  (Can be estimated by training data) $\theta' \sim N(\mu_{\theta'}, \sigma^2_{\theta'}), \quad \beta \sim N(\mu_\beta, \sigma^2_\beta)$ "Prior" ➡ $\pi(\theta') = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left\{-\frac{(\theta'-\mu_{\theta'})^2}{2\sigma_0^2}\right\} \quad \pi(\beta) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left\{-\frac{(\beta-\mu_\beta)^2}{2\sigma_1^2}\right\}$

- Step 2. **Estimate posterior distribution** of the parameters by combining priors and observed (test)data

  - i.i.d model: $L(t_i) = \theta' + \beta t_i + \varepsilon_i, \quad \varepsilon_i \overset{\text{iid}}{\sim} N(0,\sigma^2)$ ➡ $L(t_i) \mid \theta', \beta \sim N\left(\theta' + \beta t_i,\ \sigma^2\right)$ ➡ $f(L(t_1),\ldots,L(t_k) \mid \theta', \beta) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{\left[L(t_i)-\theta'-\beta t_i\right]^2}{2\sigma^2}\right\}$ "Likelihood"

    RUL regression model

    By bayes' theorem, $p(\theta', \beta \mid \{L(t_i)\}) \propto f\left(\{L(t_i)\} \mid \theta', \beta\right) \pi(\theta')\,\pi(\beta)$

    $\propto \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{k}\left[L(t_i)-\theta'-\beta t_i\right]^2\right\}$

    ➡ $p(\theta', \beta \mid \{L(t_i)\}) \propto \exp\left\{-\frac{1}{2\sigma^2}\sum_{i=1}^{k}\left[L(t_i)-\theta'-\beta t_i\right]^2 - \frac{(\theta'-\mu_{\theta'})^2}{2\sigma_0^2} - \frac{(\beta-\mu_\beta)^2}{2\sigma_1^2}\right\}$ "Posterior"

    ➡ Posterior is a bivariate normal distribution: $(\theta', \beta) \mid \{L(t_i)\} \sim \mathcal{N}\left((\mu^*_{\theta'}, \mu^*_\beta), \Sigma^*\right)$

    "This posterior is updated each time new sensor data is received!"

- Step 3. **Derive the Remaining Useful Life(RUL) Distribution**

  - RUL 'T' is the time after the current observation t_k such that $L(t_k+T) = D$

    $L(t_k+T) = \theta' + \beta(t_k+T) + \boxed{\varepsilon(t_k+T)}$ where $\varepsilon(t_k+T) \sim \boxed{\mathcal{N}(0, \sigma^2_\varepsilon(T))}$

    - In the **i.i.d. error model**: $\sigma^2_\varepsilon(T) = \sigma^2$ (constant).
    - In the **Brownian motion model**: $\sigma^2_\varepsilon(T) = \sigma^2(t_k+T)$

    Thus, conditionally:

    $L(t_k+T) \sim \mathcal{N}(\mu_{\theta'} + \mu_\beta(t_k+T), \boxed{\sigma^2_{\text{post}}(t_k+T)})$

    - For the **i.i.d. case**: $\sigma^2_{\text{post}}(t_k+T) = \sigma^2_{\theta'} + (t_k+T)^2\sigma^2_\beta + \sigma^2$
    - For the **Brownian motion case**: $\sigma^2_{\text{post}}(t_k+T) = \sigma^2_{\theta'} + (t_k+T)^2\sigma^2_\beta + \sigma^2(t_k+T)$

    Define: $\mu(t_k+T) = \mu_{\theta'} + \mu_\beta(t_k+T)$ ➡ Then: $L(t_k+T) \sim \mathcal{N}(\mu(t_k+T), \sigma^2(t_k+T))$

❑ Benchmark 1. Bayesian Updating: i.i.d & BM model (Cont')

❖ Procedure of Bayesian Updating

➢ 3 Steps: Assume prior distribution - Posterior update - Derive RUL distribution

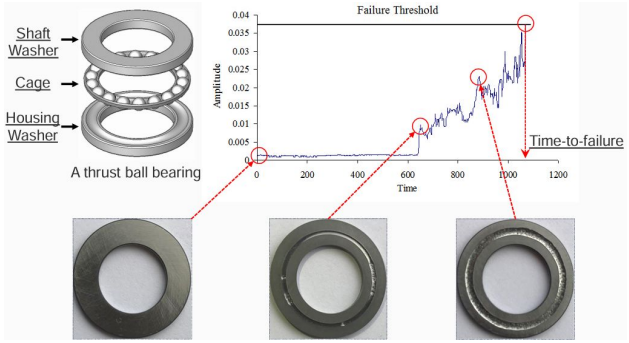
- Step 3. **Derive the Remaining Useful Life(RUL) Distribution**
  - RUL 'T' is the time after the current observation t_k such that $L(t_k + T) = D$
  - Using the posterior distribution, $L(t_k + T)$ is treated as a normally distributed variable, leading to a CDF such that

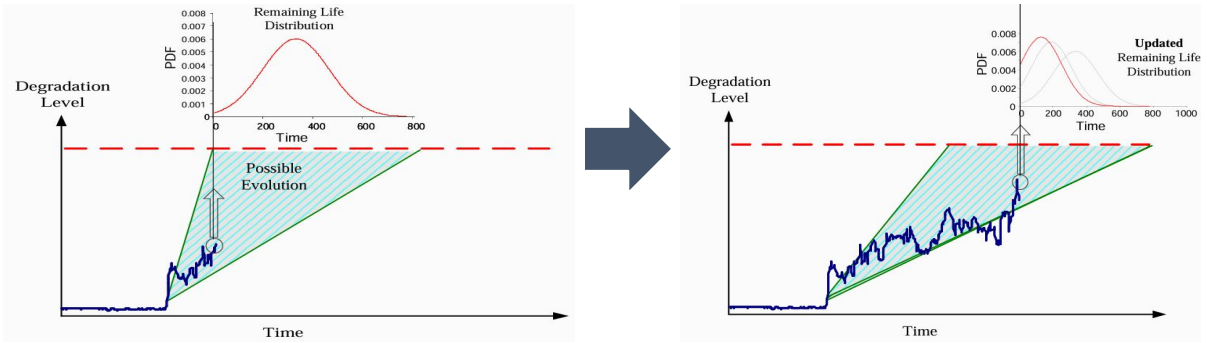
$$P\{T \le t \mid L(t_1), \ldots, L(t_k)\} = P\{L(t_k + t) \ge D \mid L(t_1), \ldots, L(t_k)\} \Rightarrow P\{L(t_k + t) \ge D\} = 1 - P\{L(t_k + t) < D\} = 1 - \Phi\left(\frac{D - \mu(t_k + t)}{\sigma(t_k + t)}\right)$$

$$\Rightarrow P\{T \le t \mid \{L(t_i)\}\} = \Phi\Big(g(t)\Big) \text{ where } g(t) = \frac{\mu(t_k + t) - D}{\sigma(t_k + t)} = \frac{\mu_{\theta'} + \mu_\beta(t_k + t) - D}{\sigma(t_k + t)}$$

$$\Rightarrow F_{T|\{L(t_i)\}}(t) \approx \Phi\Big(g(t)\Big)$$

  - PDF of RUL: $f_T(t \mid \{L(t_i)\}) = \frac{d}{dt} F_T(t) \approx \phi(g(t)) \cdot g'(t)$ Where:
    - $\phi(\cdot)$ is the standard normal PDF,
    - $g'(t)$ is the derivative of $g(t)$ with respect to t





❑ Benchmark 2. PCA + Lognormal regression model

❖ Objective: To model the RUL distribution with the dimension-reduced variables

➢ PCA on Training data(X_{train})

- Use X_{obs} of the training data for each ratio
- Center the data by subtracting the mean (mu_hat): $X_{\text{centered}} = X_{\text{obs}} - \hat{\mu}$
- Compute the covariance matrix(C) of the centered data: $C = \frac{1}{n-1} X^\top X$
- Perform eigen decomposition on 'C' to obtain eigenvalues (λ_k) and eigenvectors (ϕ_k)
- Project the data onto these eigenvectors and determine the number of principal components (k) to keep 95%
- Truncate the PCA projection to only include the top 'k' components for each ratio

➢ Whole Procedure

Step 1: PCA in training data

- Apply PCA on X_{train}:

$$X_{\text{obs}}^{\text{PCA}} = \Phi^\top X_{\text{obs}}$$

- Select number of components explaining 95% of total variance.
- The truncation here refers to retaining only the d-dimensional PCA space:

$$X_{\text{obs}}^{\text{PCA}} \in \mathbb{R}^d$$

Step 2: Lognormal regression

- Log-normal regression fit:

$$\ln(\text{RUL}) = w^\top X_{\text{obs}}^{\text{PCA}} + b + \eta, \quad \eta \sim \mathcal{N}(0, \sigma^2)$$

Step 3: Prediction on Test Data

- Project X_{test} to PCA space:

$$X_{\text{test}}^{\text{PCA}} = \Phi^\top X_{\text{test}}$$

- Predict:

$$\ln(\text{RUL}) \sim \mathcal{N}(w^\top X_{\text{test}}^{\text{PCA}} + b, \sigma^2)$$

- Back-transform:

$$\text{RUL} \sim \text{LogNormal}(\mu, \sigma^2)$$

Step 4: Sampling and RUL Distribution

- Draw M samples of RUL from the predictive distribution.
- Compute:

$$\hat{T} = \text{mean/median(RUL samples)}$$

❑ Fundamentals of Diffusion Model

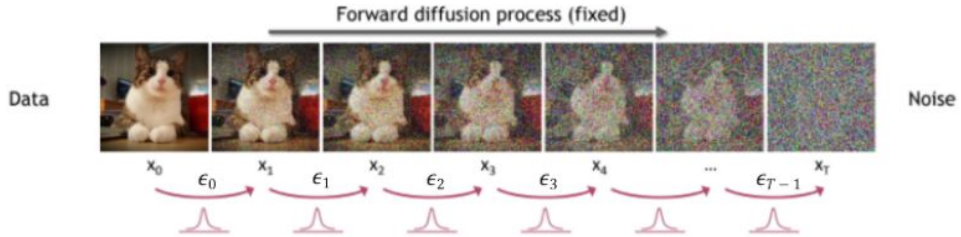
❖ DM consists of two procedure to generate NEW data by manifold learning and sampling from it

- ➢ [Forward] Take a data sample and progressively corrupt it with small amounts of noise until it becomes pure noise
- ➢ [Reverse] Train a neural network to learn the noise distribution added at each time step 't'

→ Once trained, one can sample from a standard normal distribution and apply the learned reverse transitions to remove the noise added at each time step and finally generate realistic data samples

❖ Forward Process

- ➢ Begin with $x_0 \sim p_{data}(x)$, where x_0 is assumed to have been standardized to zero mean and identity covariance
  - ■ For t = 1, 2, . . . , T, we define the forward process as Gaussian MC transition: $q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I)$
    - $\beta_t \in (0, 1)$ is the noise variance at step 't'
    - Equivalently, we can write the forward process in generative form s.t: $x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \epsilon_{t-1}, \quad \epsilon_{t-1} \sim \mathcal{N}(0, I)$



  - ■ One can check by induction that for any t, x_t remains zero mean and unit variance as well!

