TIME SERIES CONTINUOUS MODELING FOR IMPUTATION AND FORECASTING WITH IMPLICIT NEURAL REPRESENTATIONS

March, 11th, Huawei Research

Etienne Le Naour, Louis Serrano, Léon Migus, Yuan Yin, Ghislain Agoua, Nicolas Baskiotis, Patrick Gallinari, Vincent Guigue

https://github.com/etiennelnr/timeflow

université PARIS-SACLAY

MIA PARIS-SACLAY EKINOCS

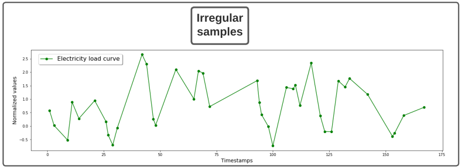
AGROPARISTECH Institut des Sciences et Industries du Vivant et de l'Environnement

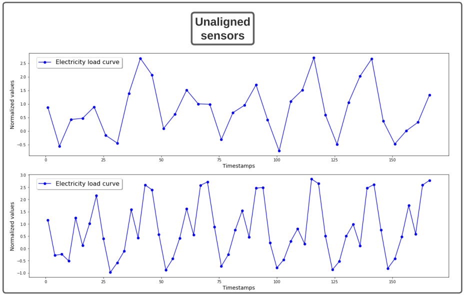
Time Series = continuous phenomena / observed partially

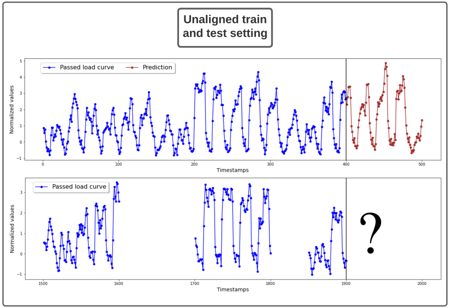
- Modeling Time Series as a continuous function

⇒ Deal with irregular sampling / unaligned sensors

⇒ Unified framework for Data imputation + Forecasting

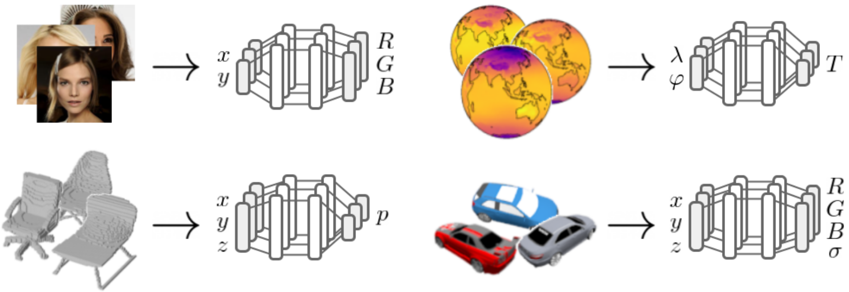






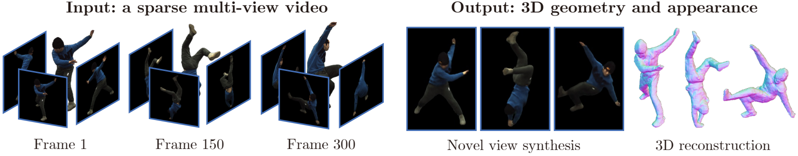
Technical options

- Gaussian Processes [Williams and Rasmussen, 2006]
- Neural Processes [Kim et al., 2019]
- Specific Architecture (e.g. mTAN) [Shukla and Marlin, 2021]
- Diffusion Model [Ho et al., 2020]
- Implicit Neural Representation (INR) [Dupont et al., 2022]

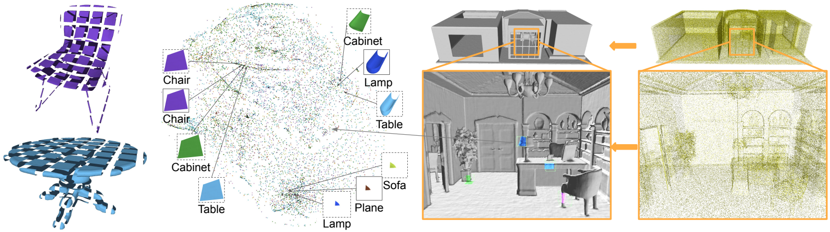


Implicit Neural Representation : a versatile solution

- Image compression [Dupont et al., 2021]
- 3D / animation modeling [Peng et al., 2021]



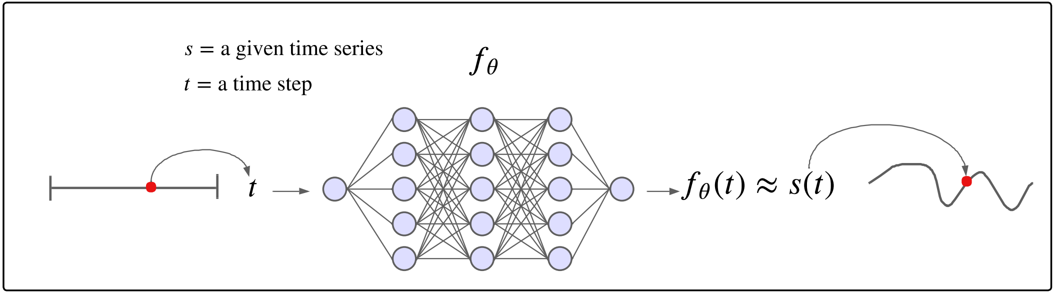
- 3D Scene / multiscale representation [Jiang et al., 2020]



(a) Training parts from ShapeNet. (b) t-SNE plot of part embeddings. (c) Reconstructing entire scenes with Local Implicit Grids

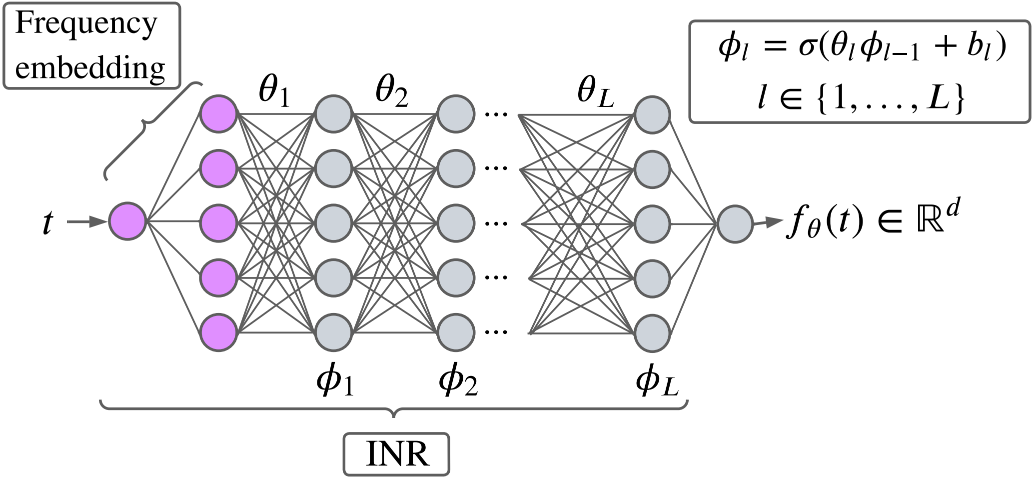
Implicit Neural Representation for Time Series

- A first attempt: DeepTime [Woo et al., 2022]
- Room for improvement:
  - Not designed for data imputation (forecasting only)
  - $\approx$ Ridge Regression on sampled Fourier descriptors



