Deep Generative Models for hydrological time-series simulations

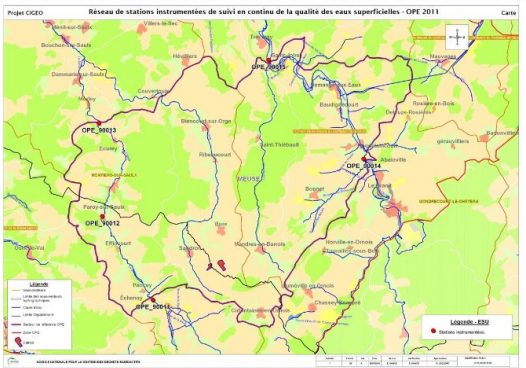
BHAVSAR Ferdinand[1], Lionel Benoit[1], Edith Gabriel[1]

Chaire Geolearning

April 8, 2026

[1]INRAE, Biostatistics and Spatial Processes (Biosp) team

Application: Water quality monitoring around the Cigéo site



Source: Andra

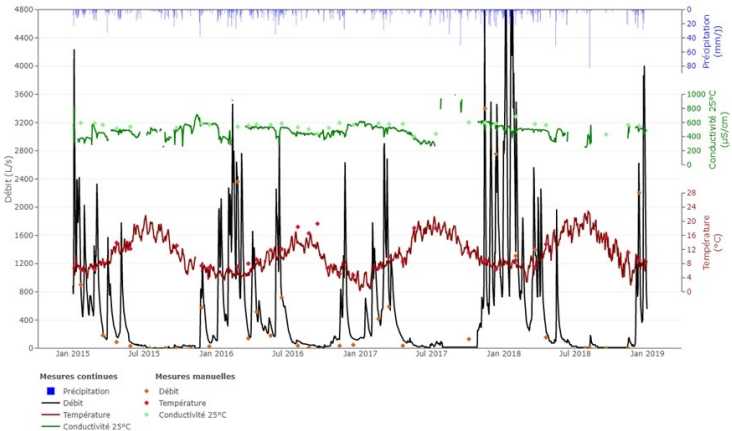
Application: Continuous monitoring of hydrological variables

1. Water level
2. Temperature
3. pH
4. Conductivity at 25°C
5. Dissolved O2
6. O2 saturation
7. Nitrates concentration
8. Turbidites
9. FDom (Fluorescent Dissolved Organic Matter) / Organical Carbon
10. PAH (Polycyclic Aromatic Hydrocarbon)
11. Chlorophyle-a
12. Dissolved Chlorides
13. Cyanobacterias

Source: Andra

Application: Continuous monitoring of hydrological variables

The data are **multivariate time-series** with many missing values (from 2012 to 2025, 4h between each observations).

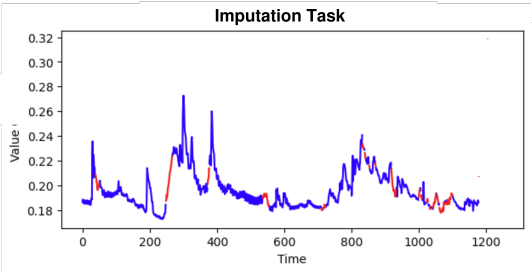


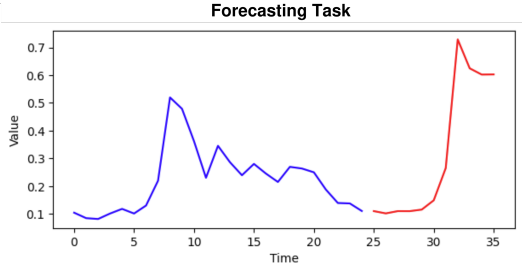
Source: Andra

Modeling multivariate hydrological variables with missing values

Postdoc objective: Develop a simulation method to simulate the target hydrological variables

We have two tasks to tackle:



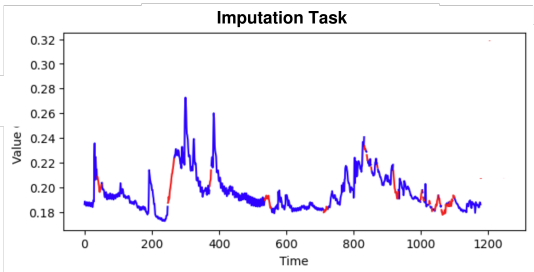


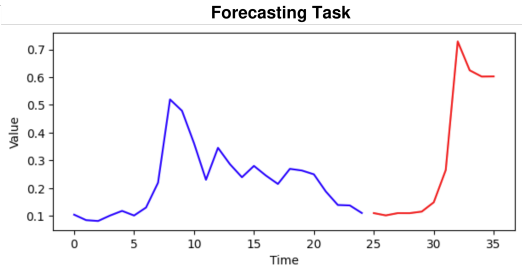
How do we find the red part ? ("Fill the gaps")