$$\mathbb{E}[x_t] = \sqrt{\alpha_t}\, \mathbb{E}[x_{t-1}] + \sqrt{\beta_t}\, \mathbb{E}[\epsilon_{t-1}] = 0, \text{ and}$$

$$\mathrm{Var}(x_t) = (1-\beta_t)\, \mathrm{Var}(x_{t-1}) + \beta_t\, \mathrm{Var}(\epsilon_{t-1}) = (1-\beta_t)I + \beta_t I = I$$

❑ Fundamentals of Diffusion Model

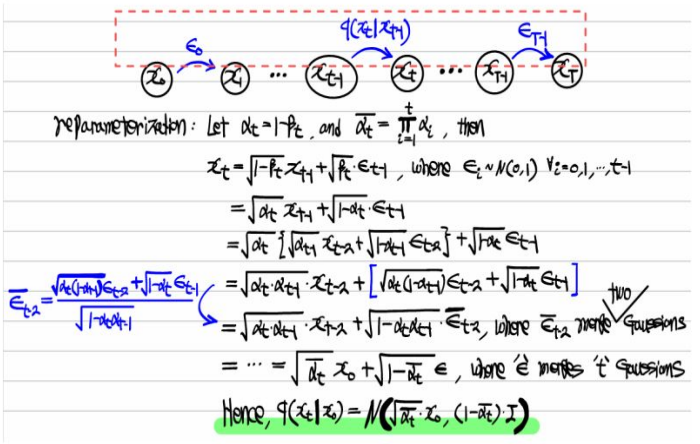
$$\alpha_t := 1 - \beta_t, \quad \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s = \prod_{s=1}^{t} (1 - \beta_s)$$

❖ Forward Process (Cont')

➢ Reparameterization: Simplifying Forward Process, $q(x_t \mid x_0)$

- By denoting $\alpha_t = 1 - \beta_t$, and combining two Gaussians,
  - One can sample x_t directly from x_0 in one step s.t:

$$q(x_t \mid x_0) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}\, x_0,\ (1 - \bar{\alpha}_t)\, I\right), \text{ or } \quad x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$



❖ Reverse Process

➢ We're assuming that reverse process is also MC, yet intractable to obtain a closed form

$q(x_{t-1} \mid x_t)$ **(intractable):** Using the conditional marginalization formula,

$$q(x_{t-1} \mid x_t) = \int q(x_{t-1}, x_0 \mid x_t)\, dx_0 = \int q(x_{t-1} \mid x_t, x_0)\, q(x_0 \mid x_t)\, dx_0,$$

where $q(x_0 \mid x_t) = \frac{q(x_t \mid x_0)\, p_{\text{data}}(x_0)}{q(x_t)}$ is intractable due to unknown p_{data}, so we cannot evaluate this integral directly.

➢ However, by leveraging the reverse-step Markov property, we can replace this intractable marginal posterior with a tractable conditional Gaussian that depends on the original sample x_0

$$q(x_{t-1} \mid x_t, x_0) = \frac{q(x_t \mid x_{t-1})\, q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)}, \text{ where } q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}\, x_{t-1}, \beta_t I), \quad q(x_{t-1} \mid x_0) = \mathcal{N}(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\, x_0, (1 - \bar{\alpha}_{t-1})I), \text{ and}$$
$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1 - \bar{\alpha}_t)I)$$

❑ Fundamentals of Diffusion Model

❖ Reverse Process (Cont')

➢ Completing the square yields the known Gaussian s.t: $q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I\right)$

- with $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ and $\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t$

➢ In practice, we often rewrite $\tilde{\mu}_t(x_t, x_0)$ in terms of the actual noise ϵ_t s.t:

$$\tilde{\mu}_t(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\right), \text{ where } \epsilon_t = \frac{x_t - \sqrt{\bar{\alpha}_t}\,x_0}{\sqrt{1-\bar{\alpha}_t}} \text{ from } x_t = \sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

- This alternate form will be crucial when we parameterize the mean of our model to predict ϵ_t directly in the reverse process

➢ We, therefore, can approximate the intractable marginal $q(x_{t-1} \mid x_t)$ by a Gaussian distribution due to $q(x_{t-1} \mid x_t, x_0) = q(x_{t-1} \mid x_t)$ by Markov property

➢ Now, we want to learn the parameters by a Neural network s.t.:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta^2(x_t, t)\, I\right)$$

➢ How to learn?

- By minimizing the KL-divergence of the two $[\mathrm{KL}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))]$, we can learn how to denoise at each step 't' iteratively to get x_0 from x_t s.t:

$$p_\theta(x_{t-1} \mid x_t) \approx q(x_{t-1} \mid x_t)$$

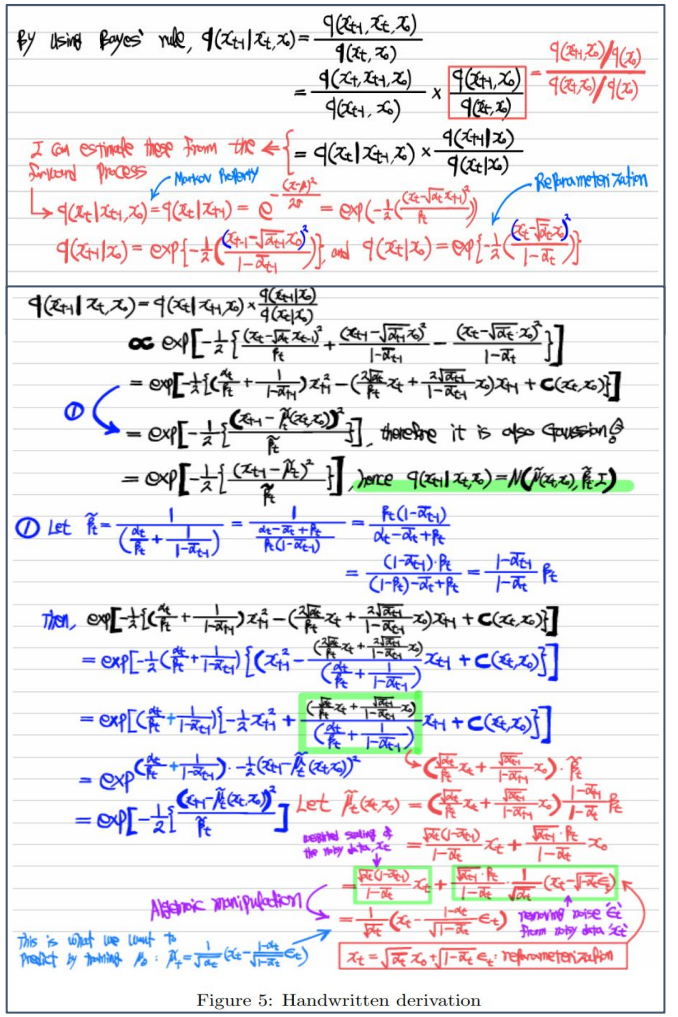


Figure 5: Handwritten derivation

Gaussian-to-Gaussian KL Formula

The KL divergence between two multivariate Gaussian distributions in $\mathbb{R}^k$, each with diagonal covariance, is given by:

$$D_{\mathrm{KL}}\left(\mathcal{N}(\mu_1, \sigma_1^2 I) \,\|\, \mathcal{N}(\mu_2, \sigma_2^2 I)\right) = \frac{1}{2}\left(\frac{\sigma_1^2}{\sigma_2^2} + \frac{\|\mu_1 - \mu_2\|^2}{\sigma_2^2} - k + k \log \frac{\sigma_2^2}{\sigma_1^2}\right)$$

This expression compares two Gaussian distributions:

- $\mathcal{N}(\mu_1, \sigma_1^2 I)$: the **true posterior** $q(x_{t-1} \mid x_t, x_0)$
- $\mathcal{N}(\mu_2, \sigma_2^2 I)$: the **learned reverse model** $p_\theta(x_{t-1} \mid x_t)$

Each term in the KL formula has an intuitive interpretation:

- $\frac{\sigma_1^2}{\sigma_2^2}$: captures how the posterior's variance compares to the model's variance
- $\frac{\|\mu_1 - \mu_2\|^2}{\sigma_2^2}$: measures the difference between means, normalized by model variance
- $-k$: compensates for dimensionality
- $k \log \frac{\sigma_2^2}{\sigma_1^2}$: accounts for the entropy gap between the distributions

❑ Fundamentals of Diffusion Model

❖ How to train the Reverse Process?

➢ The ultimate goal of training a diffusion model: Maximize the marginal likelihood of log $p_\theta(x_0)$

$$p_\theta(x_0) = \int p_\theta(x_{0:T})\, dx_{1:T} = \int p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)\, dx_{1:T}$$

- Integration space dimension = T × d ➡ Computationally infeasible & analytically intractable!

➢ Therefore, we resort to variational inference techniques such as “ELBO-based optimization”

- We can modify the marginal likelihood log $p_\theta(x_0)$ to get its ELBO function

$$\begin{aligned}
\log p_\theta(x_0) &= \log \left[\int q(x_{1:T} \mid x_0) \cdot \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\, dx_{1:T} \right] \\
&= \log \left[\mathbb{E}_{q(x_{1:T}|x_0)} \left(\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right) \right] \\
&\geq \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \left(\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right) \right] \quad \text{(by Jensen's Inequality)} \\
&= \mathbb{E}_{q(x_{1:T}|x_0)} \left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0) \right]
\end{aligned}$$

$$\log p_\theta(x_0) \geq \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)} \left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0) \right]}_{\mathcal{L}(\theta; x_0)\ \text{(ELBO)}}$$

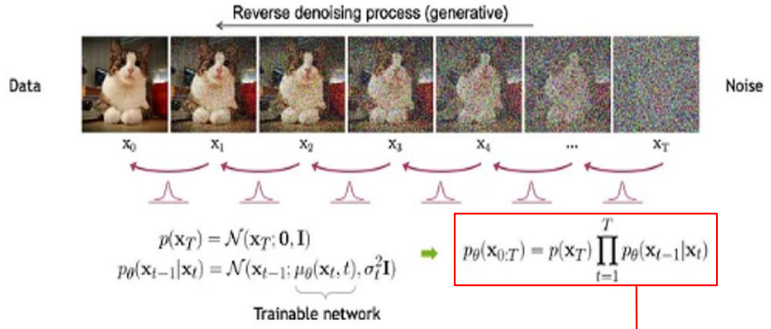
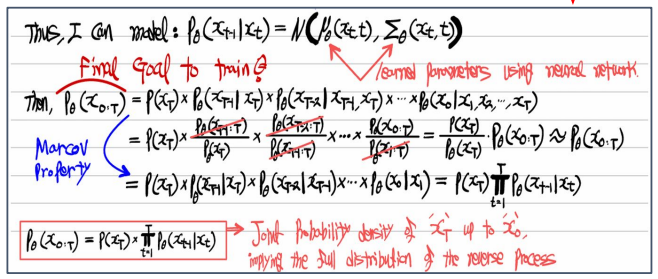


Figure 7: Reverse Diffusion Process



❑ Fundamentals of Diffusion Model

❖ How to train the Reverse Process? (Cont')

➢ ELBO Decomposition

- Recall $p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \quad q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$
- Substituting these into the ELBO gives:

$$\mathrm{ELBO}(x_0;\theta) = \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\right]$$

$$= \mathbb{E}_{q(x_{1:T}|x_0)}\left[\log p(x_T) + \sum_{t=1}^{T} \log p_\theta(x_{t-1} \mid x_t) - \sum_{t=1}^{T} \log q(x_t \mid x_{t-1})\right]$$

$$= \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}[\log p(x_T)]}_{\text{prior term}} + \sum_{t=1}^{T} \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}[\log p_\theta(x_{t-1} \mid x_t)]}_{\text{reverse transitions}} - \sum_{t=1}^{T} \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)}[\log q(x_t \mid x_{t-1})]}_{\text{forward transitions}}$$

Does not depend on the model parameters Θ.
When we take the gradient of the ELBO with respect to Θ during optimization, this term contributes nothing. Hence, it can be safely omitted from the loss function for training.