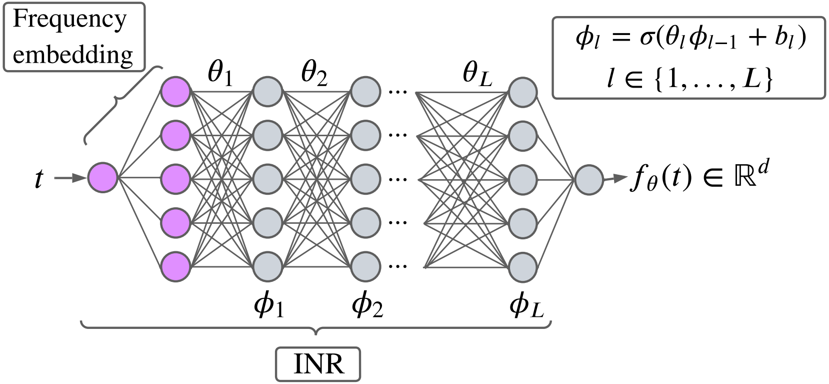
TimeFlow architecture

NeRF encoding illustration [Mildenhall et al., 2021]



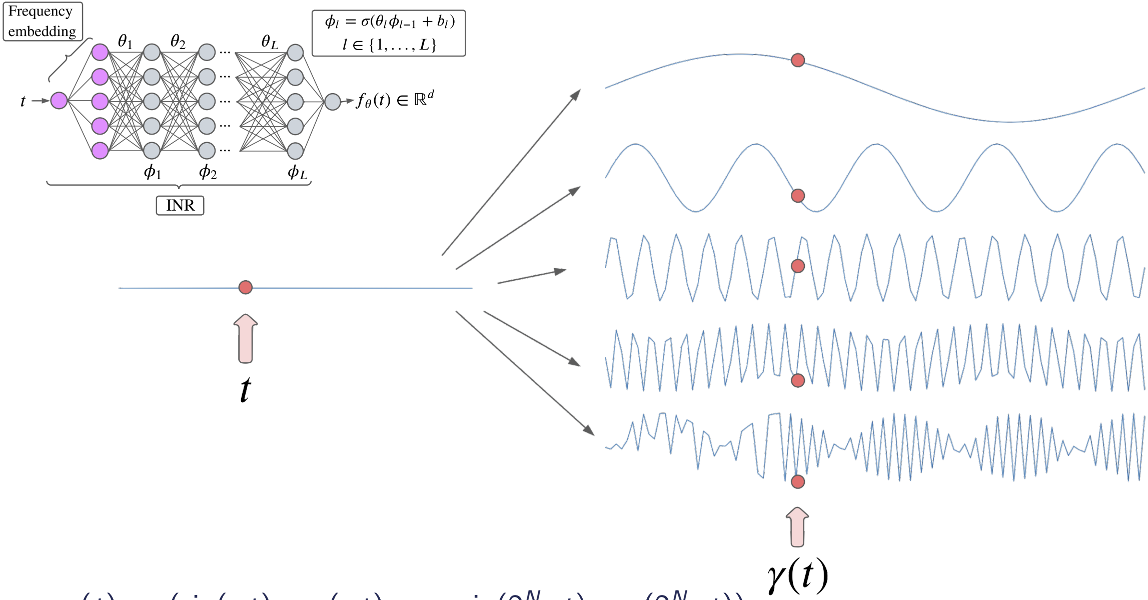
1. NeRF encoding : $t \rightarrow \gamma(t) \Rightarrow$ Fixed frequency description

NeRF encoding illustration [Mildenhall et al., 2021]



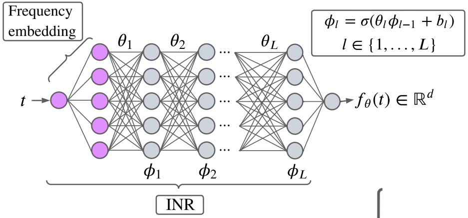
1. NeRF encoding : $t \rightarrow \gamma(t)$, N frequency bands
   $\gamma(t) := (\sin(\pi t), \cos(\pi t), \cdots, \sin(2^N \pi t), \cos(2^N \pi t))$
2. Then $\gamma(t) \rightarrow \mathrm{MLP}(\gamma(t); \theta)$ ($vs \approx$ Ridge Reg. in [Woo et al., 2022])
   Activation functions are ReLU (i.e. $ReLU(x) = max(0, x)$)

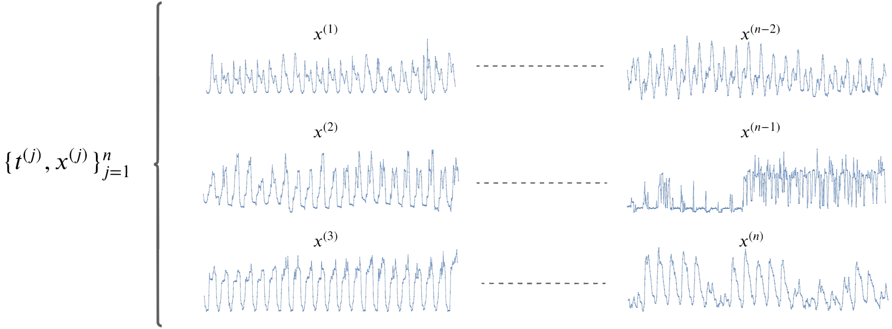
NeRF encoding illustration [Mildenhall et al., 2021]



$$\gamma(t) := (\sin(\pi t), \cos(\pi t), \cdots, \sin(2^N \pi t), \cos(2^N \pi t))$$

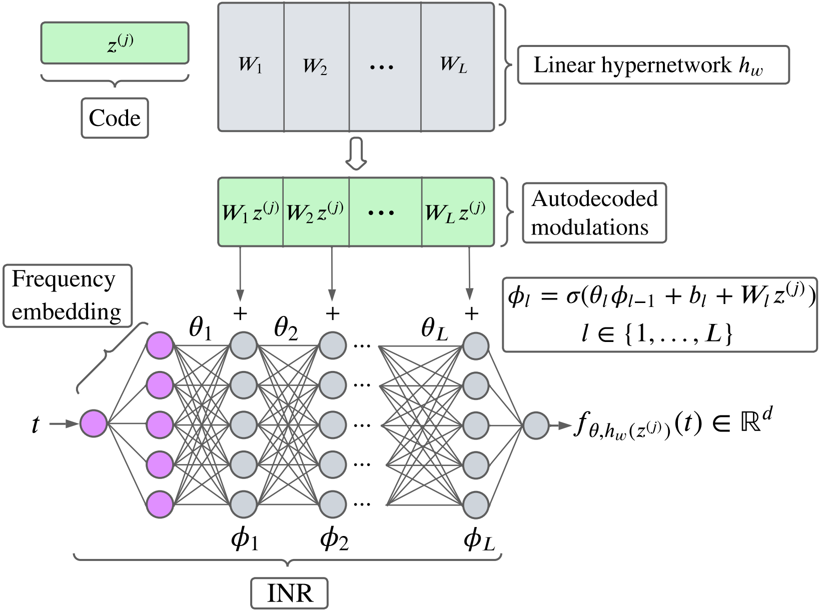
Nice to fit a *sample*, but how to deal with a *dataset*?





- Solution → **Hypernetwork that modulate the INR**
  [Dupont et al., 2022, Klocek et al., 2019, Sitzmann et al., 2020]

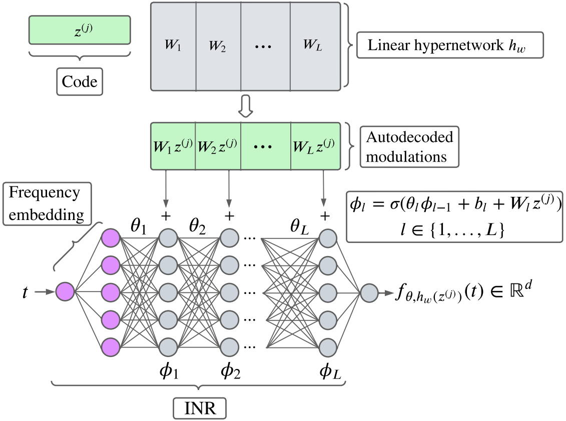
Hypernetwork and auto-decoding [Dupont et al., 2022, Yin et al., 2022]



HyperNet = linear model $\Rightarrow$ bias b

INR = fixed 1^{st} layer + MLP with θ (in) + b (modul.)