Modeling multivariate hydrological variables with missing values

Postdoc objective: Develop a simulation method to simulate the target hydrological variables

We have two tasks to tackle:





How do we find the red part ? ("Fill the gaps")

Deep generative learning: a transformation problem

Problem

Given a dataset, we want to sample new, never-seen before, convincing simulations with the same properties as the data from the dataset

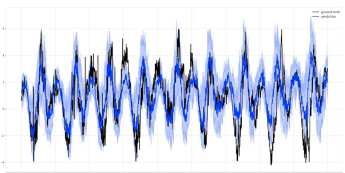
Deep generative learning: a transformation problem

Problem

Given a dataset, we want to sample new, never-seen before, convincing simulations with the same properties as the data from the dataset

Easy to sample Random Variable

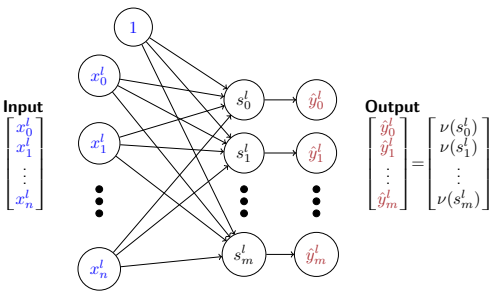
Complex and unknown Random Variable



Example of complex RV: time-series data

Generative models: Deep learning

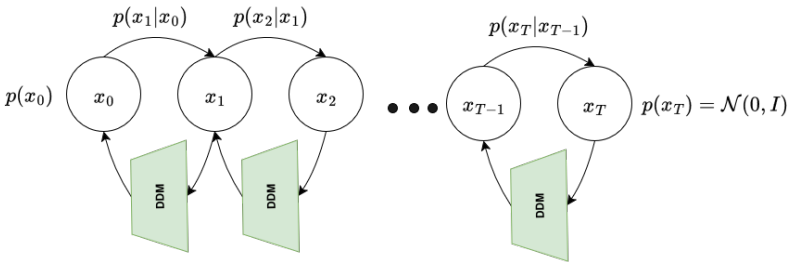
We approximate the transformation G with a neural network G_θ.



$$s_i^l = b_i^l + \sum_{j=0}^{n} W_{i,j}^l x_j^l$$

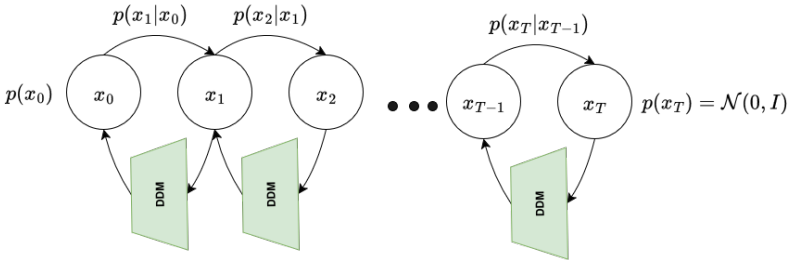
$$\begin{bmatrix} s_0^l \\ s_1^l \\ \vdots \\ s_m^l \end{bmatrix} = \begin{bmatrix} b_0^l & W_{0,0}^l & W_{0,2}^l & \dots & W_{0,n}^l \\ b_1^l & W_{1,0}^l & W_{1,2}^l & \dots & W_{1,n}^l \\ \vdots & \vdots & \vdots & & \\ b_m^l & W_{m,0}^l & W_{m,2}^l & \dots & W_{m,n}^l \end{bmatrix} \begin{bmatrix} 1 \\ x_0^l \\ x_1^l \\ \vdots \\ x_n^l \end{bmatrix}$$

Denoising Diffusion Probabilistic Models (Sohl-Dickstein et al., 2015; Ho et al., 2020)



$$\theta^* = \arg\min_\theta D_{\mathrm{KL}}\left(p_\theta(x_{t-h} \mid x_t) \,\|\, p(x_{t-h} \mid x_t, x_0)\right) \tag{1}$$

Denoising Diffusion Probabilistic Models (Sohl-Dickstein et al., 2015; Ho et al., 2020)



$$\theta^* = \arg\min_{\theta} D_{\mathrm{KL}}\big(p_\theta(x_{t-h} \mid x_t) \,\|\, p(x_{t-h} \mid x_t, x_0)\big) \tag{1}$$

CSDI: Conditional Score-based Diffusion Models for Irregular Time Series Imputation

We want a **state of the art** diffusion generative model, designed for both forecasting and imputation: **CSDI** (Tashiro et al., 2021).