Law of total expectation & Markov property

$$\mathbb{E}_{q(x_t|x_0)}\left[\mathbb{E}_{q(x_{t-1}|x_t,x_0)}\left[\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t)\right]\right] = \int q(x_t \mid x_0)\left[\int q(x_{t-1} \mid x_t, x_0)\left(\log q(x_t \mid x_{t-1}) - \log p_\theta(x_{t-1} \mid x_t)\right) dx_{t-1}\right] dx_t$$

$$= \int q(x_t \mid x_0)\left[\int q(x_{t-1} \mid x_t, x_0) \log \frac{q(x_{t-1} \mid x_t, x_0)}{p_\theta(x_{t-1} \mid x_t)} dx_{t-1}\right] dx_t$$

$$= \mathbb{E}_{q(x_t|x_0)}\left[D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))\right]$$

- $L_t := \mathbb{E}_{q(x_t|x_0)}\left[D_{\mathrm{KL}}(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t))\right]$: measures how well the learned reverse process approximates the true posterior
- Negative ELBO as Training Objective:

$$-\mathrm{ELBO}(x_0;\theta) = L_T + \sum_{t=1}^{T-1} L_t + L_0$$

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$$
$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$L_t \approx \mathbb{E}_{q(x_t|x_0)}\left[\frac{1}{2\tilde{\beta}_t}\|\mu_\theta(x_t,t) - \tilde{\mu}_t(x_t,x_0)\|^2\right] + \mathrm{C}$$

$$L_t^{\text{simple}} = \mathbb{E}_{q(x_t|x_0)}\left[\|\epsilon_\theta(x_t,t) - \epsilon_t\|^2\right]$$

$$L_{\text{total}} = \sum_{t=1}^{T} \mathbb{E}_{q(x_t,x_0)}\left[\|\epsilon_\theta(x_t,t) - \epsilon_t\|^2\right] + \text{const}$$

❑ Fundamentals of Diffusion Model

❖ Sampling Process

➢ Summarise of Reverse process: Remove the noise added at time step 't' from X_t to X_{t-1}

- True posterior is $q(x_{t-1} \mid x_t)$ which is intractable, yet it can be replaced by $q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\big(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I\big)$
  - We know the closed form solutions of the parameters, meaning we can model them!
- Learned approximation is what I am modeling via $p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t),\ \sigma_\theta^2(x_t, t)\, I\big)$
  - It takes x_t and the time index 't' as inputs, and produces a mean vector $\mu_\theta(x_t,t)$ and a covariance matrix $\Sigma_\theta(x_t,t)$

➢ Once the model p_θ such as $p_\theta(x_{t-1} \mid x_t) \approx q(x_{t-1} \mid x_t)$ is trained, we can generate new samples via following procedure:

1. Draw an initial noise sample $x_T \sim p(x_T) = \mathcal{N}(0, I)$, which approximates the marginal $q(x_T)$ corrupted by the Forward process such that

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

2. For $t = T, T-1, \ldots, 1$, sample

$$x_{t-1} \sim p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1};\ \mu_\theta(x_t, t),\ \Sigma_\theta(x_t, t)\big)$$

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$$

3. After finishing down to $t = 1$, x_0 will be the our generated output as seen in Figure 7.

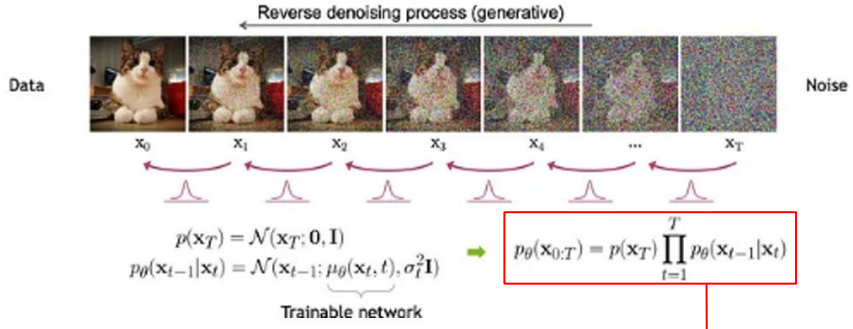
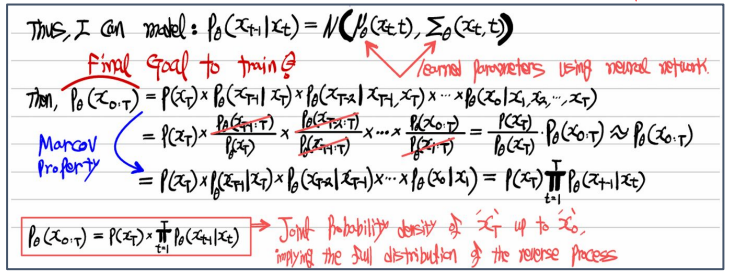


Figure 7: Reverse Diffusion Process



❑ Fundamentals of Diffusion Model

❖ Algorithm summary

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

❑ Model architecture

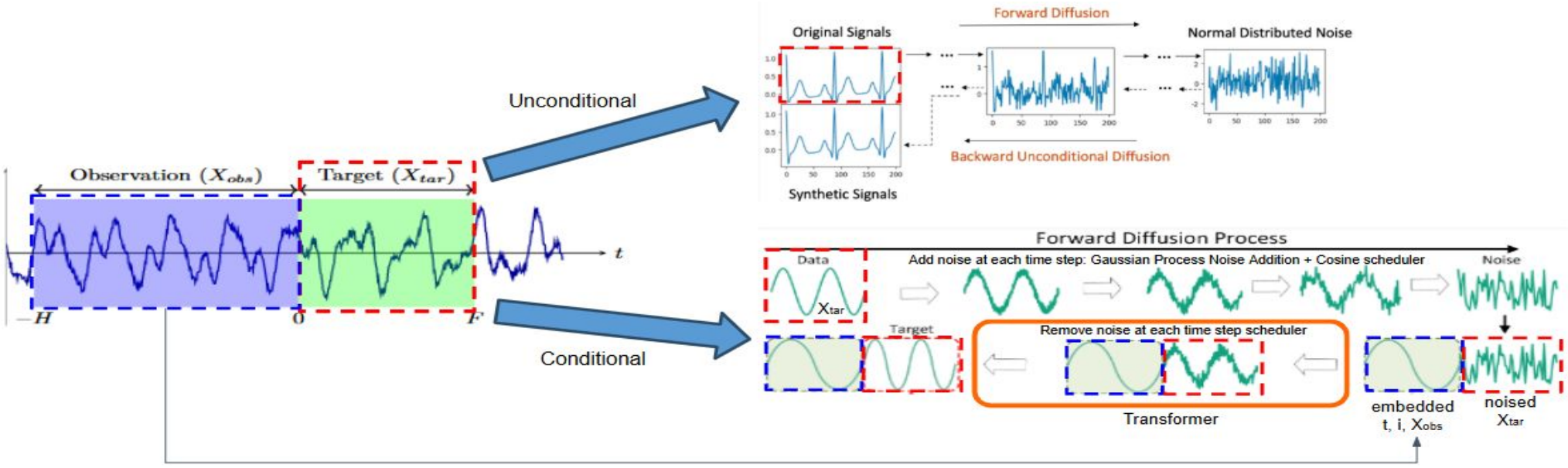
❖ Conditional Diffusion Model

➢ Forward Process

- Gaussian Process Noise Injection
- Control the magnitude of noise added at each time step: Cosine-scheduler

➢ Condition Encoder

- Positional Encoder(“Sinusoidal”): Encode time step(‘t’) & diffusion step(‘i’)
- 3 GRU(Gated Recurrent Unit) Encoder: Encode X_{obs}

