

Sim-to-Real Transfer with Asymmetric Temporal Supervision

Using Alignment Techniques on an Implicit Neural Representation (INR) Based Architecture

Julian Agudelo

Supervised by Cristina Manfredotti, Vincent Guigue, Evelyne Lutton and Hadrien Piot

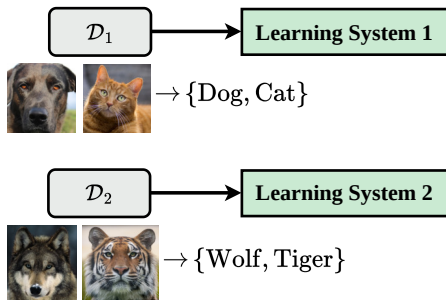


1. A Reminder on Transfer Learning
2. Task Formulation
3. Modulated Implicit Neural Representations
4. Lessons From Unsupervised Domain Alignment
5. Domain Alignment on an INR-FiLM architecture
6. Use Case
7. Results

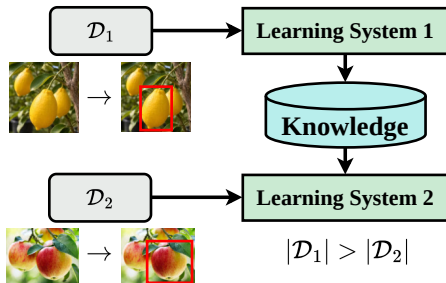
A Reminder on Transfer Learning



Traditional Learning



Transfer Learning

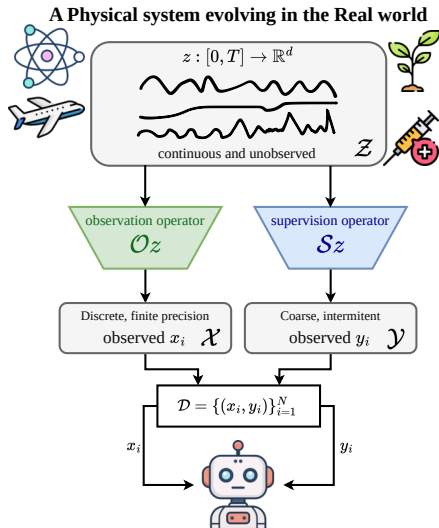


Pan and Yang 2010

Task Formulation

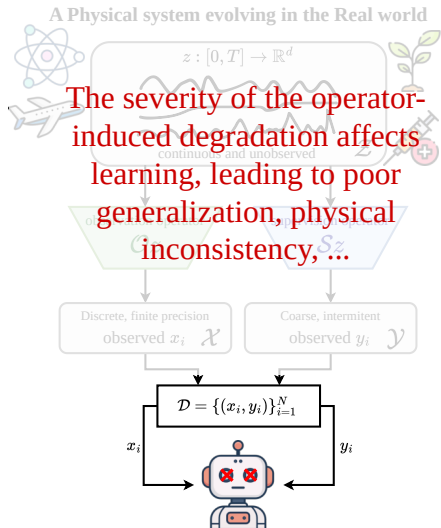


Learning From Imperfect Temporal Observations



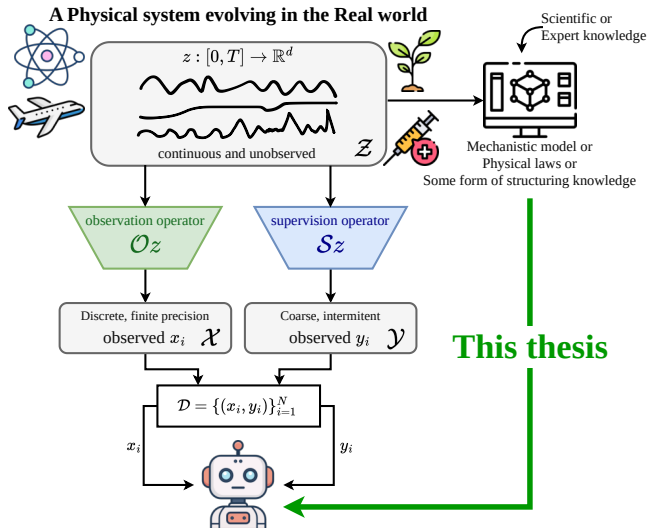


Learning From Imperfect Temporal Observations





Learning From Imperfect Temporal Observations





- We denote $\mathbf{x} \in \mathcal{X} = \{\mathbf{x}^{(d)} \cup \mathbf{x}^{(s)}\}$, the covariates.
- We denote $\{y_\tau\}_{\tau=0}^T \in \mathcal{Y}$, the label.
- Our objective is to find $f_{\theta^*} : f_{\theta^*}(\mathbf{x}) = \{\hat{y}_\tau\}_{\tau=0}^T \approx \{y_\tau\}_{\tau=0}^T$ in the real domain $\mathcal{R}$.



Transfer Learning for Time Series with Asymmetric Supervision

- We denote $\mathbf{x} \in \mathcal{X} = \{\mathbf{x}^{(d)} \cup \mathbf{x}^{(s)}\}$, the covariates.
- We denote $\{y_\tau\}_{\tau=0}^T \in \mathcal{Y}$, the label.
- Our objective is to find $f_{\theta^*} : f_{\theta^*}(\mathbf{x}) = \{\hat{y}_\tau\}_{\tau=0}^T \approx \{y_\tau\}_{\tau=0}^T$ in the real domain $\mathcal{R}$.

Supervision in the real domain $\mathcal{R}$ is strongly partial



Transfer Learning for Time Series with Asymmetric Supervision

- We denote $\mathbf{x} \in \mathcal{X} = \{\mathbf{x}^{(d)} \cup \mathbf{x}^{(s)}\}$, the covariates.
- We denote $\{y_\tau\}_{\tau=0}^T \in \mathcal{Y}$, the label.
- Our objective is to find $f_{\theta^*} : f_{\theta^*}(\mathbf{x}) = \{\hat{y}_\tau\}_{\tau=0}^T \approx \{y_\tau\}_{\tau=0}^T$ in the real domain $\mathcal{R}$.