Recent, good documentation, code available, good reputation in the community.

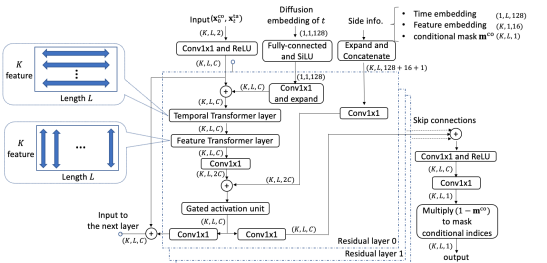
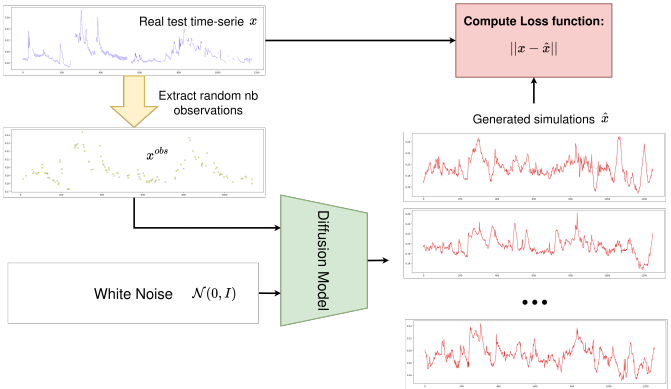


Figure 1: CSDI architecture overview (Tashiro et al., 2021)

CSDI training: Simplified overview



where x is the training time-serie, $\hat{x} = D_\theta(\epsilon, t, x^{obs})$ is the predicted time-serie, t is the diffusion time step, ϵ is Gaussian noise, $m(x)$ is the mask and x^{obs} are the observed values.

CSDI: Loss function

Training the model to learn the **reverse distribution**:

$$\theta^* = \arg\min_{\theta} D_{\mathrm{KL}}\big(p_\theta(x_{t-h} \mid x_t) \,\|\, p(x_{t-h} \mid x_t, x_0)\big) \tag{2}$$

It's the same as training the model to **denoise** x_t into $\hat{x} = D_\theta(x_t, t)$ (Ho et al., 2020):

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|x - \hat{x}_0\|^2\right] \tag{3}$$

For CSDI, the loss is computed only on the masked values (Tashiro et al., 2021), i.e.:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left\|x^{miss} - \hat{x}^{miss}\right\|^2\right] \tag{4}$$

where x^{miss} are the masked values, i.e. $x_{/x^{obs}}$, and $\hat{x}^{miss}$ are the corresponding predicted values.

CSDI: Loss function

Training the model to learn the **reverse distribution**:

$$\theta^* = \arg\min_{\theta} D_{\mathrm{KL}}\big(p_\theta(x_{t-h} \mid x_t) \,\|\, p(x_{t-h} \mid x_t, x_0)\big) \tag{2}$$

It's the same as training the model to **denoise** x_t into $\hat{x} = D_\theta(x_t, t)$ (Ho et al., 2020):

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|x - \hat{x}_0\|^2\right] \tag{3}$$

For CSDI, the loss is computed only on the masked values (Tashiro et al., 2021), i.e.:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left\|x^{miss} - \hat{x}^{miss}\right\|^2\right] \tag{4}$$

where x^{miss} are the masked values, i.e. $x_{/x^{obs}}$, and $\hat{x}^{miss}$ are the corresponding predicted values.

CSDI: Loss function

Training the model to learn the **reverse distribution**:

$$\theta^* = \arg\min_{\theta} D_{\mathrm{KL}}\big(p_\theta(x_{t-h} \mid x_t) \,\|\, p(x_{t-h} \mid x_t, x_0)\big) \tag{2}$$

It's the same as training the model to **denoise** x_t into $\hat{x} = D_\theta(x_t, t)$ (Ho et al., 2020):

$$\mathcal{L}(\theta) = \mathbb{E}\left[\|x - \hat{x}_0\|^2\right] \tag{3}$$

For CSDI, the loss is computed only on the masked values (Tashiro et al., 2021), i.e.:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\left\|x^{miss} - \hat{x}^{miss}\right\|^2\right] \tag{4}$$

where x^{miss} are the masked values, i.e. $x_{/x^{obs}}$, and $\hat{x}^{miss}$ are the corresponding predicted values.