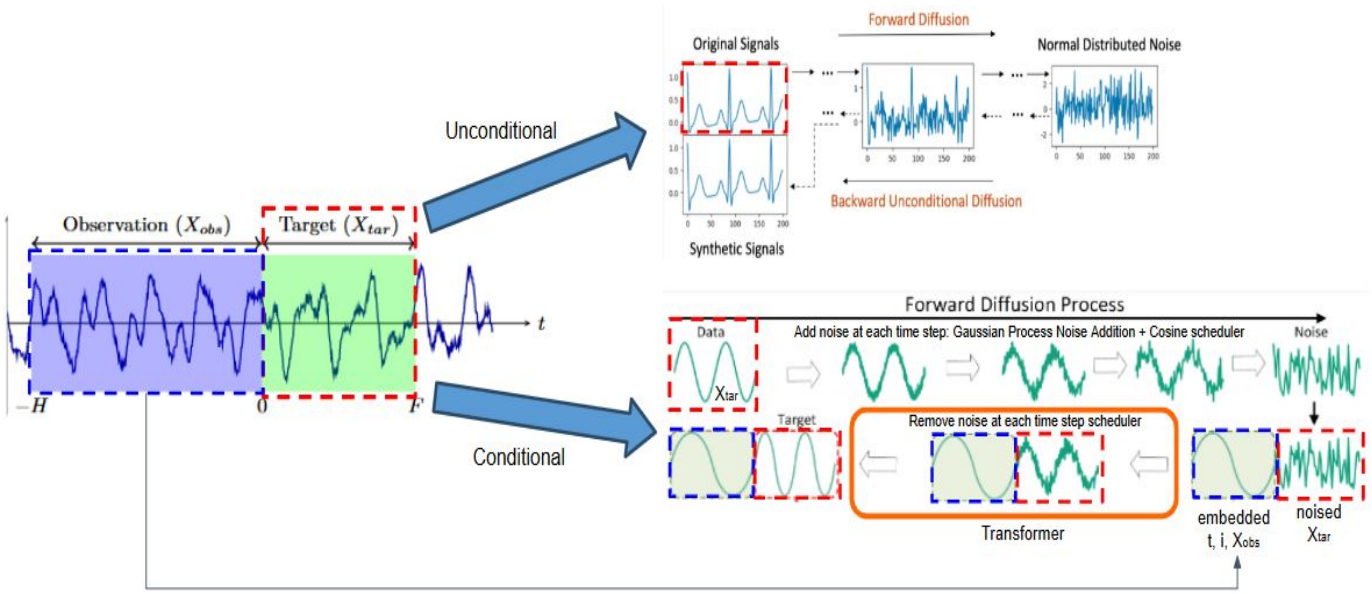


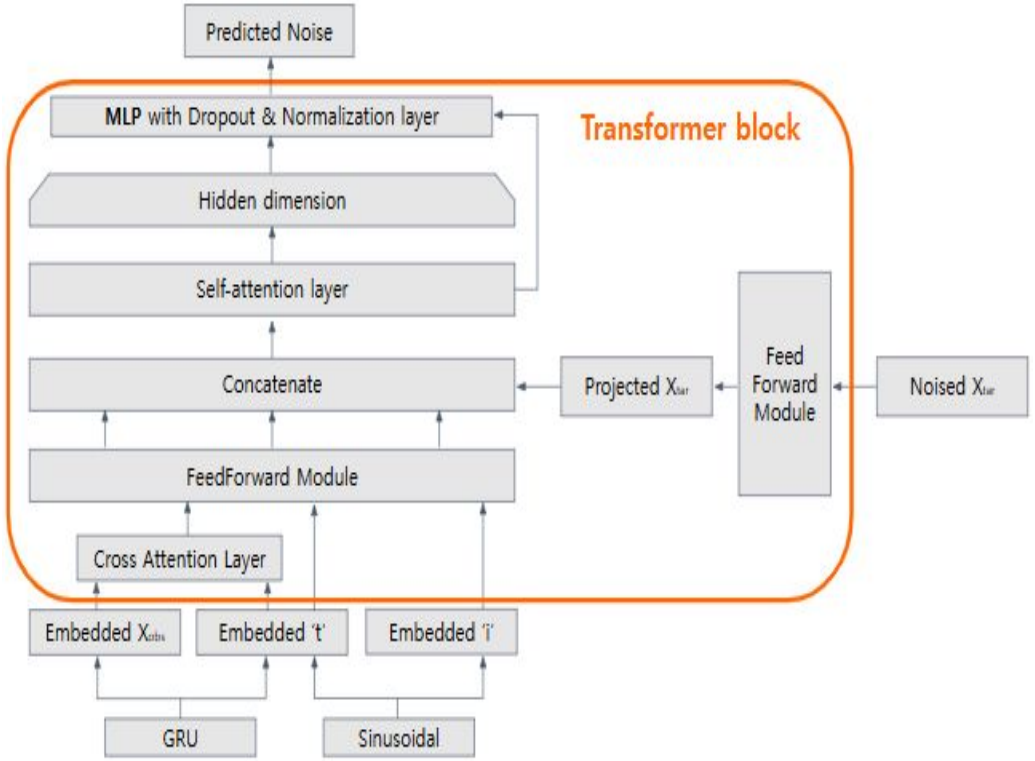
❑ Model architecture

❖ Conditional Diffusion Model (Cont')

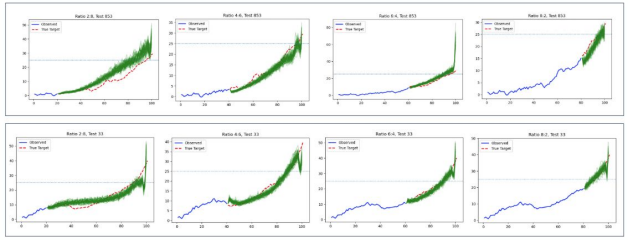
➢ NN architecture of reverse process: Transformer block

- 1st FeedForward: Simple MLP(Hidden layer + Relu) to project 'embedded conditions'
- Self-attention: Use multi-head attention with 'Residual connection'
- 2nd FeedForward: 2 Layers of MLP + Dropout + Normalization





❑ Model architecture



❖ Conditional Diffusion Model (Cont')

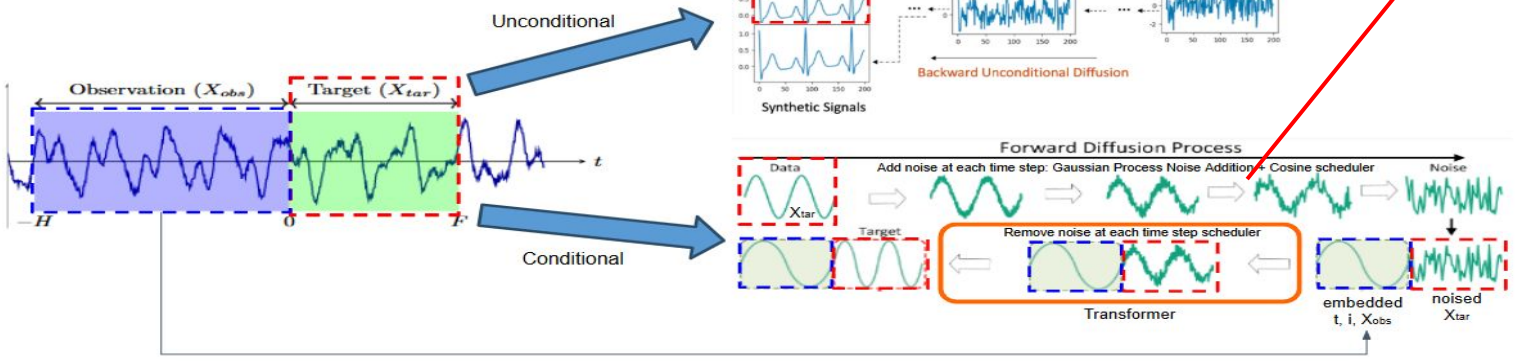
➢ Training

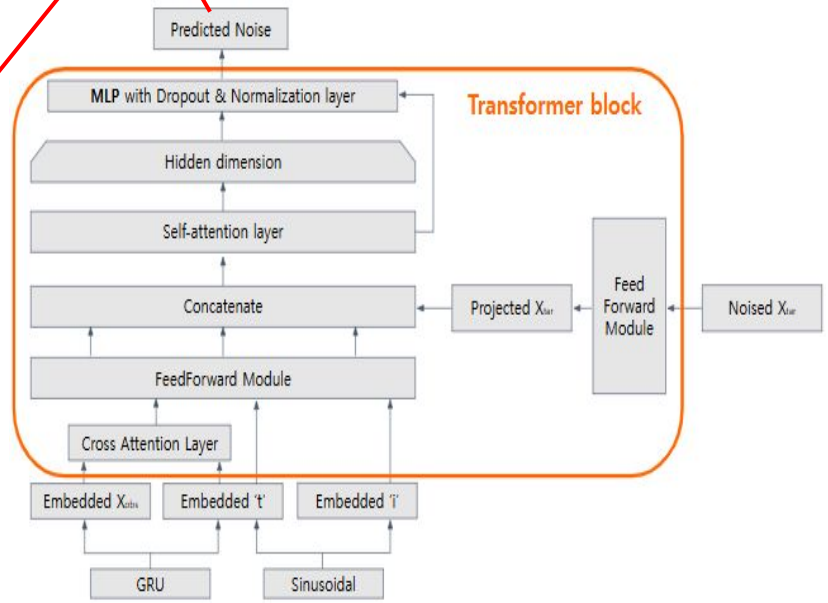
- Loss Function: MSE between the predicted noise and the true noise added to the target signal
- **Learning Rate Scheduler**: Increase 'lr' gradually up to 300 iterations and then exponentially decrease 'lr'
- **Early stopping** with a patience of 500 iterations

➢ Sampling

- Start with a random Gaussian Noise
- Remove noise at each time step: $x_t = \frac{1}{\sqrt{1-\beta}}\left(x_{t-1} - \beta \cdot \frac{\hat{\epsilon}}{\sqrt{1-\alpha}}\right) + \sqrt{\beta} \cdot z$

$$\mathcal{L} = \mathbb{E}\left[\|\hat{\epsilon} - \epsilon\|^2\right]$$

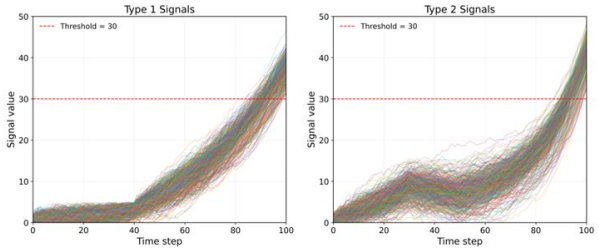




❑ Dataset

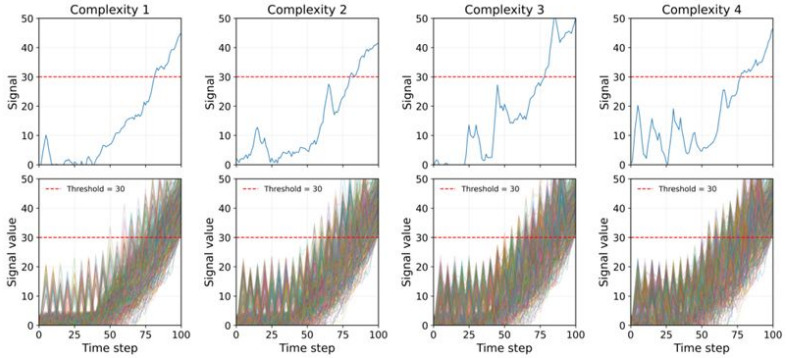
❖ Level 1. Typical degradation pattern

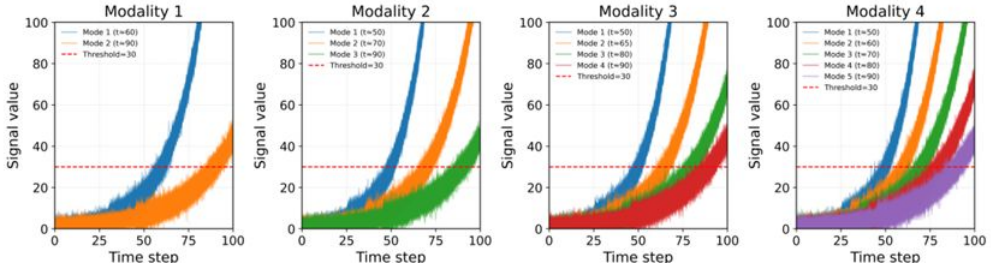
- ➢ **Type 1 (Monotone)**: Exponential growth, perturbed by Brownian-motion noise throughout
- ➢ **Type 2 (Non-monotone)**: Initial exponential growth, a short pullback, and a subsequent exponential rise, with Brownian perturbations throughout



❖ Level 2. Complex and irregular degradation pattern

- ➢ **Complexity 1~4:** Randomly inject K localized bumps on Type 1
  - ■ The levels are K = 1,2,3,4 for Complexity 1~4 respectively, and bumps sampled from U[10,20]
    - ● Induce abrupt local trend changes that mimic **thermal spikes, shocks, or control instabilities**
- ➢ **Modality 1~4:** Trajectories generated under multiple operating regimes with higher variance
  - ■ The number of extra regimes increases with the level: M=1,2,3,4 for Modality 1~4
    - ● Multimodal TTF distributions with heteroscedastic behavior, capturing **shifts in operating conditions**





❑ Two Critical points

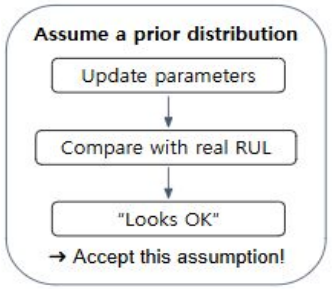
❖ Point 1. Bayesian Update vs. Diffusion

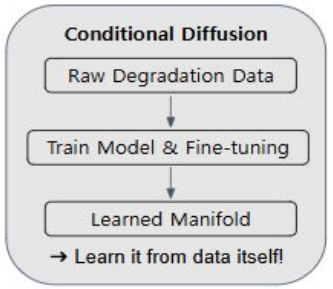
➢ Limitation of Bayesian Updating (IID and BM)

- Bayesian updating models **predefine a fixed functional form of signals** (e.g., exponential with i.i.d. or Brownian noise)
- **Update parameters** to best match the observed signal
  → But, this method still assume the fixed functional form (Analyze the signal pattern within the frame of the assumed form)
- The model performs parametric **model selection** (e.g., btw IID and BM), rather than learning the true degradation behavior
- This approach can be effective when the degradation signals exhibit smooth, monotonic trends under stable operating conditions—situations where the predefined functional form reasonably approximates the true degradation behavior. In such structured environments, the limited flexibility of Bayesian updating may help reduce overfitting and yield reliable predictions
- **However, what if the signal exhibits:**
  - Highly irregular, Non-Gaussian, or Multi-modal due to mixed failure machines?