Supervision in the real domain $\mathcal{R}$ is strongly partial

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{P_{\mathcal{R}}} \left[\sum_{t=0}^T \ell(\hat{y}_t, y_t) \right]$$

$$\mathcal{L}_{\mathcal{R}}(\theta) = \frac{1}{N_{\mathcal{R}}} \sum_{j=1}^{N_{\mathcal{R}}} [\ell(\hat{y}_0^{(j)}, y_0^{(j)}) + \ell(\hat{y}_T^{(j)}, y_T^{(j)})]$$



Transfer Learning for Time Series with Asymmetric Supervision

- We denote $\mathbf{x} \in \mathcal{X} = \{\mathbf{x}^{(d)} \cup \mathbf{x}^{(s)}\}$, the covariates.
- We denote $\{y_\tau\}_{\tau=0}^T \in \mathcal{Y}$, the label.
- Our objective is to find $f_{\theta^*} : f_{\theta^*}(\mathbf{x}) = \{\hat{y}_\tau\}_{\tau=0}^T \approx \{y_\tau\}_{\tau=0}^T$ in the real domain $\mathcal{R}$.

Supervision in the real domain $\mathcal{R}$ is strongly partial

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{P_{\mathcal{R}}} \left[\sum_{t=0}^T \ell(\hat{y}_t, y_t) \right]$$

$$\mathcal{L}_{\mathcal{R}}(\theta) = \frac{1}{N_{\mathcal{R}}} \sum_{j=1}^{N_{\mathcal{R}}} [\ell(\hat{y}_0^{(j)}, y_0^{(j)}) + \ell(\hat{y}_T^{(j)}, y_T^{(j)})]$$

But we know that the system generating the data has a mechanistic model representation, giving us access to a simulated domain $\mathcal{S}$. **Our goal is to learn a surrogate on that domain and use it for regularizing learning in the target $\mathcal{R}$ domain.**



Transfer Learning for Time Series with Asymmetric Supervision

- We denote $\mathbf{x}' \in \mathcal{X}' = \{\mathbf{x}'^{(d)} \cup \mathbf{x}'^{(s)}\}$, the covariates.
- We denote $\{y'_\tau\}_{\tau=0}^T \in \mathcal{Y}'$, the label.
- Our objective is to find $g_{\phi^*} : g_{\phi^*}(\mathbf{x}') = \{\hat{y}'_\tau\}_{\tau=0}^T \approx \{y'_\tau\}_{\tau=0}^T$ in the simulated domain $\mathcal{S}$.

$$\phi^* = \arg \min_{\phi} \mathbb{E}_{P_S} \left[\sum_{t=0}^T \ell(\hat{y}'_t, y'_t) \right]$$

$$\mathcal{L}_S(\phi) = \frac{1}{N_S} \sum_{i=1}^{N_S} \sum_{\tau=0}^T \ell(\hat{y}'_\tau^{(i)}, y_\tau'^{(i)})$$



Transfer Learning for Time Series with Asymmetric Supervision

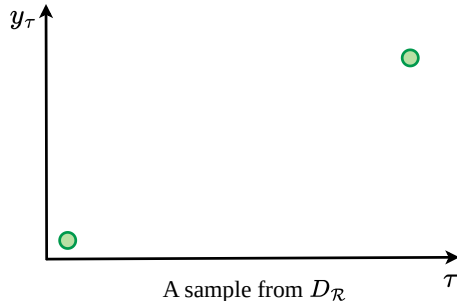
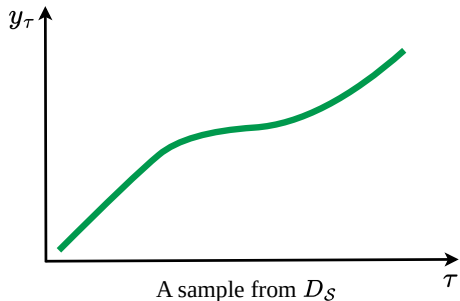
- We denote $\mathbf{x}' \in \mathcal{X}' = \{\mathbf{x}'^{(d)} \cup \mathbf{x}'^{(s)}\}$, the covariates.
- We denote $\{y'_\tau\}_{\tau=0}^T \in \mathcal{Y}'$, the label.
- Our objective is to find $g_{\phi^*} : g_{\phi^*}(\mathbf{x}') = \{\hat{y}'_\tau\}_{\tau=0}^T \approx \{y'_\tau\}_{\tau=0}^T$ in the simulated domain $\mathcal{S}$.

$$\phi^* = \arg \min_{\phi} \mathbb{E}_{P_S} \left[\sum_{t=0}^T \ell(\hat{y}'_t, y'_t) \right]$$

$$\mathcal{L}_S(\phi) = \frac{1}{N_S} \sum_{i=1}^{N_S} \sum_{\tau=0}^T \ell(\hat{y}'_\tau^{(i)}, y'_\tau^{(i)})$$

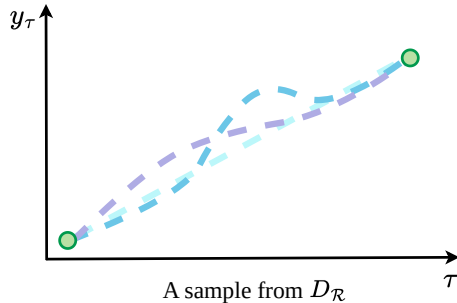
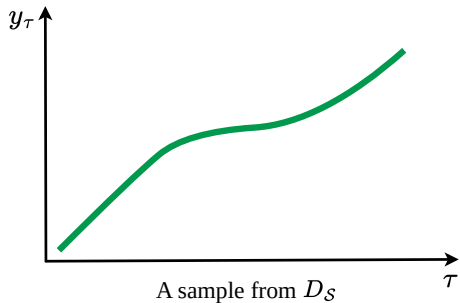
We hope that learning to solving this task will provides us some knowledge that can be *transferred* and used for solving the ill-posed task in the real domain $\mathcal{R}$.

Transfer Learning for Time Series with Asymmetric Supervision



● Available y_τ for supervision

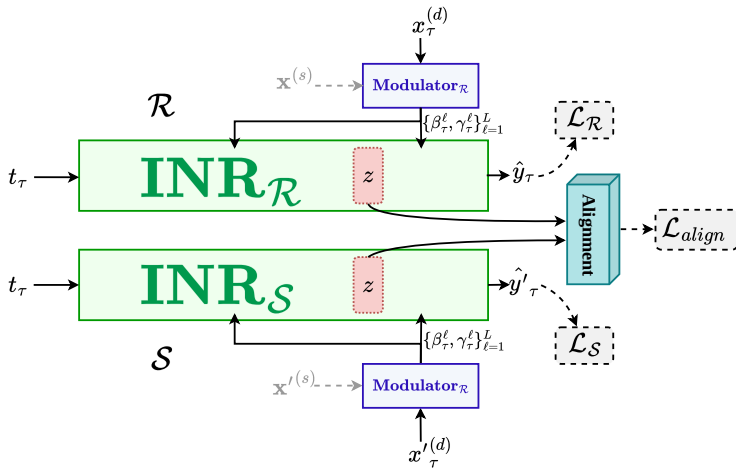
Transfer Learning for Time Series with Asymmetric Supervision



● Available y_τ for supervision



The Proposed Architecture



Modulated Implicit Neural Representations



Implicit Neural Representations consider the problem of finding a **continuous representation of data**

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$$



Implicit Neural Representations consider the problem of finding a **continuous representation of data**

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

Multi Layer Perceptron (**MLPs**) $\rightarrow$ Continuous and fully differentiable.