Insight on parameters θ, w and the $z^{(j)}$



- $\gamma(t)(= \phi_0) \in \mathbb{R}^{64}$, $z^{(j)} \in \mathbb{R}^{128}$
- $\phi_{\ell>0} \in \mathbb{R}^{256}$
- MLP: 5 layer

- $z^{(j)}$: instance coding
- θ and w = shared information across all samples
- MSE Loss

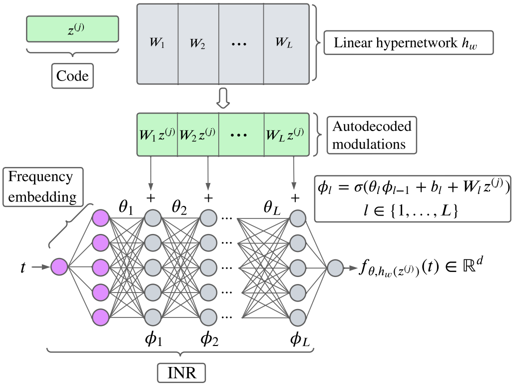
- **Training:** [Zintgraf et al., 2019] inner+outer loops

i) Sample adaptation = freeze (θ, w) + 3 grad. steps on $z^{(j)}$ [Second order grad. (Hessien comput.)]

o) (θ, w) optimization

- **Inference:** i) + forward not so fast...

Several research questions

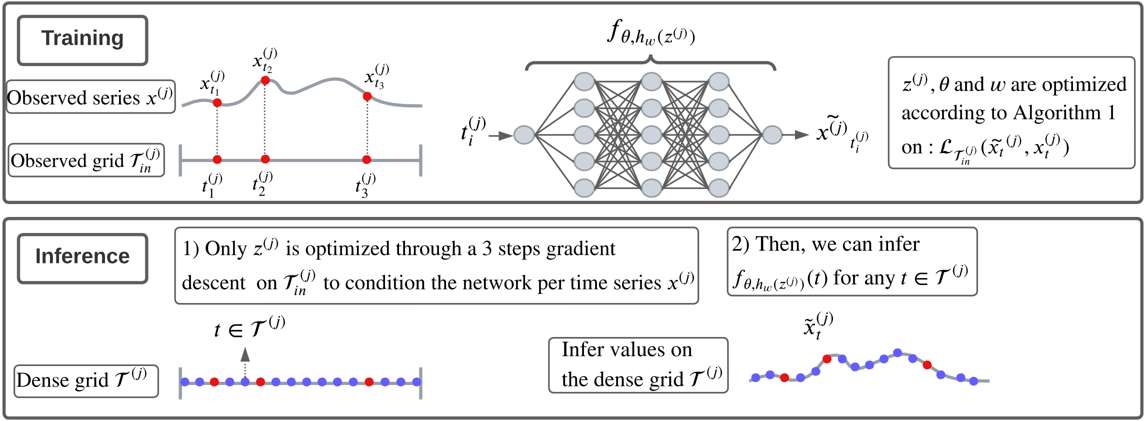


- $\gamma(t)(=\phi_0) \in \mathbb{R}^{64}$, $z^{(j)} \in \mathbb{R}^{128}$
- $\phi_{\ell>0} \in \mathbb{R}^{256}$
- MLP: 5 layer

1. How to link between hypernet. & INR?
   - *Stability*
2. Impact of grad. in the inner loop?
   - *Stability*
   - Comp. time
3. INR:
   - How to map t in input?
   - Impact of non-linearity of the INR

Experiments

Imputation


Training
Observed series $x^{(j)}$
Observed grid $\mathcal{T}_{in}^{(j)}$
$x_{t_1}^{(j)}$ $x_{t_2}^{(j)}$ $x_{t_3}^{(j)}$
$t_1^{(j)}$ $t_2^{(j)}$ $t_3^{(j)}$
$f_{\theta,h_w(z^{(j)})}$
$t_i^{(j)}$
$\tilde{x}^{(j)}_{t_i^{(j)}}$
$z^{(j)}$, θ and w are optimized according to Algorithm 1 on : $\mathcal{L}_{\mathcal{T}_{in}^{(j)}}(\tilde{x}_t^{(j)}, x_t^{(j)})$
Inference
1) Only $z^{(j)}$ is optimized through a 3 steps gradient descent on $\mathcal{T}_{in}^{(j)}$ to condition the network per time series $x^{(j)}$
2) Then, we can infer $f_{\theta,h_w(z^{(j)})}(t)$ for any $t \in \mathcal{T}^{(j)}$
$t \in \mathcal{T}^{(j)}$
Dense grid $\mathcal{T}^{(j)}$
Infer values on the dense grid $\mathcal{T}^{(j)}$
$\tilde{x}_t^{(j)}$

We compare to a wide range of baselines on three datasets

Table 1: Mean MAE imputation results on the missing grid only. τ stands for the subsampling rate. Bold results are best, underlined results are second best.

		Continuous methods				Discrete methods			
	τ	TimeFlow	DeepTime	mTAN	Neural Process	CSDI	SAITS	BRITS	TIDER
Electricity	0.05	**0.324 ± 0.013**	0.379 ± 0.037	0.575 ± 0.039	0.357 ± 0.015	0.462 ± 0.021	0.384 ± 0.019	<u>0.329 ± 0.015</u>	0.427 ± 0.010
	0.10	**0.250 ± 0.010**	0.333 ± 0.034	0.412 ± 0.047	0.417 ± 0.057	0.398 ± 0.072	0.308 ± 0.011	<u>0.287 ± 0.015</u>	0.399 ± 0.009
	0.20	**0.225 ± 0.008**	<u>0.244 ± 0.013</u>	0.342 ± 0.014	0.320 ± 0.017	0.341 ± 0.068	0.261 ± 0.008	0.245 ± 0.011	0.391 ± 0.010
	0.30	**0.212 ± 0.007**	0.240 ± 0.014	0.335 ± 0.015	0.300 ± 0.022	0.277 ± 0.059	0.236 ± 0.008	<u>0.221 ± 0.008</u>	0.384 ± 0.009
	0.50	0.194 ± 0.007	0.227 ± 0.012	0.340 ± 0.022	0.297 ± 0.016	**0.168 ± 0.003**	0.209 ± 0.008	<u>0.193 ± 0.008</u>	0.386 ± 0.009
Solar	0.05	**0.095 ± 0.015**	0.190 ± 0.020	0.241 ± 0.102	<u>0.115 ± 0.015</u>	0.374 ± 0.033	0.142 ± 0.016	0.165 ± 0.014	0.291 ± 0.009
	0.10	**0.083 ± 0.015**	0.159 ± 0.013	0.251 ± 0.081	<u>0.114 ± 0.014</u>	0.375 ± 0.038	0.124 ± 0.018	0.132 ± 0.015	0.276 ± 0.010
	0.20	**0.072 ± 0.015**	0.149 ± 0.020	0.314 ± 0.035	0.109 ± 0.016	0.217 ± 0.023	<u>0.108 ± 0.014</u>	0.109 ± 0.012	0.270 ± 0.010
	0.30	**0.061 ± 0.012**	0.135 ± 0.014	0.338 ± 0.05	0.108 ± 0.016	0.156 ± 0.002	0.100 ± 0.015	<u>0.098 ± 0.012</u>	0.266 ± 0.010
	0.50	**0.054 ± 0.013**	0.098 ± 0.013	0.315 ± 0.080	0.107 ± 0.015	<u>0.079 ± 0.011</u>	0.094 ± 0.013	0.088 ± 0.013	0.262 ± 0.009
Traffic	0.05	0.283 ± 0.016	**0.246 ± 0.010**	0.406 ± 0.074	0.318 ± 0.014	0.337 ± 0.045	0.293 ± 0.007	<u>0.261 ± 0.010</u>	0.363 ± 0.007
	0.10	**0.211 ± 0.012**	<u>0.214 ± 0.007</u>	0.319 ± 0.025	0.288 ± 0.018	0.288 ± 0.017	0.237 ± 0.006	0.245 ± 0.009	0.362 ± 0.006
	0.20	**0.168 ± 0.006**	0.216 ± 0.006	0.270 ± 0.012	0.271 ± 0.011	0.269 ± 0.017	<u>0.197 ± 0.005</u>	0.224 ± 0.008	0.361 ± 0.006
	0.30	**0.151 ± 0.007**	<u>0.172 ± 0.008</u>	0.251 ± 0.006	0.259 ± 0.012	0.240 ± 0.037	0.180 ± 0.006	0.197 ± 0.007	0.355 ± 0.006
	0.50	**0.139 ± 0.007**	0.171 ± 0.005	0.278 ± 0.040	0.240 ± 0.021	<u>0.144 ± 0.022</u>	0.160 ± 0.008	0.161 ± 0.060	0.354 ± 0.007
TimeFlow improvement		/	24.14 %	50.53 %	31.61 %	36.12 %	20.33 %	18.90 %	53.40 %