Improving the architecture for our case

Complement the Transformer architecture with Convolutions (and a bit of Feature Engineering):

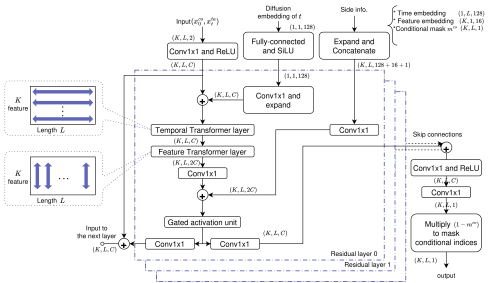


Figure 2: Our modified CSDI architecture

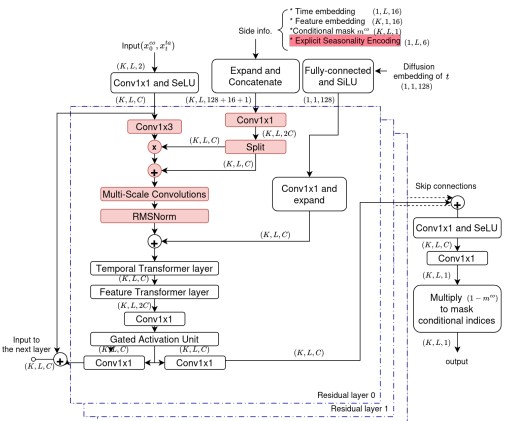
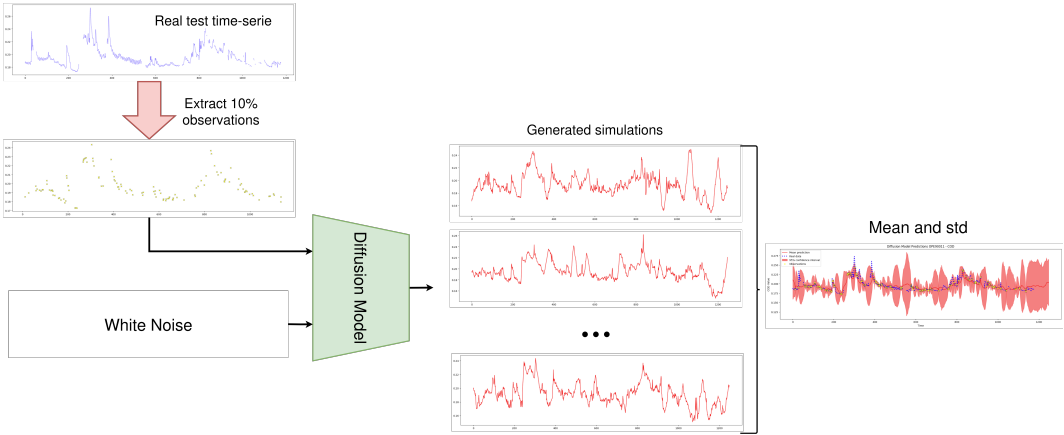


Figure 3: Our modified CSDI architecture

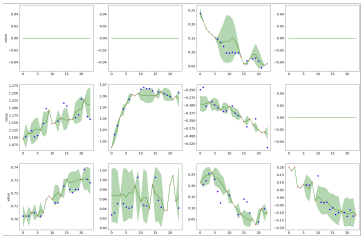
Evaluation methodology

Metrics used: RMSE, MAE, CRPS. 100 generated tests simulations.

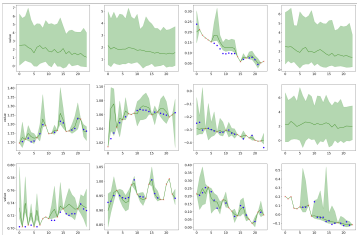


Imputation: some outputs visualized

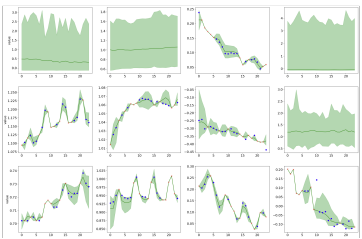
Gaussian Processes



Original CSDI



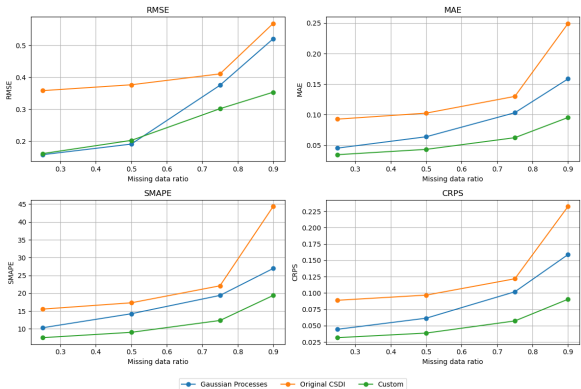
Our CSDI



Imputation: metrics

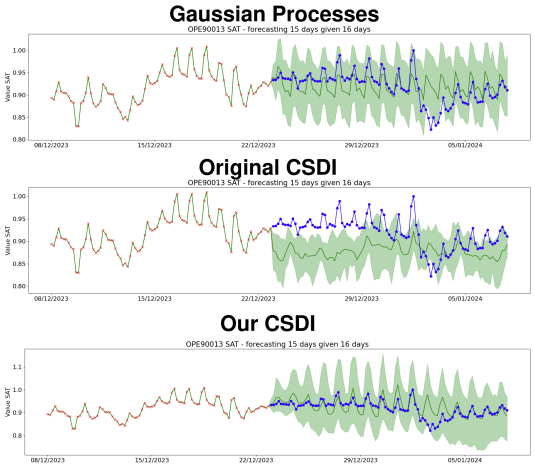
Imputation performance across models and missing ratios. Best value per metric highlighted in green.