INR

- Coordinate based MLPs
- Arbitrary resolution at inference
- Memory efficient

Explicit representations

- Point clouds, voxel grids, meshes, ...
- Resolution is fixed at acquisition time
- Less compact representations



Implicit Neural Representations consider the problem of finding a **continuous representation of data**

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

Multi Layer Perceptron (**MLPs**) $\rightarrow$ Continuous and fully differentiable.

INR

- Coordinate based MLPs
- Arbitrary resolution at inference
- Memory efficient

Explicit representations

- Point clouds, voxel grids, meshes, ...
- Resolution is fixed at acquisition time
- Less compact representations

A major problem is spectral Bias!



Implicit Neural Representations consider the problem of finding a **continuous representation of data**

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

Multi Layer Perceptron (**MLPs**) $\rightarrow$ Continuous and fully differentiable.

INR

- Coordinate based MLPs
- Arbitrary resolution at inference
- Memory efficient

Explicit representations

- Point clouds, voxel grids, meshes, ...
- Resolution is fixed at acquisition time
- Less compact representations

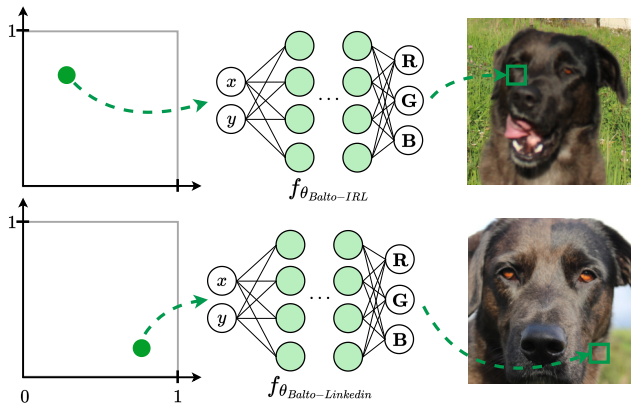
A major problem is spectral Bias!

Positional encoding or **adaptive activation functions** can capture a wide range of frequencies.

INRs for Learning Images Representations

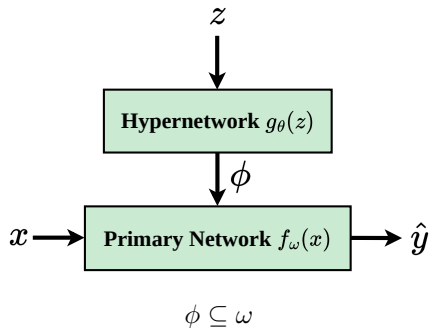


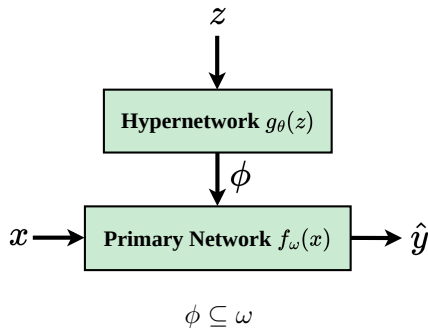
Our function of interest could be simply an image, mapping the coordinates $[x, y]$ to their pixel values $[\mathbf{R}, \mathbf{G}, \mathbf{B}]$.





Hypernetworks and Feature-Wise Linear Modulation





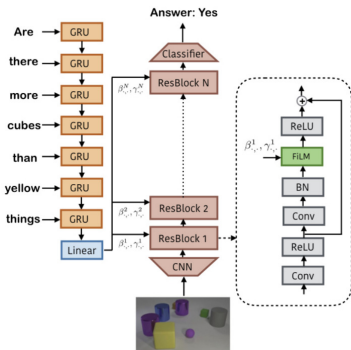
- ✓ A single hypernetwork can parameterize a family of primary networks
- ✓ High expressivity $\rightarrow$ Interesting for Multi-Task Learning
- ✗ Unstable and difficult to train
- ✗ Too expressive for most cases

Ha, Dai, and Le 2016



Feature-wise Linear Modulation (FiLM) is a method for conditioning models by inserting feature-wise transformation layers into the primary network.

$$\text{FiLM}(\mathbf{x}) = \gamma(z) \odot x + \beta(z)$$



- ✓ The FiLM generator (**modulator**) is an specialized Hypernetwork
- ✓ Sufficiently complex for most tasks
- ✓ Natural factorization of shared structure from instance-specific variation

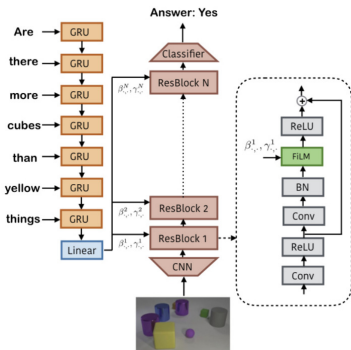
Useful for conditioning generative models, performing style transfer, ...

Perez et al. 2017



Feature-wise Linear Modulation (FiLM) is a method for conditioning models by inserting feature-wise transformation layers into the primary network.

$$\text{FiLM}(\mathbf{x}) = \gamma(z) \odot x + \beta(z)$$



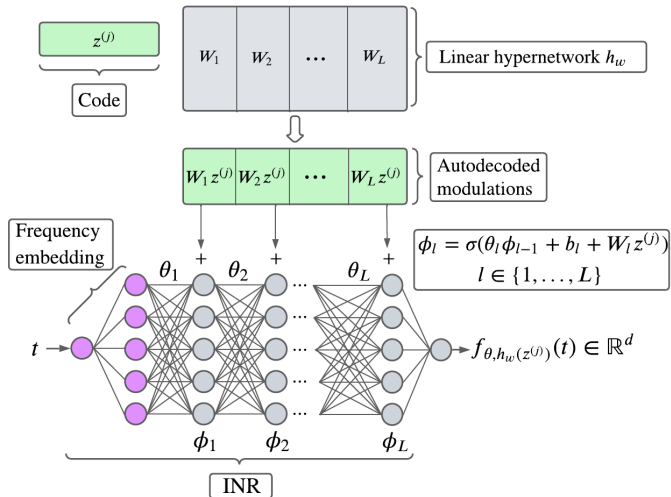
- ✓ The FiLM generator (**modulator**) is an specialized Hypernetwork
- ✓ Sufficiently complex for most tasks
- ✓ Natural factorization of shared structure from instance-specific variation (**nice for INRs ?**)

Useful for conditioning generative models, performing style transfer, ...