We compare to a wide range of baselines on three datasets

Table 1: Mean MAE imputation results on the missing grid only. τ stands for the subsampling rate. Bold results are best, underlined results are second best.

		Continuous methods				Discrete methods			
	τ	TimeFlow	DeepTime	mTAN	Neural Process	CSDI	SAITS	BRITS	TIDER
Electricity	0.05	**0.324 ± 0.013**	0.379 ± 0.037	0.575 ± 0.039	0.357 ± 0.015	0.462 ± 0.021	0.384 ± 0.019	0.329 ± 0.015	0.427 ± 0.010
	0.10	**0.250 ± 0.010**	0.333 ± 0.034	0.412 ± 0.047	0.417 ± 0.057	0.398 ± 0.072	0.308 ± 0.011	0.287 ± 0.015	0.399 ± 0.009
	0.20	**0.225 ± 0.008**	0.244 ± 0.013	0.342 ± 0.014	0.320 ± 0.017	0.341 ± 0.068	0.261 ± 0.008	0.245 ± 0.011	0.391 ± 0.010
	0.30	**0.212 ± 0.007**	0.240 ± 0.014	0.335 ± 0.015	0.300 ± 0.022	0.277 ± 0.059	0.236 ± 0.008	0.221 ± 0.008	0.384 ± 0.009
	0.50	0.194 ± 0.007	0.227 ± 0.012	0.340 ± 0.022	0.297 ± 0.016	**0.168 ± 0.003**	0.209 ± 0.008	0.193 ± 0.008	0.386 ± 0.009
Solar	0.05	**0.095 ± 0.015**	0.190 ± 0.020	0.241 ± 0.102	0.115 ± 0.015	0.374 ± 0.033	0.142 ± 0.016	0.165 ± 0.014	0.291 ± 0.009
	0.10	**0.083 ± 0.015**	0.159 ± 0.013	0.251 ± 0.081	0.114 ± 0.014	0.375 ± 0.038	0.124 ± 0.018	0.132 ± 0.015	0.276 ± 0.010
	0.20	**0.072 ± 0.015**	0.149 ± 0.020	0.314 ± 0.035	0.109 ± 0.016	0.217 ± 0.023	0.108 ± 0.014	0.109 ± 0.012	0.270 ± 0.010
	0.30	**0.061 ± 0.012**	0.135 ± 0.014	0.338 ± 0.05	0.108 ± 0.016	0.156 ± 0.002	0.100 ± 0.015	0.098 ± 0.012	0.266 ± 0.010
	0.50	**0.054 ± 0.013**	0.098 ± 0.013	0.315 ± 0.080	0.107 ± 0.015	0.079 ± 0.011	0.094 ± 0.013	0.088 ± 0.013	0.262 ± 0.009
Traffic	0.05	0.283 ± 0.016	**0.246 ± 0.010**	0.406 ± 0.074	0.318 ± 0.014	0.337 ± 0.045	0.293 ± 0.007	0.261 ± 0.010	0.363 ± 0.007
	0.10	**0.211 ± 0.012**	0.214 ± 0.007	0.319 ± 0.025	0.288 ± 0.018	0.288 ± 0.017	0.237 ± 0.006	0.245 ± 0.009	0.362 ± 0.006
	0.20	**0.168 ± 0.006**	0.216 ± 0.006	0.270 ± 0.012	0.271 ± 0.011	0.269 ± 0.017	0.197 ± 0.005	0.224 ± 0.008	0.361 ± 0.006
	0.30	**0.151 ± 0.007**	0.172 ± 0.008	0.251 ± 0.006	0.259 ± 0.012	0.240 ± 0.037	0.180 ± 0.006	0.197 ± 0.007	0.355 ± 0.006
	0.50	**0.139 ± 0.007**	0.171 ± 0.005	0.278 ± 0.040	0.240 ± 0.021	0.144 ± 0.022	0.160 ± 0.008	0.161 ± 0.060	0.354 ± 0.007
TimeFlow improvement		/	24.14 %	50.53 %	31.61 %	36.12 %	20.33 %	18.90 %	53.40 %

	TimeFlow	DeepTime	NeuralProcess	mTAN	SAITS	BRITS	TIDER
Number of parameters	602k	1315k	248k	113k	11 137k	6 220k	1 034k

Figure 1: Number of parameters for each DL methods on the imputation task on the Electricity dataset.

Qualitative comparison with BRITS

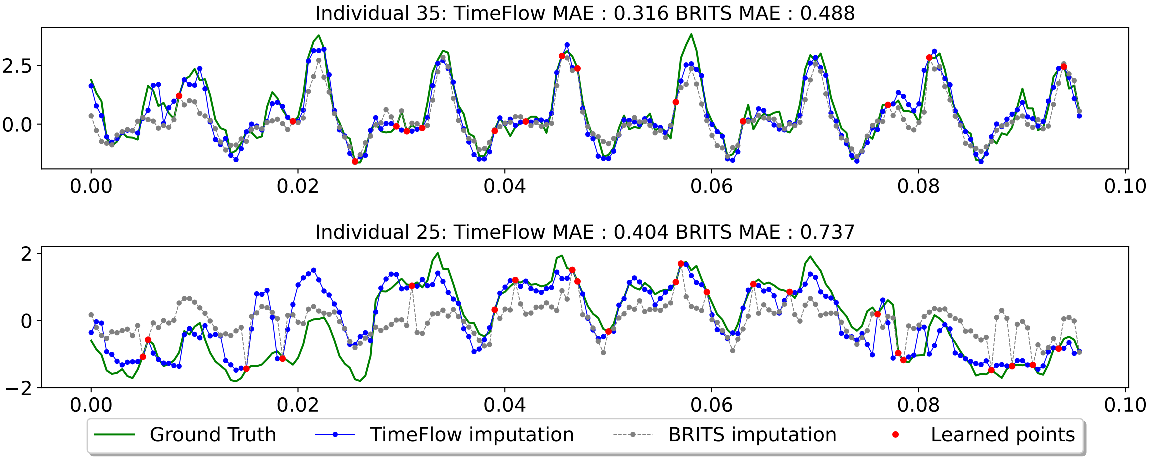
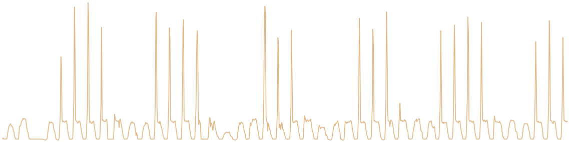


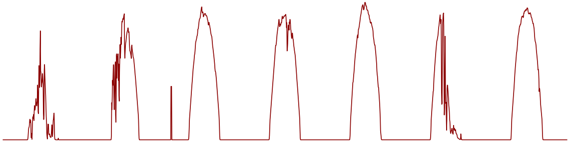
Figure 2: *Electricity dataset*. TimeFlow imputation (blue line) and BRITS imputation (gray line) with 10% of known point (red points) on the eight first days of samples 35 (top) and 25 (bottom).