➢ Our perspective: Learn the manifold itself and sample from the latent space!

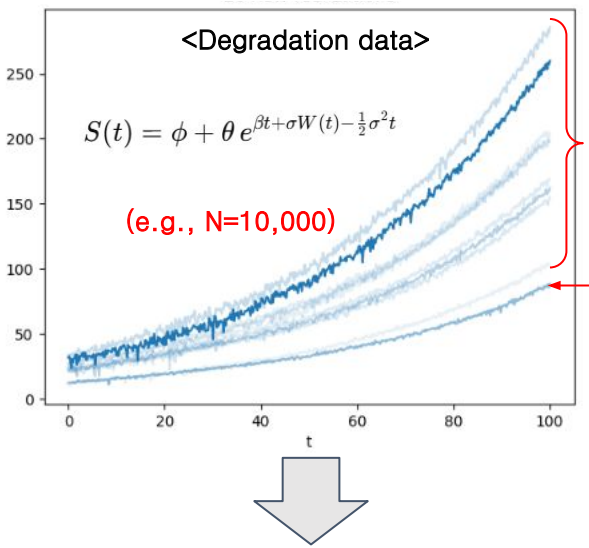
- Idea: Do not assume a prior distribution — instead, *learn* the distribution directly from the data!
- **Fundamentally different modeling philosophy**:
  → Not model selection, but **manifold learning** from raw data
- Can samples from complex signal dynamics such that:
  - Highly irregular, and even multimodal behaviors





❑ Design of Experiment for Point 1: Bayesian updating vs. Diffusion

❖ Bayesian Update Model: iid, BM



Train data

Test data

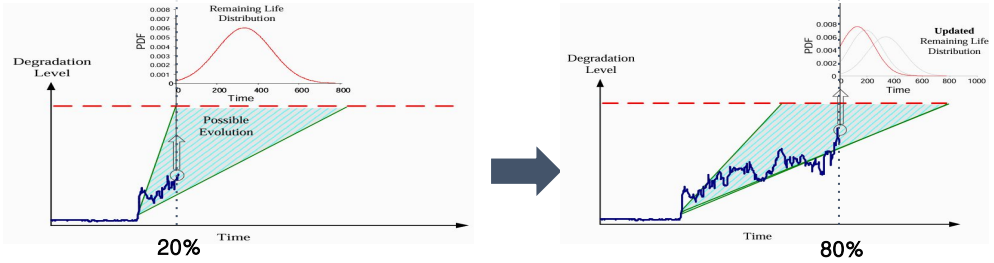
(# of total test data = T)

Step 1. Estimate the prior of parameter

$\theta' \sim N(\mu_{\theta'}, \sigma^2_{\theta'}), \quad \beta \sim N(\mu_\beta, \sigma^2_\beta)$
$(\ \theta' = \ln \theta\ )$

$$\pi(\theta') = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left\{-\frac{(\theta' - \mu_{\theta'})^2}{2\sigma_0^2}\right\}$$

$$\pi(\beta) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left\{-\frac{(\beta - \mu_\beta)^2}{2\sigma_1^2}\right\}$$

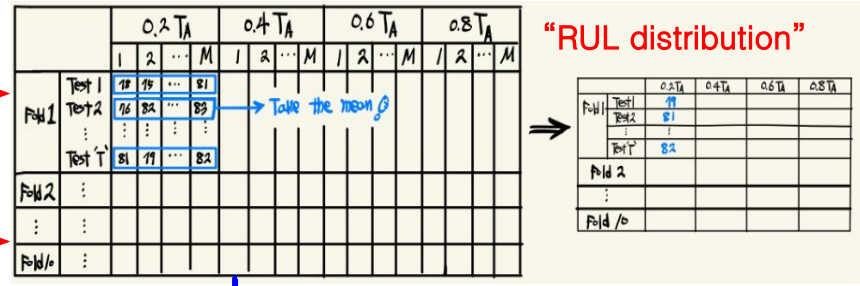


Step 2. Observe {L(t₁),…,L(tₖ)} & Compute the likelihoods
Update posterior distribution

Step 3. Derive RUL distribution in closed form solution per each test signal;
Compute RUL median via inverse-CDF;
Sample 'M' RUL per each test signal & Take mean!

		0.2 T_A	0.4 T_A	0.6 T_A	0.8 T_A
Fold 1	Test 1	82	81	79	75
	Test 2	18	77	75	72
	⋮	⋮	85	77	70
	Test T	16	75	71	68
Fold 2					
⋮					
Fold 10					

"10 Fold CV"

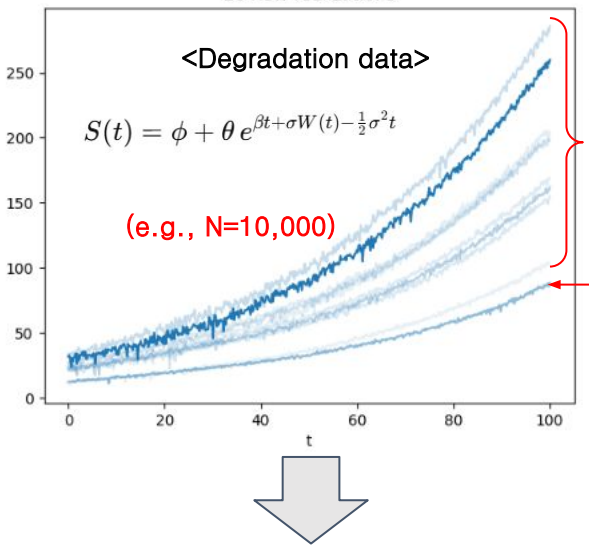


Step 4. Compare 'Predicted_mean_RUL' with the 'Real RUL' : RMSE, MAPE

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \qquad \text{MAPE} = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

❑ Design of Experiment for Point 1

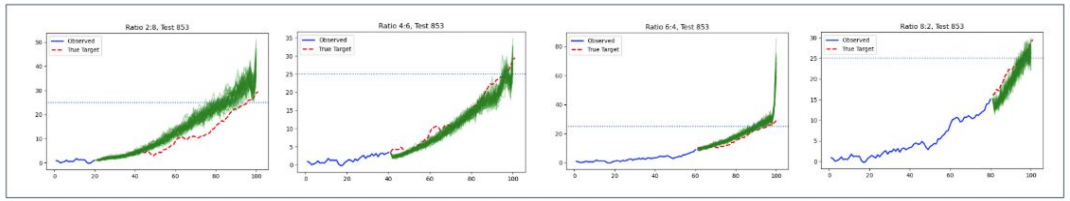
❖ Diffusion Model



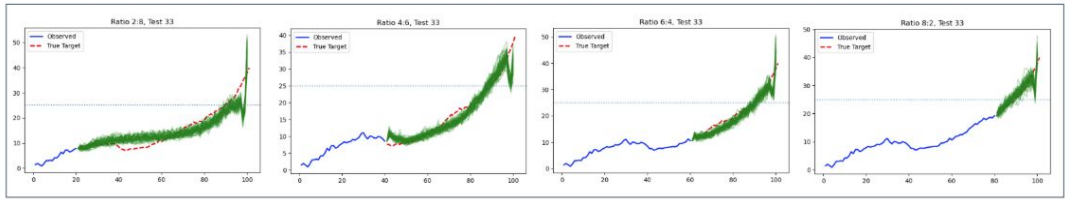
Step 1. Train a Conditional Diffusion Model for each ratio (2:8, 4:6, 6:4, and 8:2)

(# of total test data = T)

Type 1

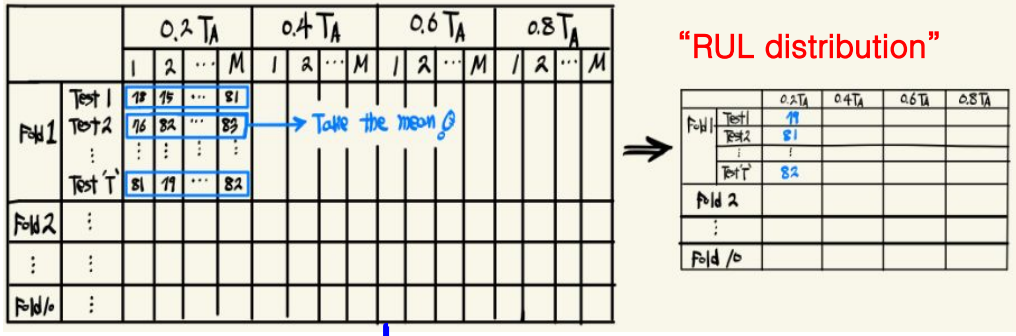


Type 2



Step 2. Use condition(observatory signals) as input for each ratio & Generate 'M' samples(target signals) per each ratio

Step 3. Compute RUL by taking mean of 'M' sample values



		0.2 T_A	0.4 T_A	0.6 T_A	0.8 T_A
Fold 1	Test 1	82	81	79	75
	Test 2	78	77	75	72
	⋮	⋮	85	77	70
	Test T	76	75	71	68
Fold 2					
⋮					
Fold 10					

"10 Fold CV"

Step4. Compare 'Predicted_mean_RUL' with the 'Real RUL' : RMSE, MAPE

❑ Two Critical points

❖ Point 2. M/L based RUL/TTF Prediction Model vs. Diffusion Model

➢ Limitation of PCA-based M/L method

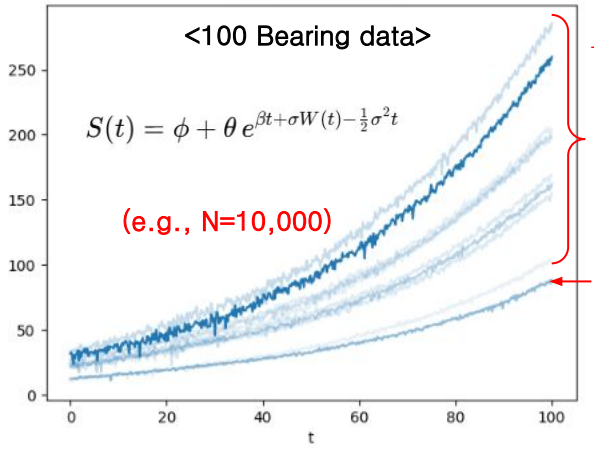
- ‘PCA’ projects original high-dimension into a lower dimension in **“linear subspace”**
  - **Cannot preserve complex, curved manifolds of degradation trajectories**
- ‘Log-normal regression’ assumes a smooth, unimodal, and skewed distribution
  - Works only when failure times follow a single, predictable trend!
    - ➢ Struggles with non-regular degradation patterns like: Heavy-tailed behavior, Multi-modal patterns
  - **‘Log-normal regression’, after PCA, models the signal pattern in linear context only**

➢ Strength of Diffusion model

- **With sufficiently large network ϵ_ϕ, any continuous data pattern can be approximated to high precision**
  - The more complex the real pattern is, the better Diffusion Model works!
  - It can directly reflect the manifold including non-regularity