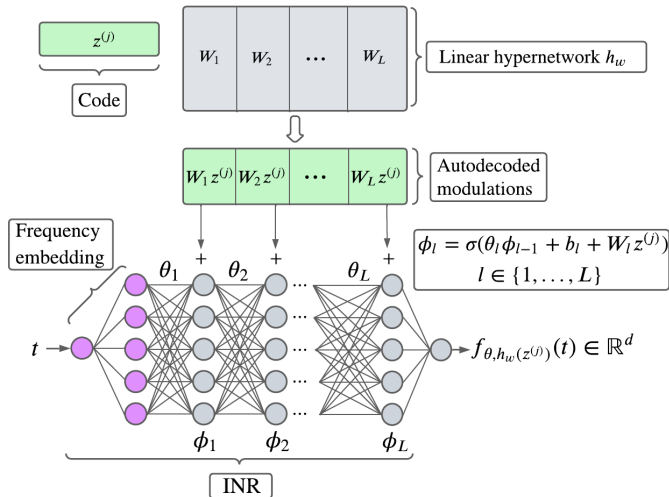
Perez et al. 2017



Modulated INRs for Time Series Modeling



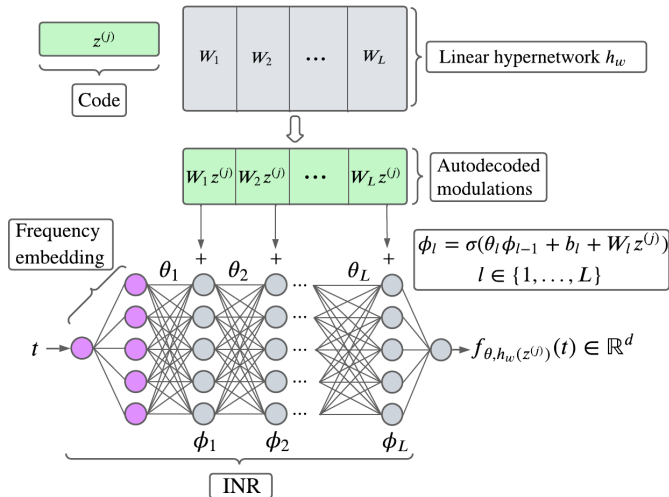
Modulated INRs for Time Series Modeling



- ✓ SOTA performance on forecasting and imputation
- ✓ Test-Time Learning Capacities (3 SGD steps)

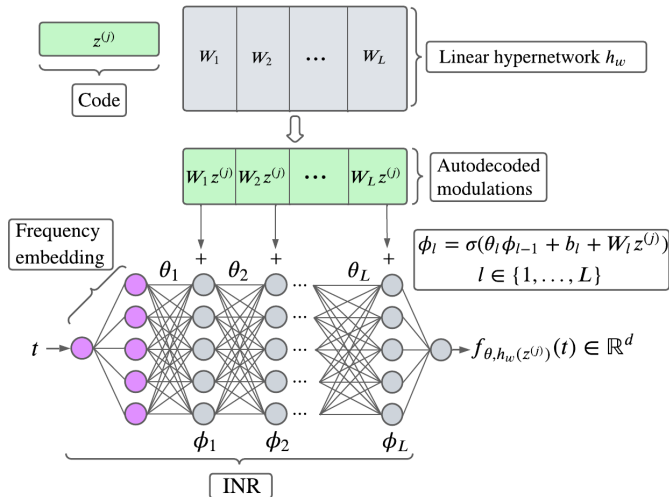


Modulated INRs for Time Series Modeling



- ✓ SOTA performance on forecasting and imputation
- ✓ Test-Time Learning Capacities (3 SGD steps)
- ✗ 2x slower at inference than other SOTA models
- ✗ Difficulty to fit time series with drastically different structure

Modulated INRs for Time Series Modeling

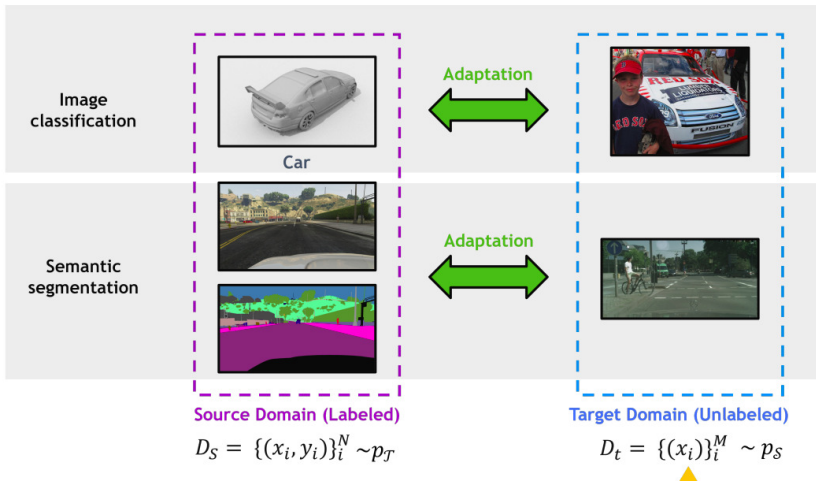


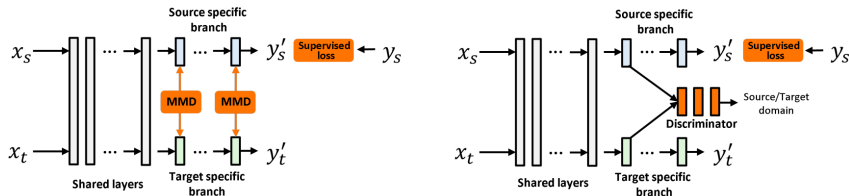
- ✓ SOTA performance on forecasting and imputation
- ✓ Test-Time Learning Capacities (3 SGD steps)
- ✗ 2x slower at inference than other SOTA models
- ✗ Difficulty to fit time series with drastically different structure