A step back: what do we learn on which signals?

- Are the series too regular in our experiments?
- Is it reasonable to predict from so few training points?
- Should we consider time windows instead of time measurements?



Road occupancy

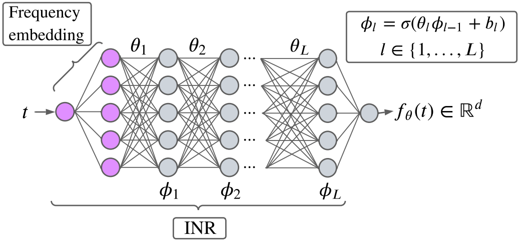


Solar power generation

Ablation: What makes architecture work?

Signal encoding:

Fourier features vs SIREN [Sitzmann et al., 2020]

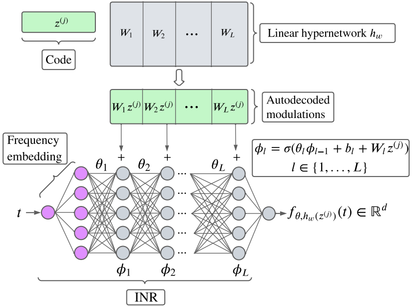


- $\gamma(t)(= \phi_0) \in \mathbb{R}^{64}$, $z^{(j)} \in \mathbb{R}^{128}$
- $\phi_{\ell>0} \in \mathbb{R}^{256}$
- MLP: 5 layer

	τ	TimeFlow	TimeFlow w SIREN
Electricity	0.05	**0.323**	0.466
	0.10	**0.252**	0.350
	0.20	**0.224**	0.242
	0.30	**0.211**	0.222
	0.50	**0.194**	0.209
Solar	0.05	**0.105**	0.114
	0.10	**0.083**	0.094
	0.20	**0.065**	0.079
	0.30	**0.061**	0.072
	0.50	**0.056**	0.066
Traffic	0.05	**0.292**	0.333
	0.10	**0.220**	0.252
	0.20	**0.168**	0.191
	0.30	**0.152**	0.163
	0.50	**0.141**	0.154

Ablation: What makes architecture work?

Latent dimensions

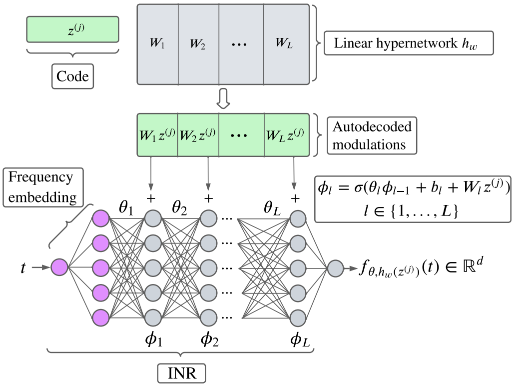


- $\gamma(t)(= \phi_0) \in \mathbb{R}^{64}$, $z^{(j)} \in \mathbb{R}^{128}$
- $\phi_{\ell > 0} \in \mathbb{R}^{256}$
- MLP: 5 layer

	H	32	64	128	256
Electricity	96	0.232 ± 0.016	0.222 ± 0.017	0.222 ± 0.018	**0.215 ± 0.019**
	192	0.245 ± 0.020	0.239 ± 0.018	**0.230 ± 0.026**	0.233 ± 0.017
	336	0.254 ± 0.029	0.244 ± 0.028	0.262 ± 0.031	**0.243 ± 0.032**
	720	0.295 ± 0.027	0.284 ± 0.028	0.303 ± 0.041	**0.283 ± 0.029**
SolarH	96	0.182 ± 0.009	0.181 ± 0.012	**0.179 ± 0.003**	0.225 ± 0.047
	192	0.195 ± 0.014	0.195 ± 0.016	**0.193 ± 0.015**	0.197 ± 0.029
	336	**0.181 ± 0.011**	0.182 ± 0.011	0.189 ± 0.013	0.183 ± 0.012
	720	0.201 ± 0.027	**0.199 ± 0.025**	0.209 ± 0.029	0.200 ± 0.030
Traffic	96	0.223 ± 0.024	0.215 ± 0.028	0.215 ± 0.037	**0.210 ± 0.033**
	192	0.214 ± 0.018	0.217 ± 0.025	0.206 ± 0.023	**0.203 ± 0.024**
	336	0.238 ± 0.029	0.231 ± 0.029	**0.226 ± 0.030**	0.229 ± 0.029
	720	0.272 ± 0.040	0.269 ± 0.035	**0.259 ± 0.038**	0.262 ± 0.040

Ablation: What makes architecture work?

How to retrieve the latent code?

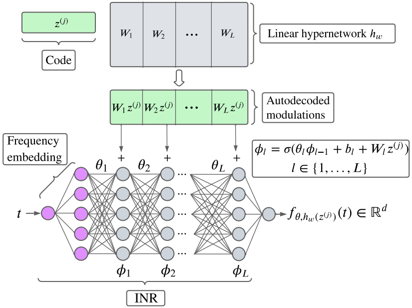


	τ	TimeFlow	TimeFlow w REPTILE
Electricity	0.05	**0.324 ± 0.013**	0.363 ± 0.062
	0.10	**0.250 ± 0.010**	0.343 ± 0.036
	0.20	**0.225 ± 0.008**	0.312 ± 0.043
	0.30	**0.212 ± 0.007**	0.308 ± 0.035
	0.50	**0.194 ± 0.007**	0.305 ± 0.046
Solar	0.05	**0.095 ± 0.015**	0.125 ± 0.025
	0.10	**0.083 ± 0.015**	0.123 ± 0.032
	0.20	**0.072 ± 0.015**	0.108 ± 0.021
	0.30	**0.061 ± 0.012**	0.105 ± 0.027
	0.50	**0.054 ± 0.013**	0.102 ± 0.021
Traffic	0.05	**0.283 ± 0.016**	0.304 ± 0.026
	0.10	**0.211 ± 0.012**	0.264 ± 0.009
	0.20	**0.168 ± 0.006**	0.242 ± 0.019
	0.30	**0.151 ± 0.007**	0.218 ± 0.020
	0.50	**0.139 ± 0.007**	0.216 ± 0.017

Gradient algo.

Ablation: What makes architecture work?

How to retrieve the latent code?

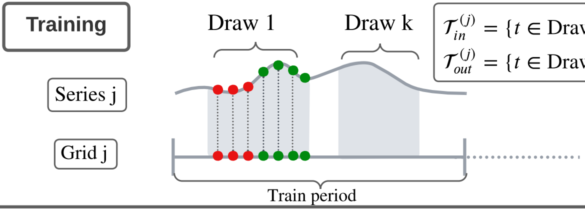


	H	1	3	10	50
Electricity	96	0.259 ± 0.020	**0.222 ± 0.018**	**0.222 ± 0.017**	0.228 ± 0.019
	192	0.269 ± 0.020	**0.230 ± 0.026**	0.232 ± 0.020	0.233 ± 0.026
	336	0.273 ± 0.033	**0.262 ± 0.031**	0.264 ± 0.032	0.268 ± 0.032
	720	0.351 ± 0.038	0.303 ± 0.041	0.300 ± 0.040	**0.299 ± 0.039**
SolarH	96	0.487 ± 0.196	**0.179 ± 0.003**	0.181 ± 0.003	0.186 ± 0.003
	192	0.411 ± 0.088	**0.193 ± 0.015**	0.195 ± 0.014	0.199 ± 0.013
	336	0.435 ± 0.153	**0.189 ± 0.013**	0.203 ± 0.006	0.223 ± 0.012
	720	0.394 ± 0.173	0.209 ± 0.029	**0.203 ± 0.006**	0.209 ± 0.027
Traffic	96	0.320 ± 0.038	**0.215 ± 0.037**	0.219 ± 0.043	0.226 ± 0.046
	192	0.299 ± 0.023	**0.206 ± 0.023**	0.209 ± 0.026	0.214 ± 0.027
	336	0.345 ± 0.038	**0.226 ± 0.030**	0.228 ± 0.031	0.233 ± 0.032
	720	0.321 ± 0.034	**0.259 ± 0.038**	0.260 ± 0.038	0.266 ± 0.039