Missing	Model	RMSE	MAE	SMAPE	CRPS
25%	Custom	0.1695	0.0354	7.6360	0.0321
	GP	0.1600	0.0456	10.2526	0.0447
	UNet	0.7249	0.5088	88.6534	0.4539
	CSDI	0.3567	0.0924	15.5531	0.0885
50%	Custom	0.1907	0.0412	8.8598	0.0374
	GP	0.2538	0.0660	14.3375	0.0647
	UNet	0.7941	0.5639	99.0945	0.5045
	CSDI	0.3665	0.1037	17.4125	0.0971
75%	Custom	0.2626	0.0575	12.0405	0.0527
	GP	0.3506	0.0975	19.1469	0.0976
	UNet	0.8952	0.6261	111.5064	0.5665
	CSDI	0.4245	0.1345	22.1077	0.1252
90%	Custom	0.3499	0.0968	19.0959	0.0924
	GP	0.4842	0.1610	26.5633	0.1608
	UNet	0.9820	0.6786	120.9972	0.6226
	CSDI	0.5336	0.2434	42.3684	0.2265



Forecasting: some outputs visualized

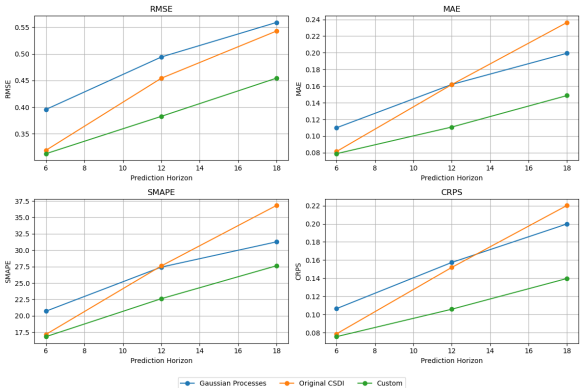
16 days seen, 15 days to predict



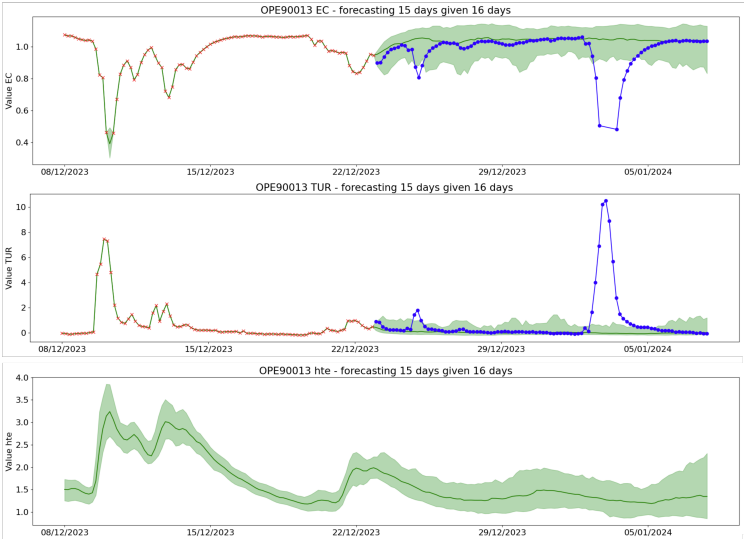
Forecasting: Metrics

Forecasting results for horizons 1, 2 and 3 days. Best values highlighted in green.