A beautiful example of using INRs for tasks other than representation

Naour et al. 2024

Lessons From Unsupervised Domain Alignment





$$\mathcal{L}_{\text{MMD}} = \mathbb{E}_{x,x'}[k(x, x')] + \mathbb{E}_{y,y'}[k(y, y')] - 2 \mathbb{E}_{x,y}[k(x, y)],$$

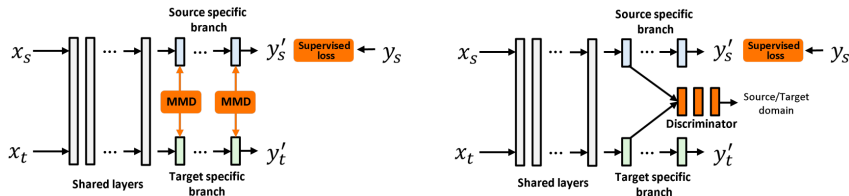
where $x, x' \sim z_{\text{synth}}$, $y, y' \sim z_{\text{real}}$, and the kernel function is $k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$

$$\mathcal{L}_{\text{adv}} = - \sum_i [y_i \log \sigma(f_i) + (1 - y_i) \log (1 - \sigma(f_i))],$$

where $y_i = 0$ denotes source samples, $y_i = 1$ denotes target samples, f_i is the discriminator logit, and $\sigma(\cdot)$ is the sigmoid function.



Unsupervised Domain Adaptation



$$\mathcal{L}_{\text{MMD}} = \mathbb{E}_{x,x'}[k(x, x')] + \mathbb{E}_{y,y'}[k(y, y')] - 2 \mathbb{E}_{x,y}[k(x, y)],$$

where $x, x' \sim z_{\text{synth}}$, $y, y' \sim z_{\text{real}}$, and the kernel function is $k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$

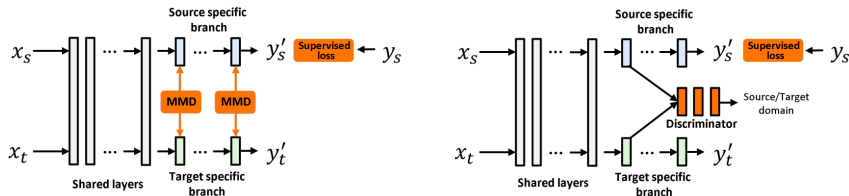
$$\mathcal{L}_{\text{adv}} = - \sum_i [y_i \log \sigma(f_i) + (1 - y_i) \log (1 - \sigma(f_i))],$$

where $y_i = 0$ denotes source samples, $y_i = 1$ denotes target samples, f_i is the discriminator logit, and $\sigma(\cdot)$ is the sigmoid function.

Liu et al. 2022



Unsupervised Domain Adaptation



$$\mathcal{L}_{\text{MMD}} = \mathbb{E}_{x,x'}[k(x, x')] + \mathbb{E}_{y,y'}[k(y, y')] - 2 \mathbb{E}_{x,y}[k(x, y)],$$

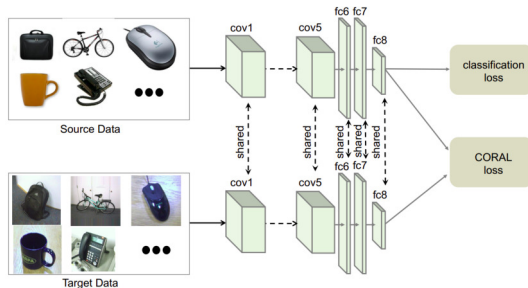
where $x, x' \sim z_{\text{synth}}$, $y, y' \sim z_{\text{real}}$, and the kernel function is $k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$

$$\mathcal{L}_{\text{adv}} = - \sum_i [y_i \log \sigma(f_i) + (1 - y_i) \log (1 - \sigma(f_i))],$$

where $y_i = 0$ denotes source samples, $y_i = 1$ denotes target samples, f_i is the discriminator logit, and $\sigma(\cdot)$ is the sigmoid function.

Liu et al. 2022

Unsupervised Domain Adaptation: CORAL



$$\mathcal{L}_{\text{CORAL}} = \frac{1}{4d^2} \|C_s - C_r\|_F^2$$

where $C_s, C_r \in \mathbb{R}^{d \times d}$ are the unbiased sample covariance matrices computed from the latent representations, and $\|\cdot\|_F$ denotes the Frobenius norm.

Sun and Saenko 2016

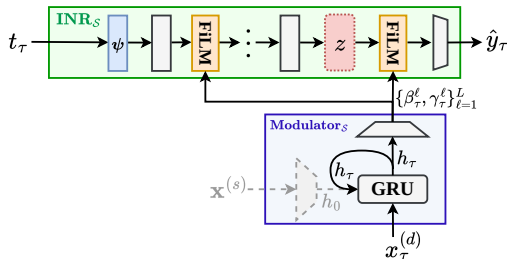


- **Kernel-Based Distribution Matching (MMD):** aligns domain distributions in a Reproducing Kernel Hilbert Space (RKHS).
- **Adversarial Feature Alignment** aligns distributions implicitly via a minimax game between feature extractor and domain discriminator.
- **Alignment via Second-Order Statistics (CORAL)** Minimizes domain shift by matching covariance matrices.

Domain Alignment on an INR-FiLM architecture



The Proposed Architecture



The Modulator receives a time series $\mathbf{x}^{(d)} = \{\mathbf{x}_\tau\}_{\tau=1}^T$ together with static covariates $\mathbf{x}^{(s)}$

The static covariates initialise the hidden state of the GRU encoder through a MLP:

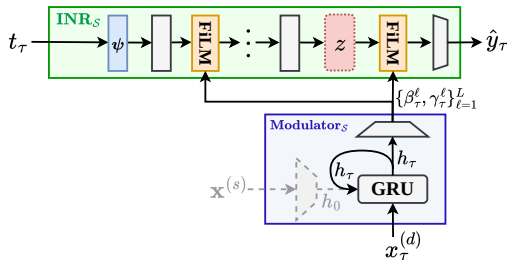
$$\mathbf{h}_0 = \text{MLP}(\mathbf{x}^{(s)})$$

Then, the GRU processes the dynamic inputs sequentially:

$$\mathbf{h}_\tau = \text{GRU}(\mathbf{x}_\tau^{(d)}, \mathbf{h}_{\tau-1}), \quad \tau = 1, \dots, T$$