Nb gradient steps

Forecasting

Training



$\mathcal{T}_{in}^{(j)} = \{t \in \text{Draw} : t = \bullet\}$

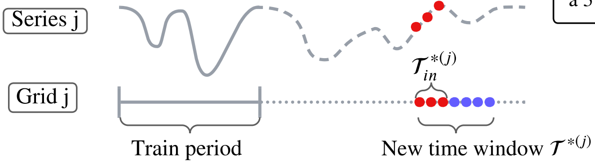
$\mathcal{T}_{out}^{(j)} = \{t \in \text{Draw} : t = \bullet\}$

$\forall t^{(j)} \in \mathcal{T}_{in}^{(j)} \cup \mathcal{T}_{out}^{(j)},\ \tilde{x}_{t^{(j)}}^{(j)} = f_{\theta,h_w(z^{(j)})}(t^{(j)})$

$z^{(j)}$ is uniquely optimized on $\mathcal{T}_{in}^{(j)}$

θ and w are optimized on $\mathcal{T}_{in}^{(j)}$ and $\mathcal{T}_{out}^{(j)}$

Inference



1) Only $z^{*(j)}$ is optimized through a 3 steps gradient descent on $\mathcal{T}_{in}^{*(j)}$

2) Then, we can infer $f_{\theta,h_w(z^{*(j)})}(t)$ for any $t \in \mathcal{T}^{*(j)}$

Infer values on $\mathcal{T}^{*(j)}$

Wide range of baselines on three datasets

Table 2: Mean MAE forecast results for adjacent time windows. H stands for the horizon. Bold results are best, underline results are second best. Look-back window size = 512

		Continuous methods			Discrete methods			
	H	TimeFlow	DeepTime	Neural Process	Patch-TST	DLinear	AutoFormer	Informer
Electricity	96	<u>0.218 ± 0.017</u>	0.240 ± 0.027	0.392 ± 0.045	**0.214** ± **0.020**	0.236 ± 0.035	0.310 ± 0.031	0.293 ± 0.0184
	192	<u>0.238 ± 0.012</u>	0.251 ± 0.023	0.401 ± 0.046	**0.225** ± **0.017**	0.248 ± 0.032	0.322 ± 0.046	0.336 ± 0.032
	336	<u>0.265 ± 0.036</u>	0.290 ± 0.034	0.434 ± 0.075	**0.242** ± **0.024**	0.284 ± 0.043	0.330 ± 0.019	0.405 ± 0.044
	720	<u>0.318 ± 0.073</u>	0.356 ± 0.060	0.605 ± 0.149	**0.291** ± **0.040**	0.370 ± 0.086	0.456 ± 0.052	0.489 ± 0.072
SolarH	96	**0.172** ± **0.017**	<u>0.197 ± 0.002</u>	0.221 ± 0.048	0.232 ± 0.008	0.204 ± 0.002	0.261 ± 0.053	0.273 ± 0.023
	192	**0.198** ± **0.010**	<u>0.202 ± 0.014</u>	0.244 ± 0.048	0.231 ± 0.027	0.211 ± 0.012	0.312 ± 0.085	0.256 ± 0.026
	336	<u>0.207 ± 0.019</u>	**0.200** ± **0.012**	0.241 ± 0.005	0.254 ± 0.048	0.212 ± 0.019	0.341 ± 0.107	0.287 ± 0.006
	720	**0.215** ± **0.016**	<u>0.240 ± 0.011</u>	0.403 ± 0.147	0.271 ± 0.036	0.246 ± 0.015	0.368 ± 0.006	0.341 ± 0.049
Traffic	96	<u>0.216 ± 0.033</u>	0.229 ± 0.032	0.283 ± 0.028	**0.201** ± **0.031**	0.225 ± 0.034	0.299 ± 0.080	0.324 ± 0.113
	192	<u>0.208 ± 0.021</u>	0.220 ± 0.020	0.292 ± 0.023	**0.195** ± **0.024**	0.215 ± 0.022	0.320 ± 0.036	0.321 ± 0.052
	336	<u>0.237 ± 0.040</u>	0.247 ± 0.033	0.305 ± 0.039	**0.220** ± **0.036**	0.244 ± 0.035	0.450 ± 0.127	0.394 ± 0.066
	720	**0.266** ± **0.048**	0.290 ± 0.045	0.339 ± 0.037	<u>0.268 ± 0.050</u>	0.290 ± 0.047	0.630 ± 0.043	0.441 ± 0.055
TimeFlow improvement		/	6.56 %	30.79 %	2.64 %	7.30 %	35.43 %	33.07 %

Forecast on sparsely observed look-back window (1/2)

Table 3: MAE results for forecasting with missing values in the look-back window. τ stands for the percentage of observed values in the look-back window. Best results are in bold. Look-back window size = 512

			TimeFlow		DeepTime		Neural Process	
	H	τ	Imputation error	Forecast error	Imputation error	Forecast error	Imputation error	Forecast error
Electricity	96	0.5	**0.151** ± **0.003**	**0.239** ± **0.013**	0.209 ± 0.004	0.270 ± 0.019	0.460 ± 0.048	0.486 ± 0.078
		0.2	**0.208** ± **0.006**	**0.260** ± **0.015**	0.249 ± 0.006	0.296 ± 0.023	0.644 ± 0.079	0.650 ± 0.095
		0.1	**0.272** ± **0.006**	**0.295** ± **0.016**	0.284 ± 0.007	0.324 ± 0.026	0.740 ± 0.083	0.737 ± 0.106
	192	0.5	**0.149** ± **0.004**	**0.235** ± **0.011**	0.204 ± 0.004	0.265 ± 0.018	0.461 ± 0.045	0.498 ± 0.070
		0.2	**0.209** ± **0.006**	**0.257** ± **0.013**	0.244 ± 0.007	0.290 ± 0.023	0.601 ± 0.075	0.626 ± 0.101
		0.1	**0.274** ± **0.010**	**0.289** ± **0.016**	0.282 ± 0.007	0.315 ± 0.025	0.461 ± 0.045	0.724 ± 0.090
Traffic	96	0.5	**0.180** ± **0.016**	**0.219** ± **0.026**	0.272 ± 0.028	0.243 ± 0.030	0.436 ± 0.025	0.444 ± 0.047
		0.2	**0.239** ± **0.019**	**0.243** ± **0.027**	0.335 ± 0.026	0.293 ± 0.027	0.596 ± 0.049	0.597 ± 0.075
		0.1	**0.312** ± **0.020**	**0.290** ± **0.027**	0.385 ± 0.025	0.344 ± 0.027	0.734 ± 0.102	0.731 ± 0.132
	192	0.5	**0.176** ± **0.014**	**0.217** ± **0.017**	0.241 ± 0.027	0.234 ± 0.021	0.477 ± 0.042	0.476 ± 0.043
		0.2	**0.233** ± **0.017**	**0.236** ± **0.021**	0.286 ± 0.027	0.276 ± 0.020	0.685 ± 0.109	0.678 ± 0.108
		0.1	**0.304** ± **0.019**	**0.277** ± **0.021**	0.331 ± 0.025	0.324 ± 0.021	0.888 ± 0.178	0.877 ± 0.174
TimeFlow improvement			/	/	18.97 %	11.87 %	61.88 %	58.41 %

Forecast on sparsely observed look-back window (2/2)