❑ Design of Experiment for Point 2: M/L-based model vs. Diffusion

❖ Classical Method: PCA + Lognormal regression



Train data

Test data

Step 1. PCA on X_{obs} of train data(X_{train}) with 95% variational explain
for each ratio model (20, 40, 60, 80%)
Fit a log-normal regression model with [$X_{obs\text{-}after\text{-}PCA}$, log(RUL)]

$$\ln(\mathrm{RUL}) = w^\top \left(\phi_1^\top X_{\mathrm{obs}}, \ldots, \phi_k^\top X_{\mathrm{obs}}\right) + b + \eta, \quad \eta \sim N(0, \sigma^2)$$

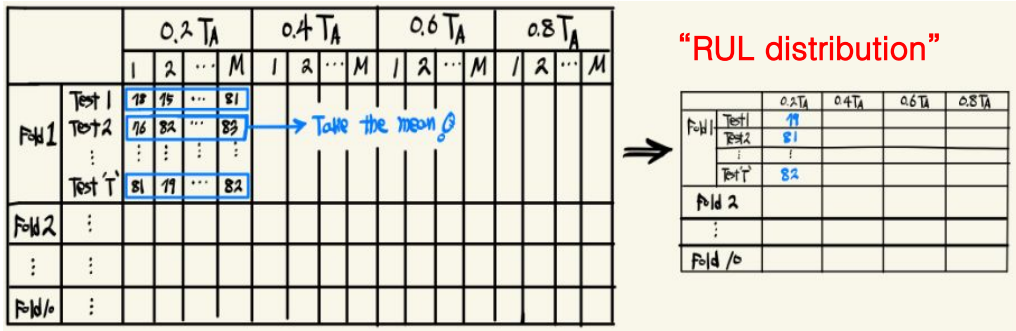
(# of total test data = T)

Step 2. Project X_{obs} of X_{test} onto PCA-space using eigenvector(ϕ_k)
Compute the prediction by applying $X_{obs\text{-}after\text{-}PCA}$ onto the log-normal model: $\ln(\mathrm{RUL}) \sim \mathcal{N}(\mu, \sigma^2)$ where $\mu = w^\top x + b$

Step 3. Sample 'M' RUL samples at each ratio & Take the mean

		0.2T_A	0.4T_A	0.6T_A	0.8T_A
Fold 1	Test 1	82	81	79	82
	Test 2	18	83	81	77
	⋮	⋮	85	77	82
	Test 'T'	16	81	82	71
Fold 2					
⋮					
Fold 10					

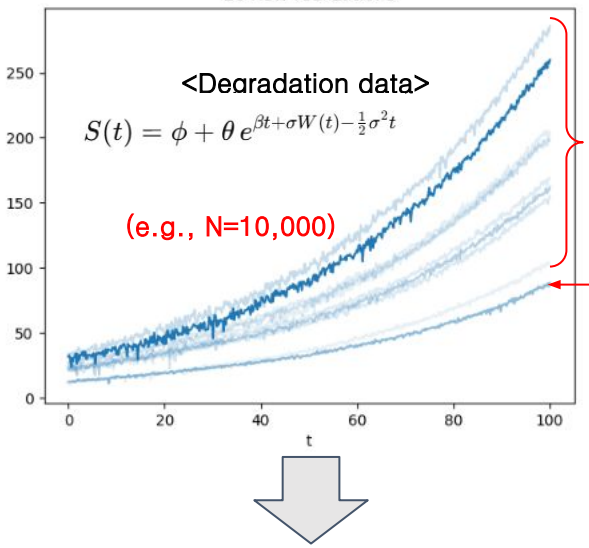
"10 Fold CV"



Step 5. Compare 'Predicted_mean_RUL' with the 'Real RUL' : RMSE, MAPE

❑ Design of Experiment for Point 2

❖ Diffusion Model



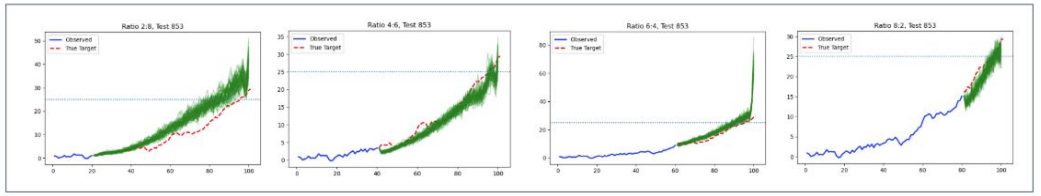
Step 1. Train a Conditional Diffusion Model for each ratio (2:8, 4:6, 6:4, and 8:2)

(# of total test data = T)

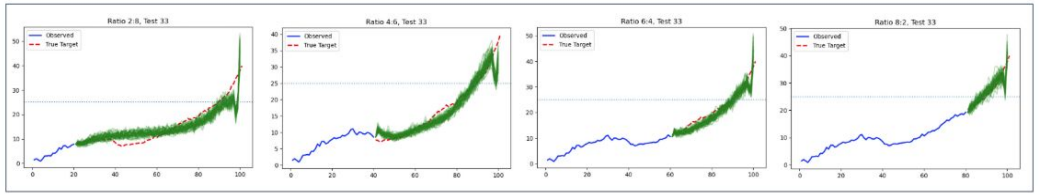
		0.2T_A	0.4T_A	0.6T_A	0.8T_A
Fold 1	Test 1	82	81	79	82
	Test 2	18	83	81	77
	⋮	⋮	85	77	82
	Test T	16	81	82	71
Fold 2					
⋮					
Fold 10					

"10 Fold CV"

Type 1

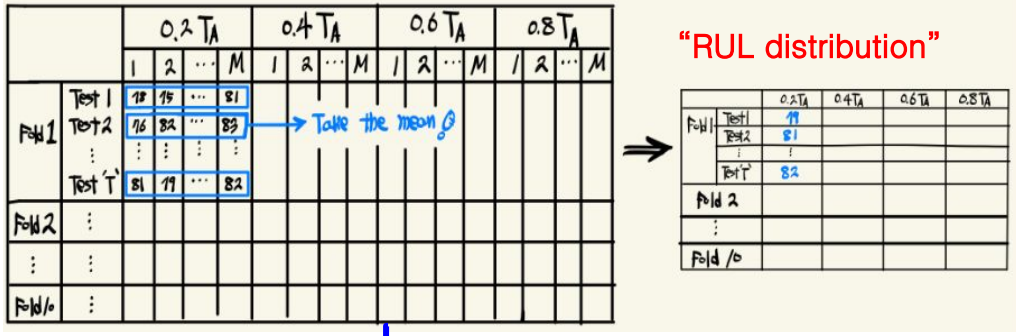


Type 2



Step 2. Use condition(observatory signals) as input for each ratio & Generate 'M' samples(target signals) per each ratio

Step 3. Compute RUL by taking mean of 'M' sample values



Step4. Compare 'Predicted_mean_RUL' with the 'Real RUL' : RMSE, MAPE

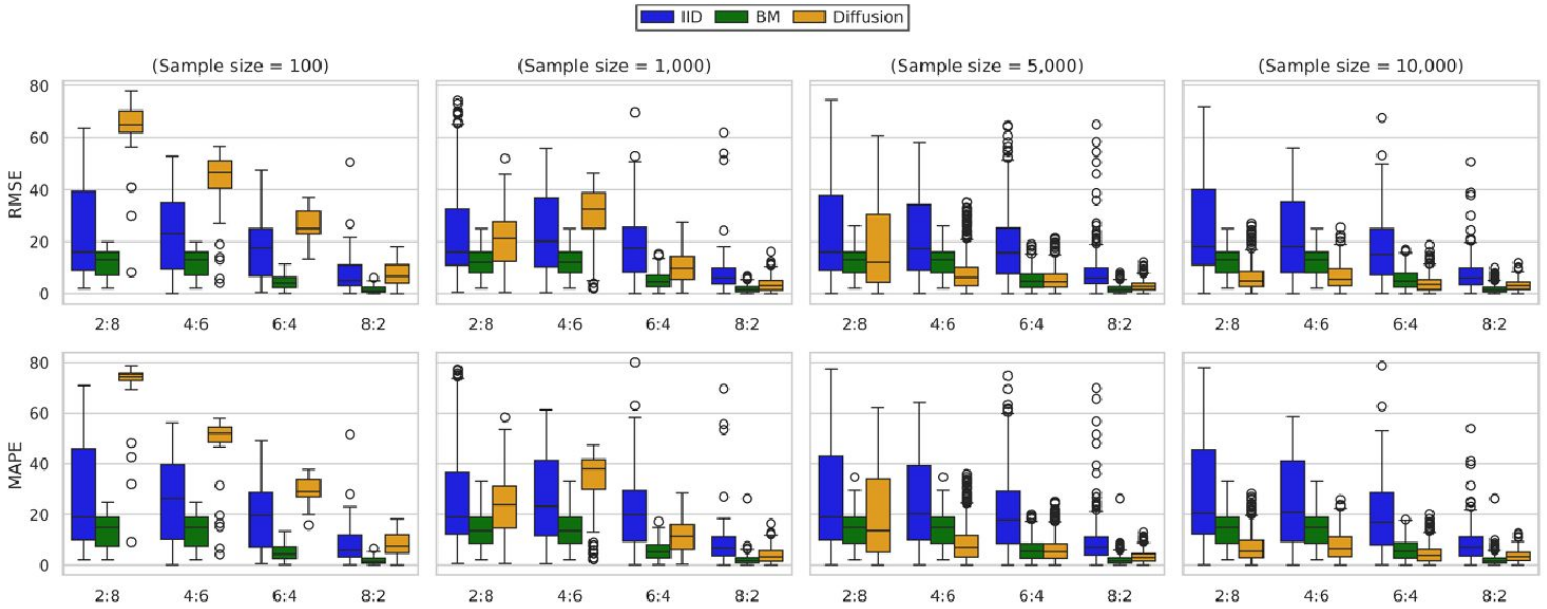
❑ Results Analysis

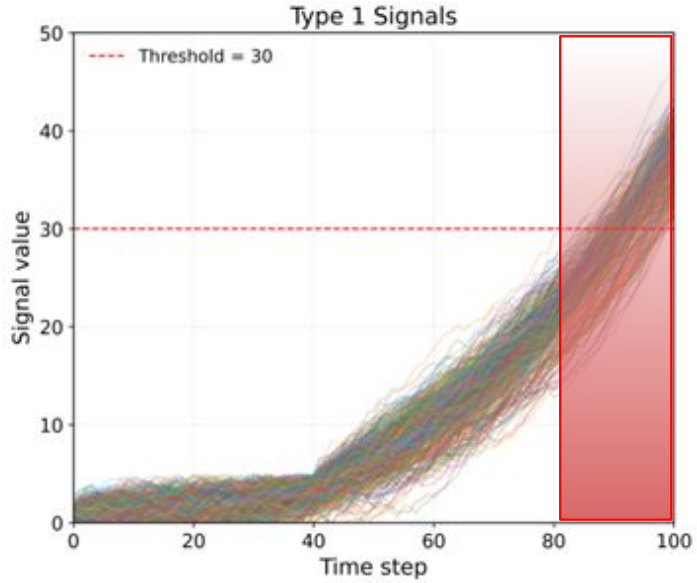
❖ Recall the focus of analysis

- ➢ Model performance comparison under varying signal characteristics
  - ■ To find **where Diffusion-based RUL/TTF model perform better than SOTA models and how much** better?
- ➢ Assessing **how performance evolves as more of the signal** is revealed