The Proposed Architecture



The Modulator receives a time series $\mathbf{x}^{(d)} = \{\mathbf{x}_\tau\}_{\tau=1}^T$ together with static covariates $\mathbf{x}^{(s)}$

The static covariates initialise the hidden state of the GRU encoder through a MLP:

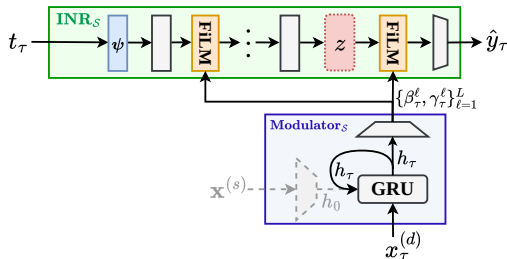
$$\mathbf{h}_0 = \text{MLP}(\mathbf{x}^{(s)})$$

Then, the GRU processes the dynamic inputs sequentially:

$$\mathbf{h}_\tau = \text{GRU}(\mathbf{x}_\tau^{(d)}, \mathbf{h}_{\tau-1}), \quad \tau = 1, \dots, T$$



The Proposed Architecture



The Modulator receives a time series $\mathbf{x}^{(d)} = \{\mathbf{x}_\tau\}_{\tau=1}^T$ together with static covariates $\mathbf{x}^{(s)}$

The static covariates initialise the hidden state of the GRU encoder through a MLP:

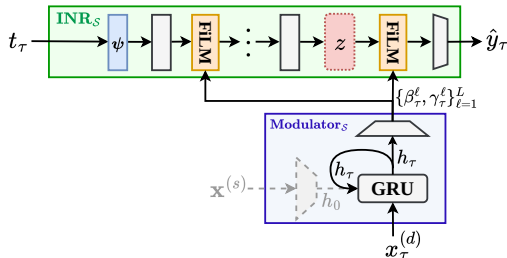
$$\mathbf{h}_0 = \text{MLP}(\mathbf{x}^{(s)})$$

Then, the GRU processes the dynamic inputs sequentially:

$$\mathbf{h}_\tau = \text{GRU}(\mathbf{x}_\tau^{(d)}, \mathbf{h}_{\tau-1}), \quad \tau = 1, \dots, T$$



The Proposed Architecture

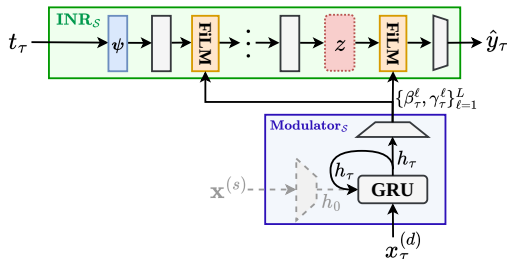


For each layer $\ell = 1, \dots, L$ of the INR and for each step τ , the modulation parameters are obtained from the current GRU hidden state $\mathbf{h}_\tau$ through linear projections:

$$\begin{aligned} \gamma_\tau^\ell &= \mathbf{W}^{\ell, \gamma} \mathbf{h}_\tau + \mathbf{b}^{\ell, \gamma}, \\ \beta_\tau^\ell &= \mathbf{W}^{\ell, \beta} \mathbf{h}_\tau + \mathbf{b}^{\ell, \beta}, \end{aligned} \quad \gamma_\tau^\ell, \beta_\tau^\ell \in \mathbb{R}^d,$$



The Proposed Architecture

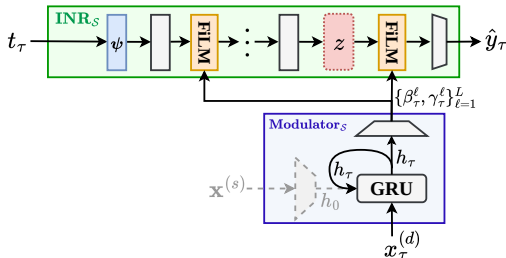


For each layer $\ell = 1, \dots, L$ of the INR and for each step τ , the modulation parameters are obtained from the current GRU hidden state $\mathbf{h}_\tau$ through linear projections:

$$\begin{aligned} \gamma_\tau^\ell &= \mathbf{W}^{\ell, \gamma} \mathbf{h}_\tau + \mathbf{b}^{\ell, \gamma}, \\ \beta_\tau^\ell &= \mathbf{W}^{\ell, \beta} \mathbf{h}_\tau + \mathbf{b}^{\ell, \beta}, \end{aligned} \quad \gamma_\tau^\ell, \beta_\tau^\ell \in \mathbb{R}^d,$$



The Proposed Architecture



The INR is evaluated independently at each time step t_τ using modulation parameters generated from $\mathbf{h}_\tau$

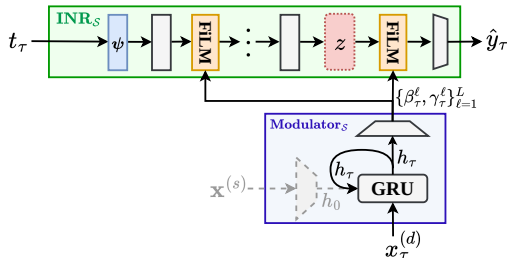
First, we use a Fourier feature encoding $\psi : [0, 1] \rightarrow \mathbb{R}^{2K}$

$$\psi(t) = [\sin(2^0 \pi t), \cos(2^0 \pi t), \dots, \sin(2^{K-1} \pi t), \cos(2^{K-1} \pi t)]$$

Mildenhall et al. 2020; Zhong et al. 2020



The Proposed Architecture



The INR is evaluated independently at each time step t_τ using modulation parameters generated from $\mathbf{h}_\tau$

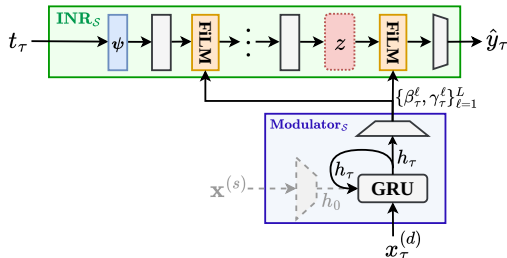
First, we use a Fourier feature encoding $\psi : [0, 1] \rightarrow \mathbb{R}^{2K}$

$$\psi(t) = [\sin(2^0 \pi t), \cos(2^0 \pi t), \dots, \sin(2^{K-1} \pi t), \cos(2^{K-1} \pi t)]$$