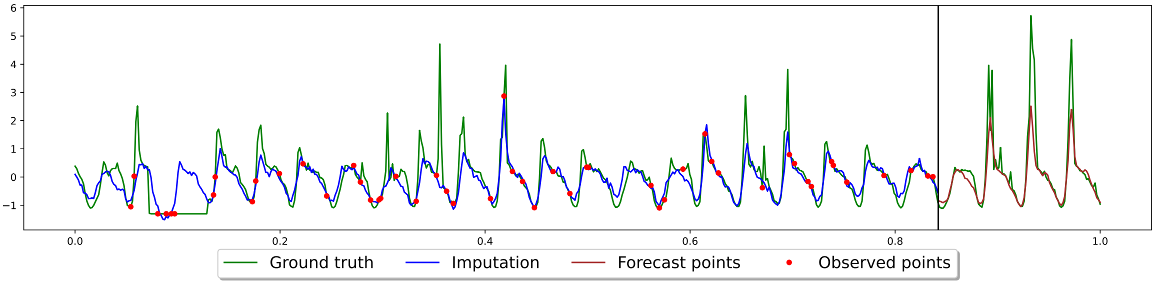
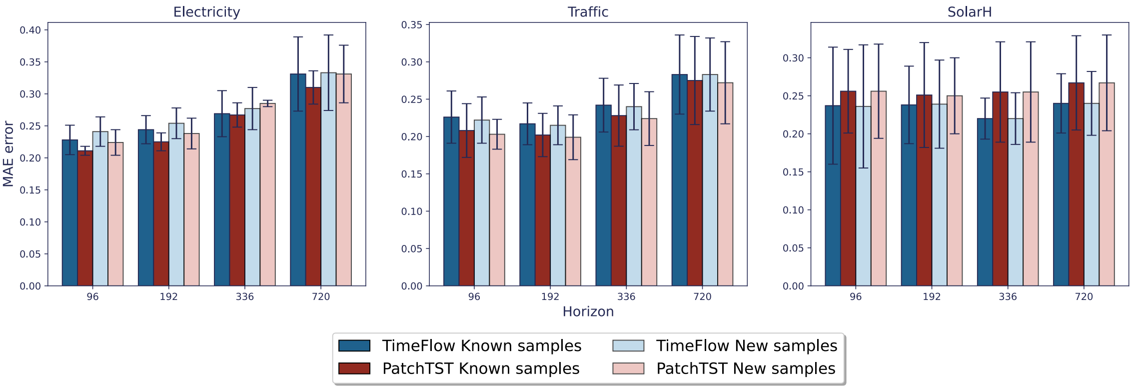


Figure 3: *Traffic dataset, sample 95*. In this figure, TimeFlow simultaneously imputes and forecasts at horizon 96 with a 10% partially observed look-back window of length 512.

Known *vs* New Samples

- TimeFlow *vs* PatchTST

$\Rightarrow$ Very close performances: Known $\approx$ New / TimeFlow $\approx$ PatchTST



Quantify uncertainty with TimeFlow ($\mathcal{L}$ is the pinball loss)

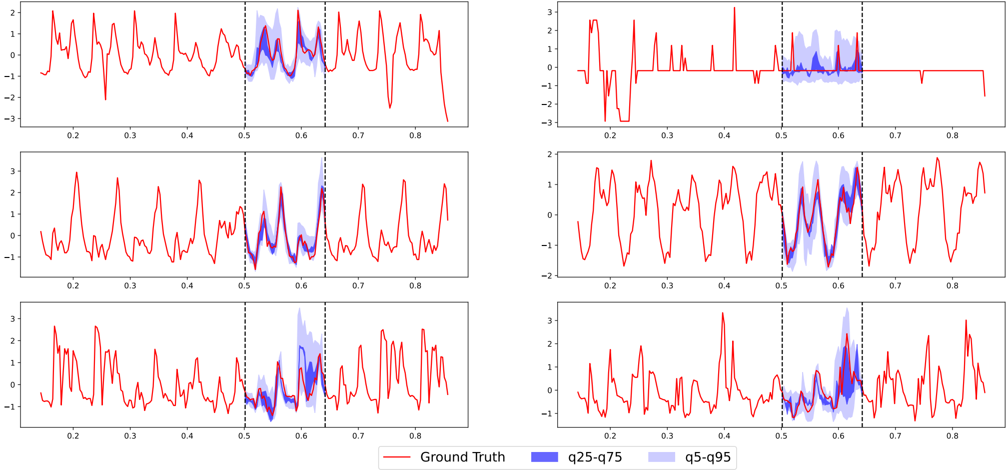


Figure 4: Quantifying uncertainty in block imputation of two missing days in the *Electricity* dataset.

An interesting latent space

Latent space exploration

- For a given time series family $\{x^{(j)}\}_{j=1}^{n}$ we learn a family of codes $\{z^{(j)}\}_{j=1}^{n}$ in the latent space.

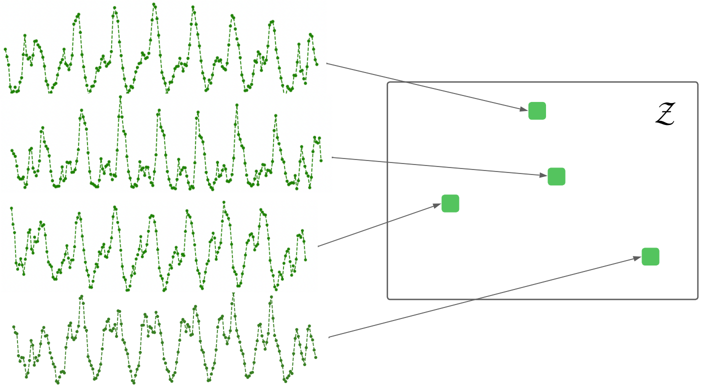


Figure 5: Latent space visualization

Bezier path between two codes z

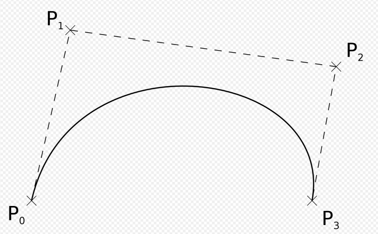


Figure 6: Bezier path between two points

$$B(\lambda) = (1-\lambda)^3\mathbf{P}_0 + 3(1-\lambda)^2\lambda\mathbf{P}_1 + 3(1-\lambda)\lambda^2\mathbf{P}_2 + \lambda^3\mathbf{P}_3$$

Generate new time series through latent space interpolation

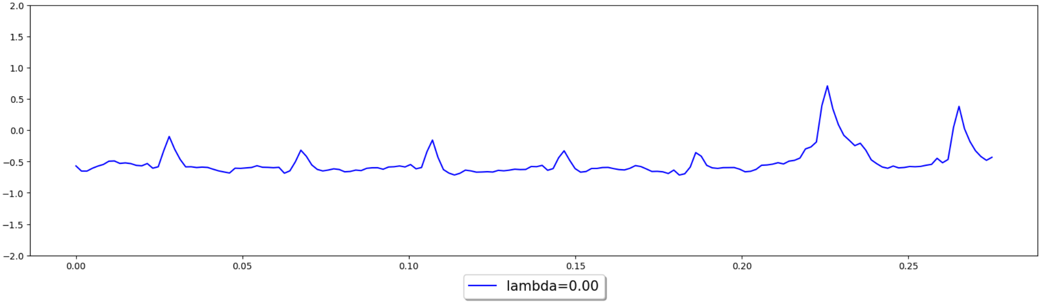
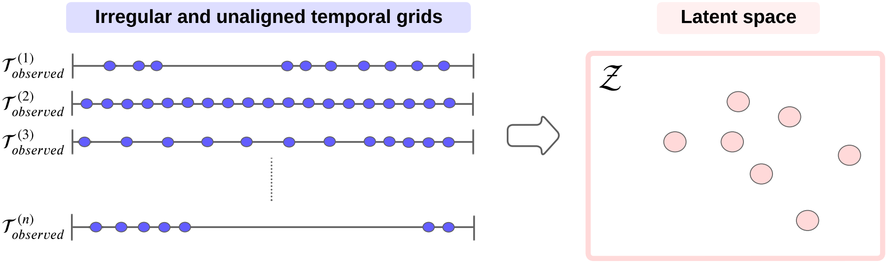