Horizon	Model	RMSE	MAE	SMAPE	CRPS
6	Custom	0.3110	0.0790	16.8951	0.0760
	GP	0.3951	0.1108	20.9929	0.1068
	UNet	0.9607	0.6691	120.1378	0.6174
	CSDI	0.3211	0.0819	17.2357	0.0791
12	Custom	0.3823	0.1113	22.6745	0.1061
	GP	0.4862	0.1609	27.1990	0.1563
	UNet	1.0015	0.6913	123.2085	0.6386
	CSDI	0.4524	0.1608	27.5070	0.1506
18	Custom	0.4548	0.1486	27.4929	0.1406
	GP	0.5589	0.1992	31.1799	0.1992
	UNet	1.0153	0.6985	124.4150	0.6448
	CSDI	0.5454	0.2374	36.9239	0.2213



A problem with the method: long term forecasting and unforeseen events

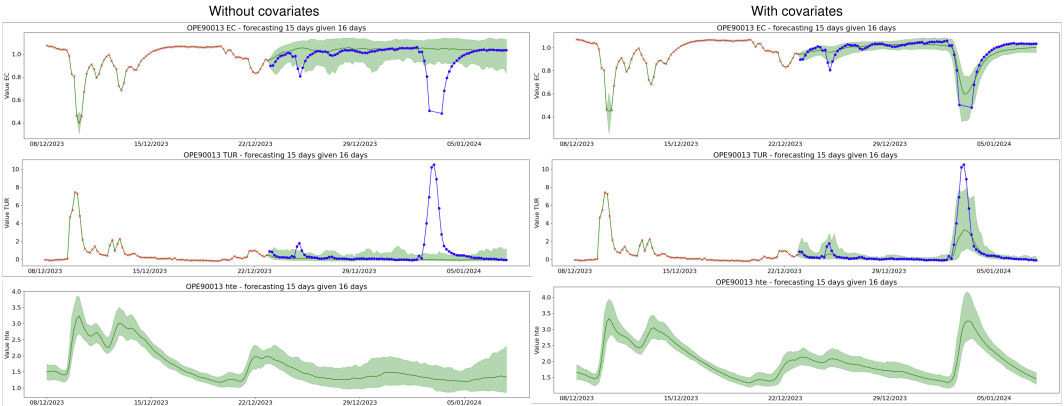


Daily SIM / SAFRAN-grid Data (weather)

- Data source: SAFRAN reanalysis from Météo-France (Vidal et al., 2010)

- Precipitation (liquid and solid) - daily totals
- Air temperature (min, max, or mean) at 2m
- Wind speed (e.g. 10m)
- Specific humidity or relative humidity at 2m
- Global / direct / diffuse solar radiation
- Snow- and soil-related variables: soil wetness index, soil water content, snow water equivalent, evapotranspiration

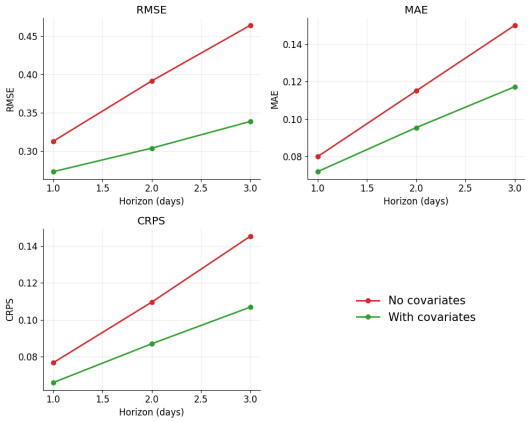
Forecasting using covariates: some outputs visualized



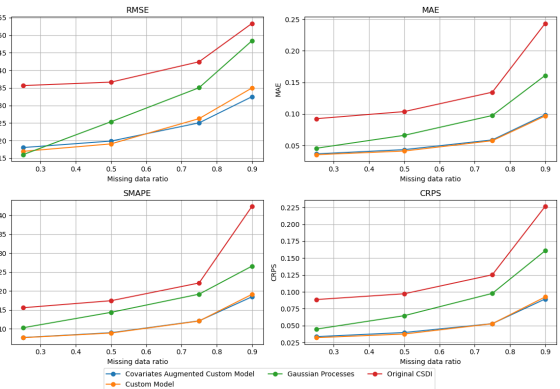
Forecasting using covariates: metrics

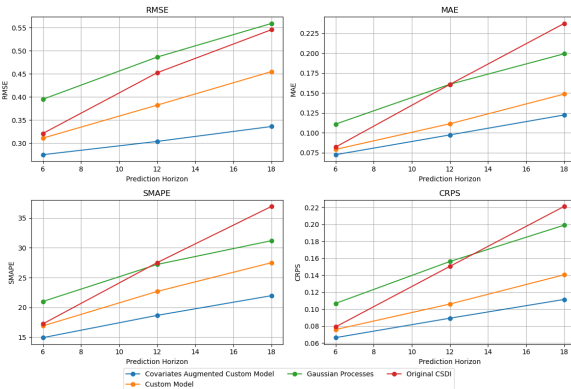
Forecasting results for horizons 1, 2 and 3 days. Best values highlighted in green.