❖ Point 1 (Diffusion vs. Bayesian Updating) Results

- ➢ **In type 1,** Diffusion outperformed Bayesian updating **as data size increases (sample size ≥ 5000)**
  - ■ **Bayesian update:** competitive performance under limited training conditions (sample size < 5000)
    - ● **Due to the alignment btw their parametric assumption and the structure of the data**



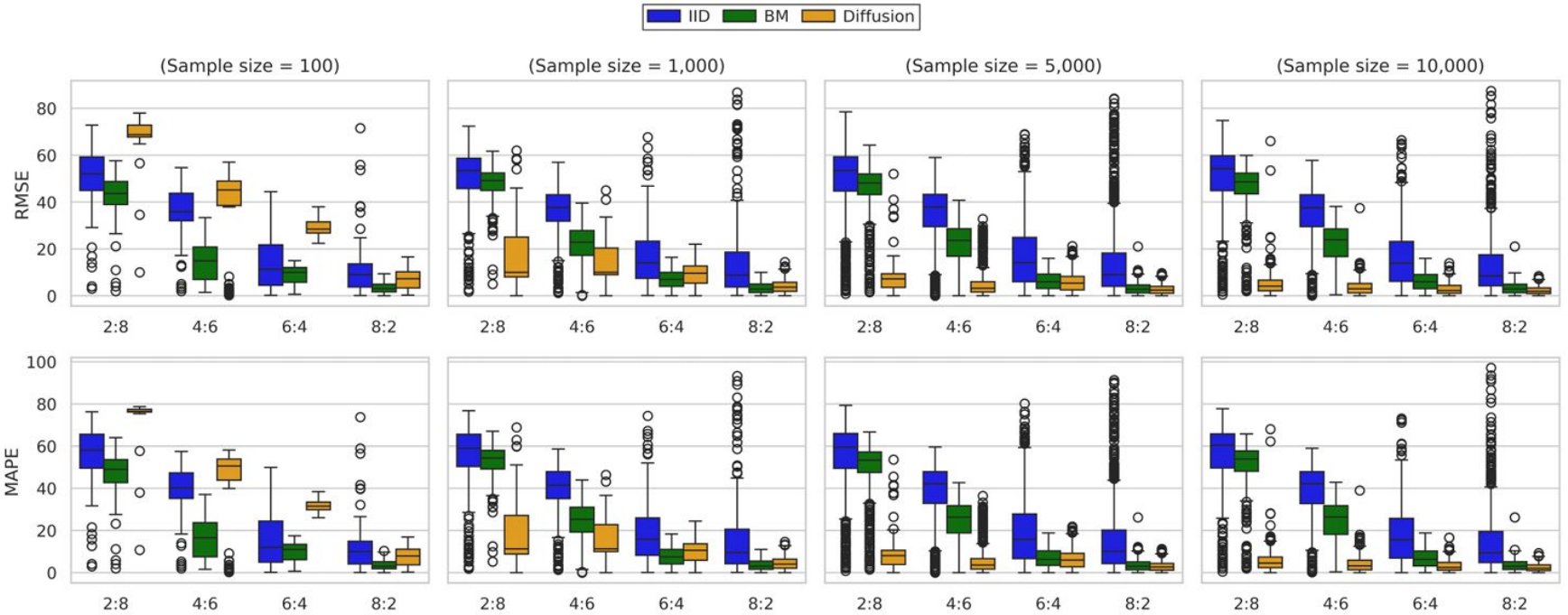


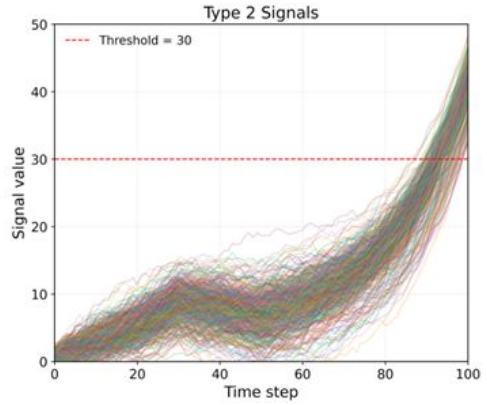
❑ Results Analysis

❖ Point 1 (Diffusion vs. Bayesian Updating) Results (Cont')

➢ **In type 2,** Diffusion demonstrates clear / robust superiority across all observation ratios (including 8:2)

- **Even with smaller data (≥ 1000),** Diffusion consistently yields lower RMSE and MAPE than BM model
  - **This performance gap widens as the training set becomes larger**;
  - Highlighting the ability of diffusion to model complex degradation dynamics
- BM model plateaus due to its heavy reliance on predefined priors





$$\Delta = \frac{\text{BM} - \text{Diff}}{\text{BM}} \times 100\,\%$$

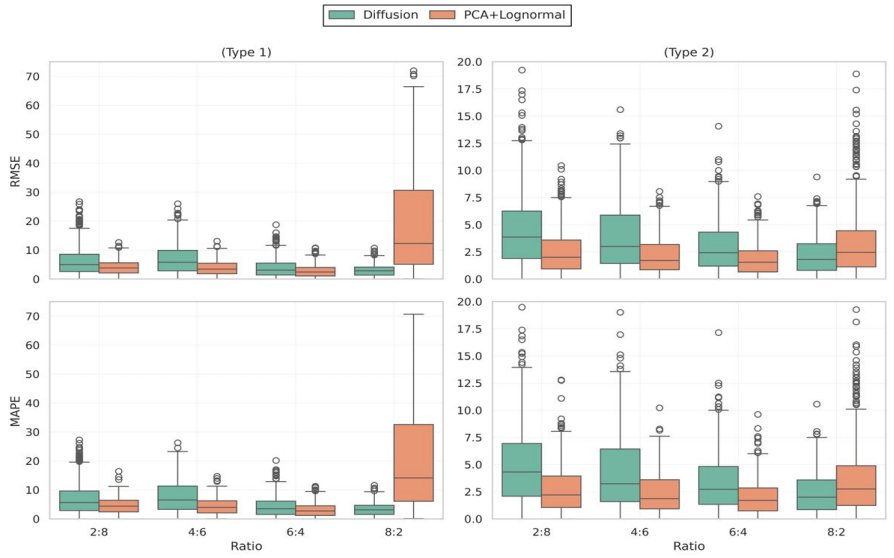
Setting	ΔRMSE (%)	ΔMAPE (%)
Type 1	10.04 ± 56.89	3.97 ± 68.77
Type 2	63.35 ± 27.08	63.12 ± 27.40

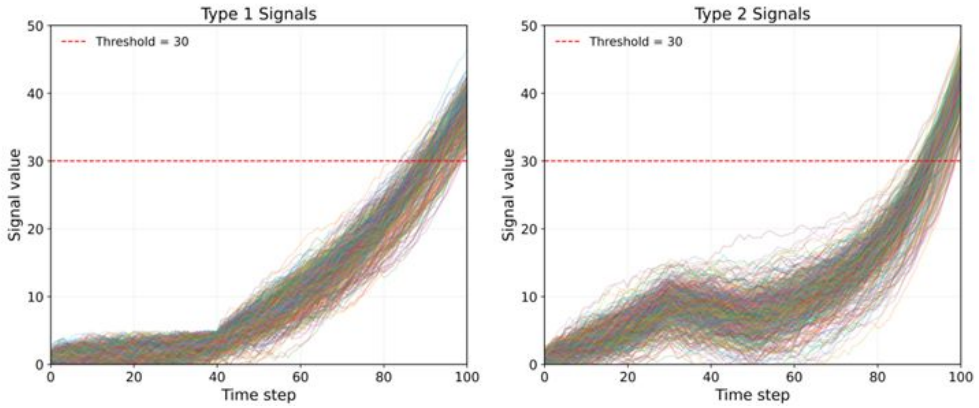
Aggregate improvements of Diffusion over BM when N≥10,000 (Mean ± Std)

❑ Results Analysis

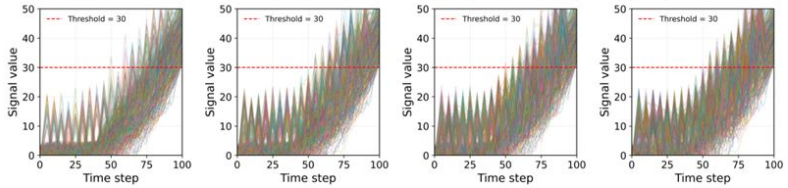
❖ Point 2 (Diffusion vs. PCA-based Lognormal) Results

- ➢ PCA+Lognormal performs well **in Type 1 and 2 between 2:8 and 6:4**
  - ■ Threshold crossings are **drift-dominated** with approximately homoscedastic variability
    - ● Residual-life distribution remains well behaved and a PCA feature set with a lognormal TTF fit is adequate!
- ➢ **At 8:2, Diffusion attains lower RMSE and MAPE in both types**
  - ■ Near-threshold crossings are **noise-dominated** and heteroscedastic, yielding skewed and unstable residual-life distributions that a linear PCA feature set with a parametric lognormal fit cannot represent!
- ➢ **Insight:** Diffusion-based model outperform M/L-base model when RUL distributions are noise-dominated!
  - ■ This will give us a chance to evaluate model's robustness under challenging conditions