Mildenhall et al. 2020; Zhong et al. 2020



The Proposed Architecture



The INR is evaluated independently at each time step t_τ using modulation parameters generated from $\mathbf{h}_\tau$

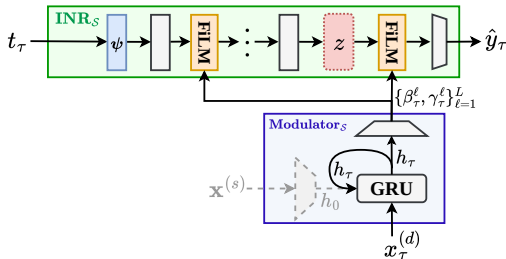
First, we use a Fourier feature encoding $\psi : [0, 1] \rightarrow \mathbb{R}^{2K}$

$$\psi(t) = [\sin(2^0 \pi t), \cos(2^0 \pi t), \dots, \sin(2^{K-1} \pi t), \cos(2^{K-1} \pi t)]$$

Mildenhall et al. 2020; Zhong et al. 2020



The Proposed Architecture



The first layer of **the INR** is $\mathbf{z}_\tau^{\ell=0} = \psi(t_\tau)$

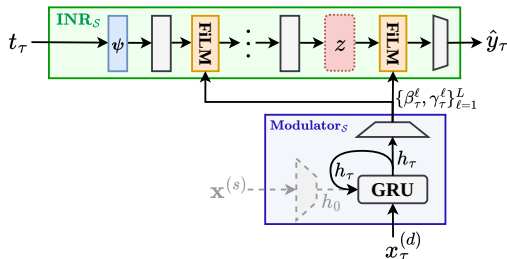
Then, each of the remaining layers computes:

$$\mathbf{z}_\tau^\ell = \gamma_\tau^\ell \odot \phi(\mathbf{W}^\ell \mathbf{z}_\tau^{\ell-1} + \mathbf{b}^\ell) + \beta_\tau^\ell, \quad \ell = 1, \dots, L,$$

where ϕ denotes the activation function and $\odot$ the element-wise multiplication.



The Proposed Architecture



The first layer of **the INR** is $\mathbf{z}_\tau^{\ell=0} = \psi(t_\tau)$

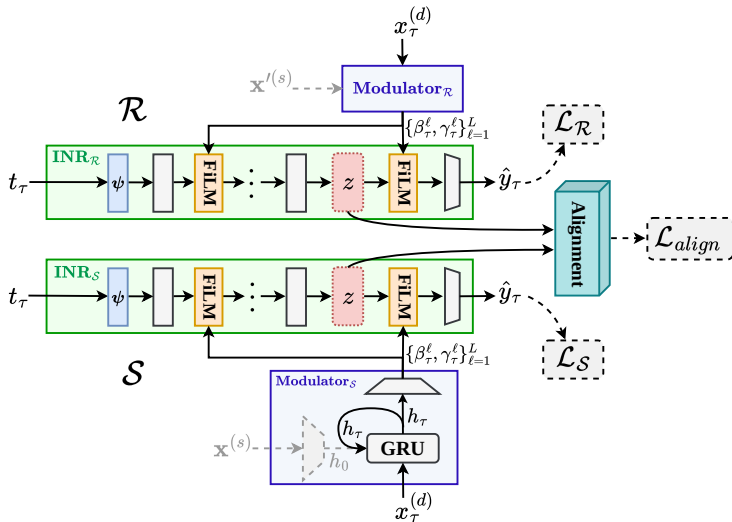
Then, each of the remaining layers computes:

$$\mathbf{z}_\tau^\ell = \gamma_\tau^\ell \odot \phi(\mathbf{W}^\ell \mathbf{z}_\tau^{\ell-1} + \mathbf{b}^\ell) + \beta_\tau^\ell, \quad \ell = 1, \dots, L,$$

where ϕ denotes the activation function and $\odot$ the element-wise multiplication.



The Proposed Architecture





Training Strategies

Alignment Techniques

MMD

$$\mathcal{L}_{\text{MMD}} = \mathbb{E}_{x, x'}[k(x, x')] + \mathbb{E}_{y, y'}[k(y, y')] - 2 \mathbb{E}_{x, y}[k(x, y)]$$

CORAL

$$\mathcal{L}_{\text{CORAL}} = \frac{1}{4d^2} \|C_s - C_r\|_F^2$$

Adv(μ, σ)

$$\mathcal{L}_{\text{adv}} = - \sum_i [y_i \log \sigma(f_i) + (1 - y_i) \log (1 - \sigma(f_i))]$$

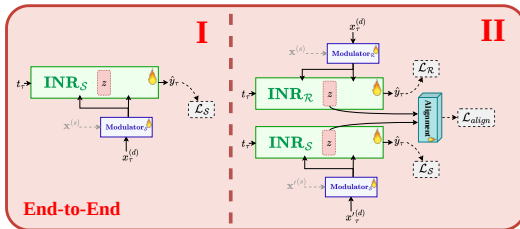
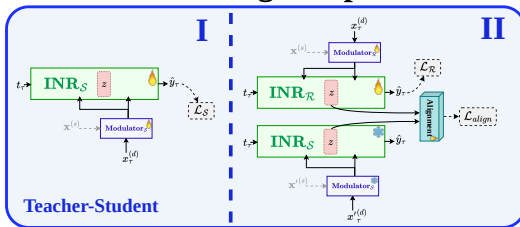
$$f_i = \text{MLP}([\mu_i, \sigma_i]), \quad \mu_i = \frac{1}{B} \sum_b h_b, \quad \sigma_i = \text{std}_b(h_b)$$

Adv(cov)

$$\mathcal{L}_{\text{adv}} = - \sum_i [y_i \log \sigma(f_i) + (1 - y_i) \log (1 - \sigma(f_i))]$$

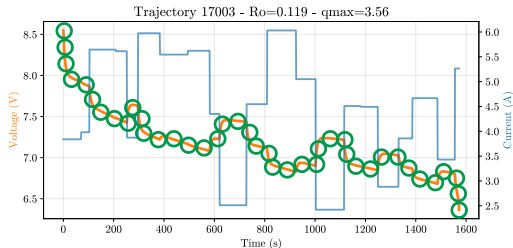
$$f_i = \mathbf{w}^\top \frac{1}{H} \sum_{j=1}^H \text{MLP}(\mathbf{C}_i[j, :]), \quad \mathbf{C}_i = \frac{1}{B-1} \tilde{H}_i^\top \tilde{H}_i$$