Horizon	Setting	RMSE	MAE	CRPS
6	w/o Cov.	0.3129	0.0799	0.0768
	with Cov.	0.2733	0.0719	0.0661
12	w/o Cov.	0.3918	0.1150	0.1096
	with Cov.	0.3040	0.0954	0.0871
18	w/o Cov.	0.4645	0.1500	0.1452
	with Cov.	0.3389	0.1172	0.1069



Metrics


RMSE
MAE
SMAPE
CRPS
Missing data ratio
Covariates Augmented Custom Model
Gaussian Processes
Original CSDI
Custom Model


RMSE
MAE
SMAPE
CRPS
Prediction Horizon
Covariates Augmented Custom Model
Gaussian Processes
Original CSDI
Custom Model

Conclusion

We need to adapt the architecture of CSDI to our specific case study to obtain good results.
Our model seems to be performing better than the baselines for both **imputation and forecasting.**
Using relevant covariates (weather data) improves forecasting results.

Not shown: We used the model on another, synthetic dataset, with similar results.

Next steps:

- Sensibility Analysys for covariates selection (Yachouti et al., 2025)
- Take into account dry periods
- Take into account the spatial component of the data
- Interpolation on the river network

Conclusion

We need to adapt the architecture of CSDI to our specific case study to obtain good results. Our model seems to be performing better than the baselines for both **imputation and forecasting.**

Using relevant covariates (weather data) improves forecasting results.

Not shown: We used the model on another, synthetic dataset, with similar results.

Next steps:

- Sensibility Analysys for covariates selection (Yachouti et al., 2025)
- Take into account dry periods
- Take into account the spatial component of the data
- Interpolation on the river network

Conclusion

We need to adapt the architecture of CSDI to our specific case study to obtain good results.
Our model seems to be performing better than the baselines for both **imputation and forecasting.**
Using relevant covariates (weather data) improves forecasting results.

Not shown: We used the model on another, synthetic dataset, with similar results.

Next steps:

- Sensibility Analysys for covariates selection (Yachouti et al., 2025)
- Take into account dry periods
- Take into account the spatial component of the data
- Interpolation on the river network

Conclusion

We need to adapt the architecture of CSDI to our specific case study to obtain good results.
Our model seems to be performing better than the baselines for both **imputation and forecasting.**
Using relevant covariates (weather data) improves forecasting results.

Not shown: We used the model on another, synthetic dataset, with similar results.

Next steps:

- Sensibility Analysys for covariates selection (Yachouti et al., 2025)
- Take into account dry periods
- Take into account the spatial component of the data
- Interpolation on the river network

Ho, J., A. Jain, and P. Abbeel (2020). Denoising diffusion probabilistic models.

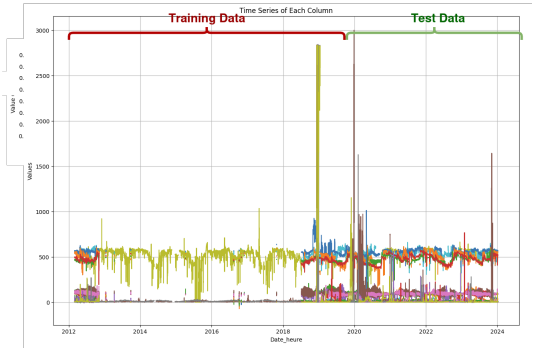
Sohl-Dickstein, J., E. A. Weiss, N. Maheswaranathan, and S. Ganguli (2015). Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR abs/1503.03585*.

Tashiro, Y., J. Song, Y. Song, and S. Ermon (2021). Csdi: Conditional score-based diffusion models for probabilistic time series imputation.

Vidal, J.-P., E. Martin, L. Franchistéguy, M. Baillon, and J.-M. Soubeyroux (2010, September). A 50-year high-resolution atmospheric reanalysis over France with the Safran system. *International Journal of Climatology 30*(11), P. 1627–1644. DOI: 10.1002/joc.2003. Publié en ligne dans Wiley InterScience (www.interscience.wiley.com). Version auteur dans fichier pdf attaché.

Yachouti, M., G. Perrin, and J. Garnier (2025, April). Towards History-aware Sensitivity Analysis For Time Series. working paper or preprint.

Adapting the data to deep learning

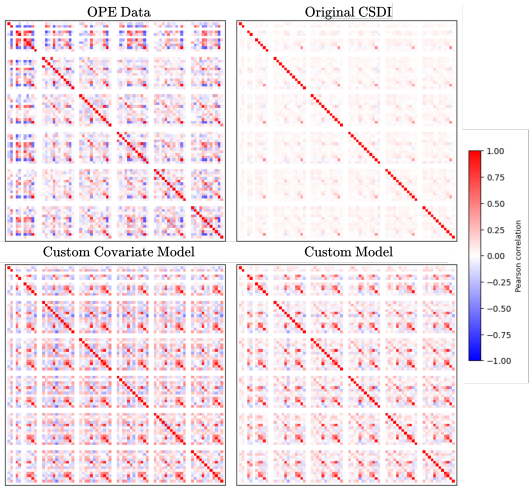


Deep learning does not like unormalized data. We do min-max normalization on the time-series along the time axis:

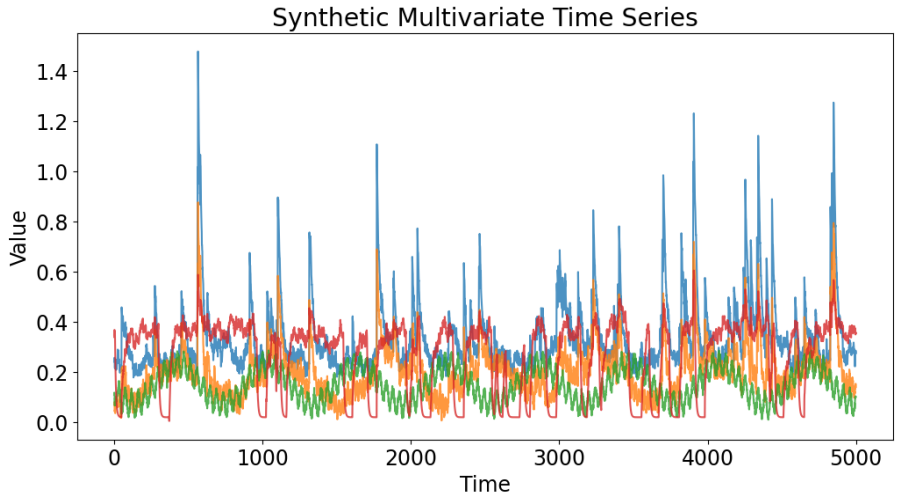
$$x^c_{norm} = \frac{x^c - x^c_{min}}{x^c_{max} - x^c_{min}} \tag{5}$$

where $c = \{0, C\}$, C is the number of variables.

Linear Correlations



Synthetic Data



Horizon	Model	RMSE	MAE	SMAPE	CRPS
6	Custom	0.0570 ± 0.0001	0.0250 ± 0.0000	7.5062 ± 0.0149	0.0608 ± 0.0006
	GP	0.0692 ± 0.0007	0.0371 ± 0.0002	11.7030 ± 0.0538	0.0858 ± 0.0005
	UNet	0.2500 ± 0.0004	0.2039 ± 0.0006	86.2811 ± 0.6309	0.4360 ± 0.0008
	NHITS	0.0622	0.0354	13.0673	–
	TSMixer-D	0.0587	0.0297	9.9398	–
	TSMixer-P	0.0571 ± 0.0001	0.0284 ± 0.0000	9.4563 ± 0.0106	0.0665 ± 0.0000
	CSDI	0.0558 ± 0.0001	0.0245 ± 0.0000	7.2570 ± 0.0125	0.0592 ± 0.0004
12	Custom	0.0768 ± 0.0001	0.0350 ± 0.0001	10.8655 ± 0.0317	0.0852 ± 0.0004
	GP	0.0967 ± 0.0002	0.0556 ± 0.0001	17.6600 ± 0.0449	0.1321 ± 0.0002
	UNet	0.2576 ± 0.0005	0.2100 ± 0.0003	89.2779 ± 0.2750	0.4458 ± 0.0006
	NHITS	0.0795	0.0436	15.3978	–
	TSMixer-D	0.0785	0.0401	13.5219	–
	TSMixer-P	0.0777 ± 0.0000	0.0385 ± 0.0000	12.9498 ± 0.0087	0.0917 ± 0.0000
	CSDI	0.2692 ± 0.0004	0.2348 ± 0.0004	118.9657 ± 0.2340	0.5976 ± 0.0011
18	Custom	0.0885 ± 0.0001	0.0425 ± 0.0001	13.6663 ± 0.0443	0.1021 ± 0.0007
	GP	0.1078 ± 0.0001	0.0646 ± 0.0000	20.7009 ± 0.0178	0.1609 ± 0.0001
	UNet	0.2594 ± 0.0005	0.2116 ± 0.0003	89.9733 ± 0.2486	0.4479 ± 0.0006
	NHITS	0.0901	0.0488	17.0870	–
	TSMixer-D	0.0895	0.0467	15.9980	–
	TSMixer-P	0.0897 ± 0.0001	0.0452 ± 0.0000	15.3929 ± 0.0194	0.1079 ± 0.0000
	CSDI	0.4034 ± 0.0006	0.3708 ± 0.0005	181.5056 ± 0.1379	0.9335 ± 0.0019

Figure 4: Synthetic dataset performance comparison. Best values are highlighted in green; second-best in blue. **Top**: Imputation metrics across missing ratios. **Bottom**: Forecasting metrics across horizons.