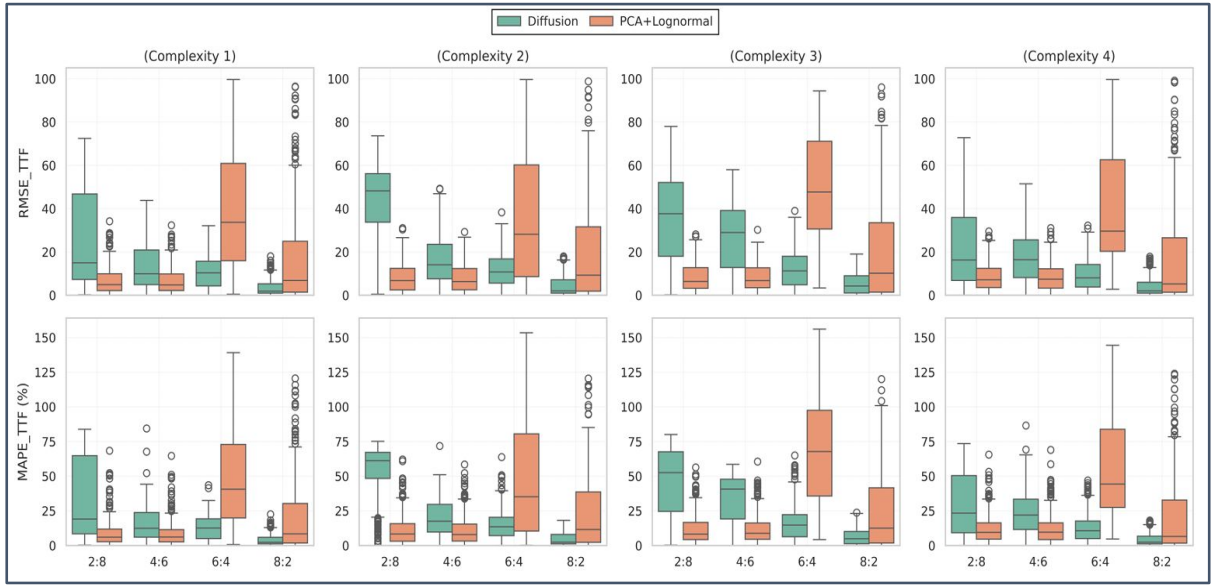
❑ Results Analysis



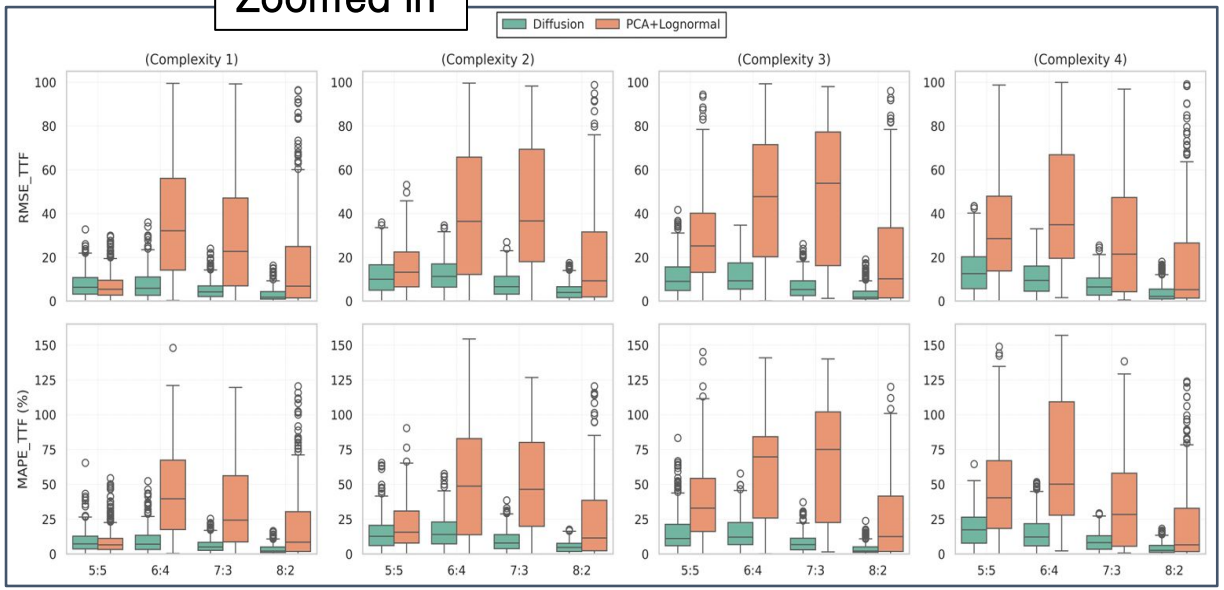
❖ Point 2 (Diffusion vs. PCA-based Lognormal) Results (Cont')

➢ Complexity 1–4 Results

- **Low-Mid ratios:** Threshold crossings are largely drift-dominated with approximately homoscedastic variability
  - PCA features with a lognormal TTF fit remain competitive!
- **From 6:4 ratio:** Crossings become noise-dominated and heteroscedastic
  - Diffusion achieves lower RMSE and MAPE, as it learns the dynamics near the threshold that are not well-captured by linear projections with a parametric lognormal fit



Zoomed in

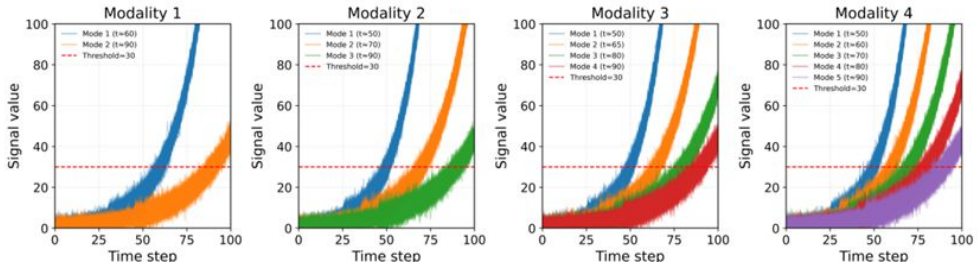


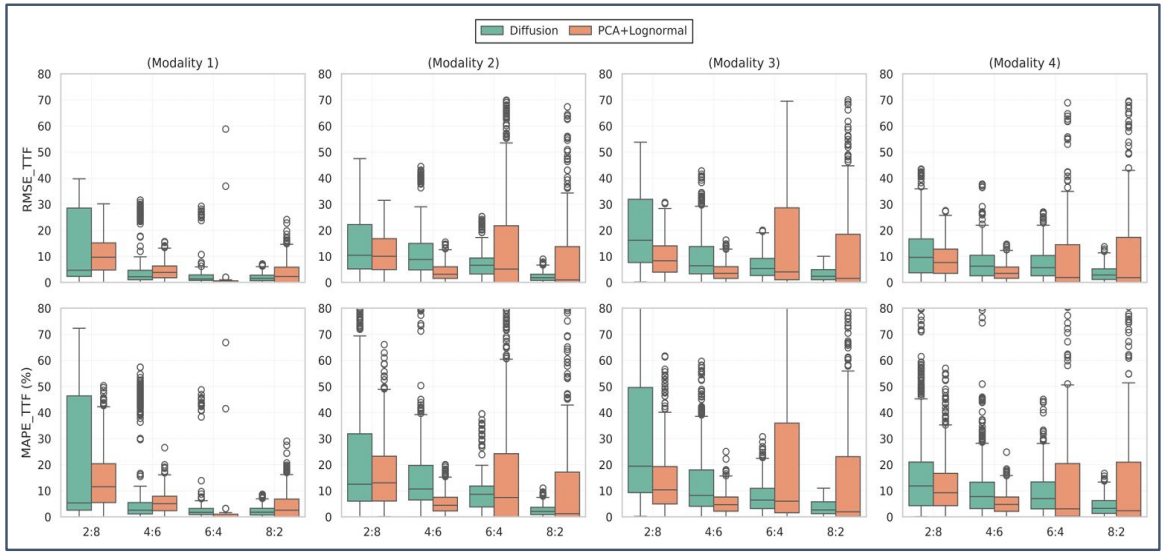
❑ Results Analysis

❖ Point 2 (Diffusion vs. PCA-based Lognormal) Results (Cont')

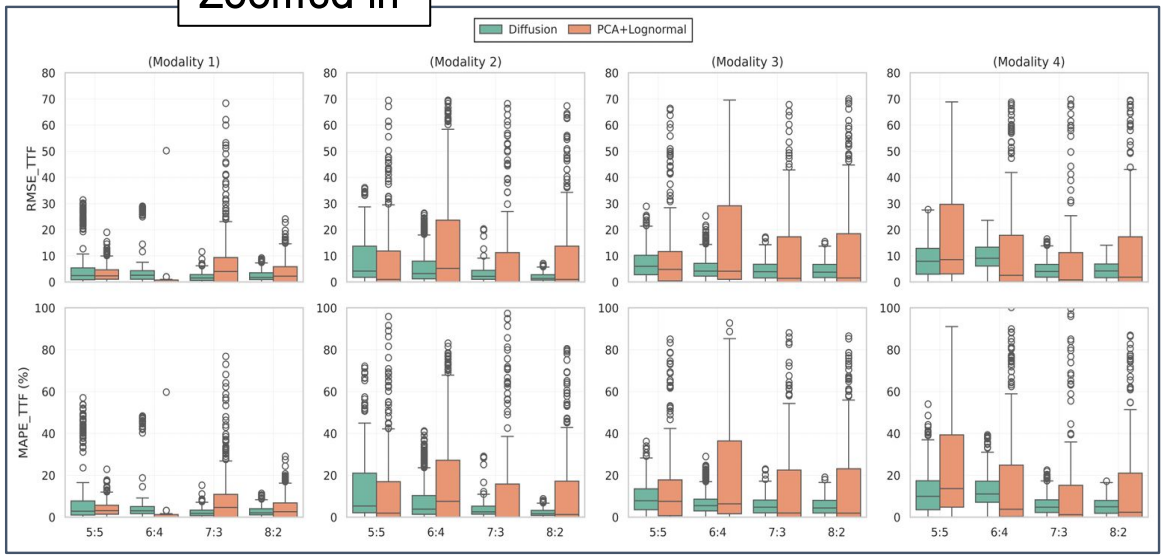
➢ Modality 1–4 Results

- **Modality 1**
  - Multi-modality ends near the 6:4 ratio
  - Noise-dominated: '4:6' and '8:2'
- **Modality 2~4**
  - Regime switching and sensor drift raise 'structural variance' and introduce 'between-regime mixtures'
  - Noise-dominated: '5:5' to '8:2'





Zoomed in



❑ Results Analysis

❖ Point 2 (Diffusion vs. PCA-based Lognormal) Results (Cont')

➢ In both Type 1 and 2, observation ratios from 2:8 to 6:4 are largely drift dominated with near-homoscedastic variability. PCA+lognormal baseline remains competitive. Near 8:2, the crossings become noise dominated and heteroscedastic, and diffusion attains lower RMSE and MAPE by modeling the near-threshold dynamics that linear projections with a parametric lognormal fit cannot capture.

- **At 8:2, Diffusion achieves about 90% lower error than PCA+lognormal in Type 1 and roughly 32% in Type 2**

Setting	ΔRMSE (%)	ΔMAPE (%)
Type 1	90.55 ± 2.10	90.05 ± 2.13
Type 2	33.14 ± 18.51	31.74 ± 19.42

➢ **Across Complexity 1 ~ 4,** the average improvement over 6:4 ~ 8:2 is 74.47 ± 6.17% (RMSE) and 75.94 ± 6.50% (MAPE),

- rising to 78.62 ± 5.09% and 79.14 ±5.75% in the zoomed in case of 5:5 ~ 8:2

➢ **Across Modality 1 ~ 4,** the corresponding gains are 57.21±19.17% and 56.53±20.51% for 6:4 ~ 8:2

- rising to 59.50±15.71% and 59.81 ± 17.54% for the zoomed in case of 5:5 ~ 8:2

Improvements of Diffusion over PCA+Lognormal on Complexity 1–4

Setting	ΔRMSE (%)	ΔMAPE (%)
Average at 6:4 and 8:2	74.47 ± 6.17	75.94 ± 6.50
Zoom-in (5:5–8:2)	78.62 ± 5.09	79.14 ± 5.75

Improvements of Diffusion over PCA+Lognormal on Modality 1–4

Setting	ΔRMSE (%)	ΔMAPE (%)
Average at 6:4 and 8:2	57.21 ± 19.17	56.53 ± 20.51
Zoom-in (5:5–8:2)	59.50 ± 15.71	59.81 ± 17.54

❑ Limitations and Further Research

❖ Evaluation relies on simulated datasets

- ➢ Although these simulations let me **control data complexity and variability**, they cannot yet fully reproduce real sensor noise, covariate shift, or operational interventions observed in practice
- ➢ To enhance external validity, I plan to use real-world data for generalization under realistic operating conditions
  - ■ IMS bearing data, NASA C-MAPSS, and etc.

❖ Diffusion-based inference is more computationally demanding than baselines

- ➢ Few-step samplers (e.g., DDIM), progressive distillation, and optimized noise schedules can reduce the number of reverse steps required

❖ Deployment often requires online/streaming inference as new observations arrive

- ➢ Federated learning for privacy-preserving deployment, Meta learning, Few-shot learning, and etc

Questions?