Figure 7: Autodecoded generate z_λ for several λ's

Learn representations from irregular/unaligned time series

Extract aligned representations



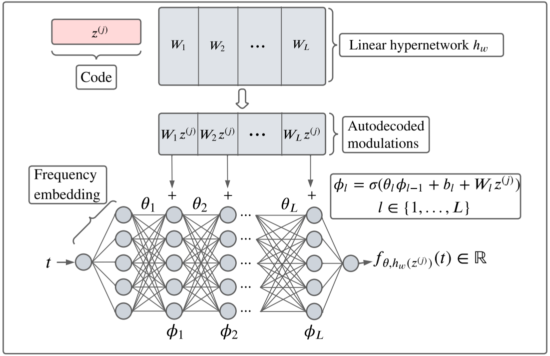
Motivations

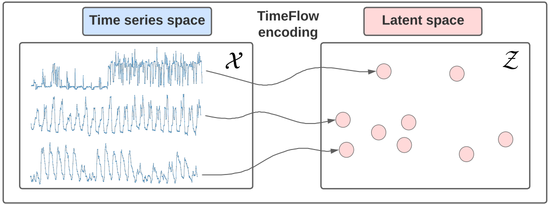
- A well-aligned latent space makes it easier to perform downstream tasks
- This two-stage approach is underexplored

TimeFlow can capture flexible representations

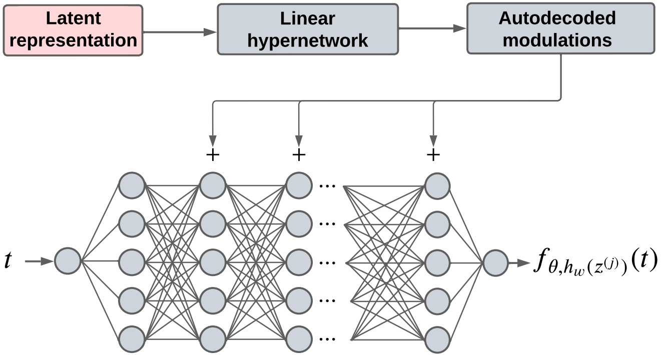
- Auto-decoding mechanism extracts a representation $z^{(j)} \in \mathbb{R}^d$ for a time series $x^{(j)} \in \mathbb{R}^{T^{(j)}}$
- For a time series dataset $\{\mathbf{x}^{(j)}\}_{j=1}^{n}$ we can build the corresponding representations dataset $\{\mathbf{z}^{(j)}\}_{j=1}^{n}$

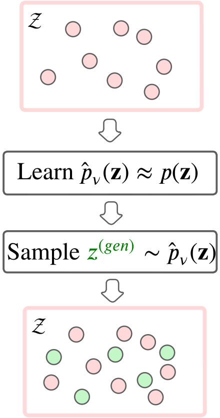
- Is the latent space $\mathcal{Z}$ efficient for downstream tasks ?





Example of downstream task : unconditional generation





- **Motivations**: data augmentation, overcoming privacy/property constraints
- **Training procedure**: (i) Fit TimeFlow (ii) Learn a Denoising Diffusion Probabilistic Model (DDPM) on the learned representations
- **Inference procedure**: (i) Sample a new representation (ii) Decode the representation

Experiments

Experimental setup

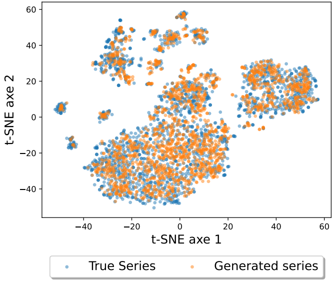
- Training on 8000 hourly time series (two-weeks long) from *Electricity*
- 2000 time series for testing and 2000 generated time series
- We compare with two baselines : DDPM only and TimeGan [Yoon et al., 2019]
- We want to assess the fidelity and diversity

	TimeFlow + DDPM	**DDPM only**	**TimeGAN**	**Fully separable generation**
Discriminative score ↓	**0.1388**	0.1704	0.4890	0.5000

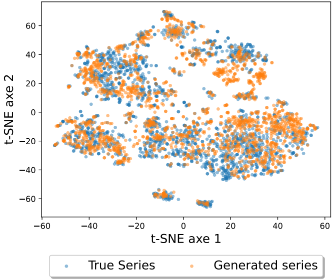
Experiments

Experimental setup

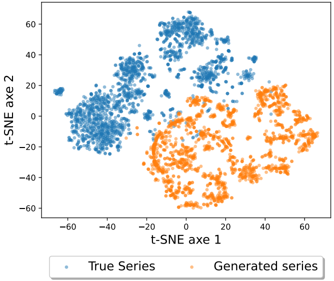
- Training on 8000 hourly time series (two-weeks long) from *Electricity*
- 2000 time series for testing and 2000 generated time series
- We compare with two baselines : DDPM only and TimeGan [Yoon et al., 2019]
- We want to assess the fidelity and diversity



TimeFlow + DDPM



DDPM only



TimeGan.

Conclusion

Synthesis

1. A semantically rich latent space
2. The representations can encode irregular/unaligned time series
3. First unconditional generation experiments are convincing

Limitations and perspectives

1. Unconditional generation experiments performed on only one dataset
2. Other downstream tasks should be explored to assess usefulness for downstream tasks

Conclusion

Key takeaways

TimeFlow offers:

- Unified + Continuous approach for time series **imputation & forecasting**.
- Adaptability to new contexts through meta-learning optimization.
- High performances in all situations (same hyper-parameters)
- Wide range of experiments to measure the benefits of all components

Limitation:

- Inference computation time (10-100 slower that competitors)

Perspectives:

- Moving to multivariate Time-Series

Perspectives

- Time Series... + context modeling
  - Public transportation
  - Agronomy : Growth model / yield prediction
- Time representation $\Rightarrow$ Move to a time window
  - How transformers failed to represent local data
  - Memory networks
  - VQ-VAE / time series

References I

E. Dupont, A. Golinski, M. Alizadeh, Y. W. Teh, and A. Doucet. Coin: Compression with implicit neural representations. In Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021, 2021.

E. Dupont, H. Kim, S. A. Eslami, D. J. Rezende, and D. Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In International Conference on Machine Learning, pages 5694–5725. PMLR, 2022.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 33:6840–6851, 2020.

C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser, et al. Local implicit grid representations for 3d scenes. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 6001–6010, 2020.

References II

H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh. Attentive neural processes. arXiv preprint arXiv:1901.05761, 2019.

S. Klocek, Ł. Maziarka, M. Wołczyk, J. Tabor, J. Nowak, and M. Śmieja. Hypernetwork functional image representation. In Artificial Neural Networks and Machine Learning–ICANN 2019: Workshop and Special Sessions: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings 28, pages 496–510. Springer, 2019.

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1):99–106, 2021.

References III

S. Peng, Y. Zhang, Y. Xu, Q. Wang, Q. Shuai, H. Bao, and X. Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 9054–9063, 2021.

S. N. Shukla and B. M. Marlin. Multi-time attention networks for irregularly sampled time series. arXiv preprint arXiv:2101.10318, 2021.

V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. Advances in Neural Information Processing Systems, 33:7462–7473, 2020.

C. K. Williams and C. E. Rasmussen. Gaussian processes for machine learning, volume 2. MIT press Cambridge, MA, 2006.

References IV

G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. C. H. Hoi. Deeptime: Deep time-index meta-learning for non-stationary time-series forecasting. CoRR, abs/2207.06046, 2022.

Y. Yin, M. Kirchmeyer, J.-Y. Franceschi, A. Rakotomamonjy, and P. Gallinari. Continuous pde dynamics forecasting with implicit neural representations. arXiv preprint arXiv:2209.14855, 2022.

J. Yoon, D. Jarrett, and M. Van der Schaar. Time-series generative adversarial networks. Advances in neural information processing systems, 32, 2019.

L. Zintgraf, K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson. Fast context adaptation via meta-learning. In International Conference on Machine Learning, pages 7693–7702. PMLR, 2019.