Training Setups



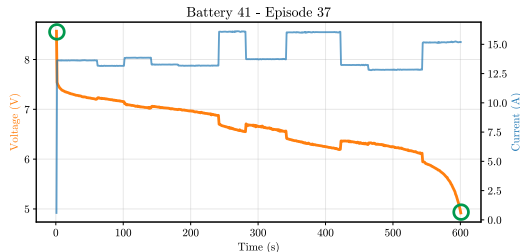
Use Case

Batteries Discharge Curve Prediction



A sample from D_S

● Available y_t for supervision



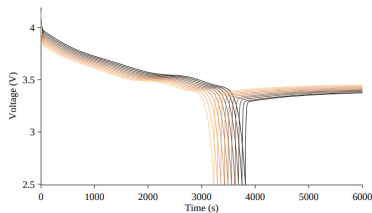
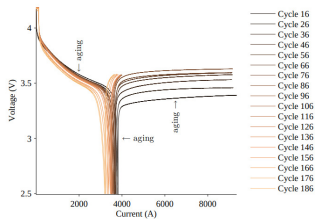
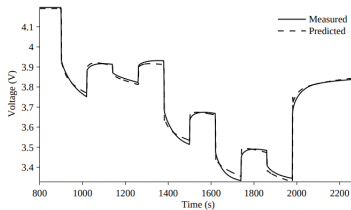
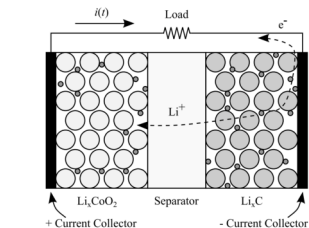
A sample from D_R

- 20000 samples: 1S, 2S and 3S
- R_0 and $q_{\max}$ (LHS)
- $\mathbf{x}^{(s)} = \{R_0, q_{\max}\}$
- 70/15/15, length [100 – 3834]

- 4382 samples from 15 batteries (2S)
- $\mathbf{x}^{(s)} = \{\text{Cycle_number}\}$
- 70/15/15, length [152 – 3693]



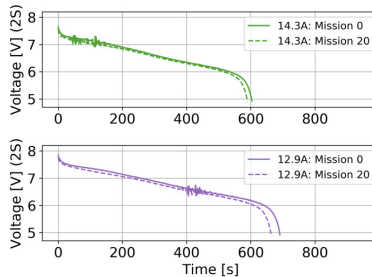
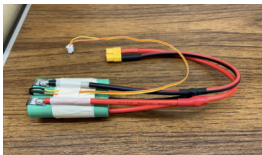
ProgPy



Daigle and Kulkarni 2013

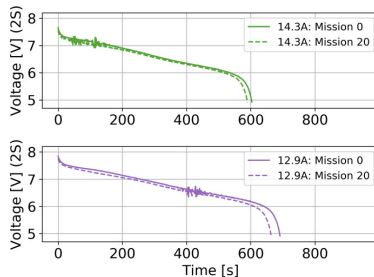
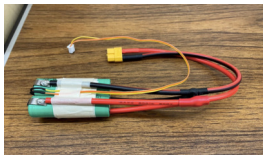


Samsung INR18650-25R





Samsung INR18650-25R



- The batteries are the same model as the ones used for the physical model's default parameters !
- We know the entire voltage sequence
- **Non-uniform time step** and **good representation of sensor artifacts**.

Fricke et al. 2023

Results



Quantitative Results

Alignment method	Simulated Domain $\mathcal{S}$				Real-world Domain $\mathcal{R}$					
	T-S		E2E		T-S			E2E		
	MSE	V_{mon}	MSE	V_{mon}	$MSE_{\{0,T\}}$	V_{mon}	MSE	$MSE_{\{0,T\}}$	V_{mon}	MSE
MMD	0.075	0.397	0.084	0.369	9.41e-4	0.380	0.337	9.29e-4	0.363	0.301
CORAL	0.074	0.377	0.081	0.397	7.18e-4	0.062	0.494	7.84e-4	0.199	0.147
Adv(μ, σ)	0.071	0.333	0.091	0.425	5.65e-4	0.451	0.391	1.98e-3	0.475	0.870
Adv(cov)	0.075	0.384	0.086	0.394	8.26e-4	0.028	0.897	1.43e-3	0.407	0.446

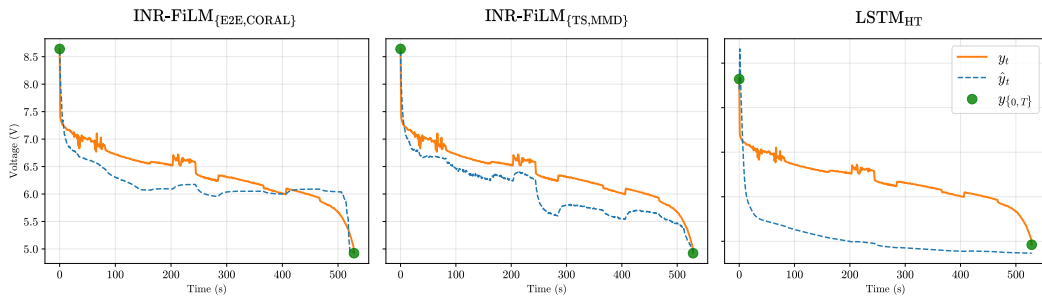


Quantitative Results

Alignment method	Simulated Domain $\mathcal{S}$				Real-world Domain $\mathcal{R}$					
	T-S		E2E		T-S			E2E		
	MSE	V_{mon}	MSE	V_{mon}	$MSE_{\{0,T\}}$	V_{mon}	MSE	$MSE_{\{0,T\}}$	V_{mon}	MSE
MMD	0.075	0.397	0.084	0.369	9.41e-4	0.380	0.337	9.29e-4	0.363	0.301
CORAL	0.074	0.377	0.081	0.397	7.18e-4	0.062	0.494	7.84e-4	0.199	0.147
Adv(μ, σ)	0.071	0.333	0.091	0.425	5.65e-4	0.451	0.391	1.98e-3	0.475	0.870
Adv(cov)	0.075	0.384	0.086	0.394	8.26e-4	0.028	0.897	1.43e-3	0.407	0.446

Model	$MSE_{\{0,T\}}$	MSE
LSTM _{FT}	0.371	1.851
LSTM _{HT}	0.434	1.487
INR _{FT}	0.366	2.968
INR _{mod}	0.363	2.600
INR – FiLM _{E2E,CORAL} (Best Ours)	7.84e-4	0.147

Qualitative Results

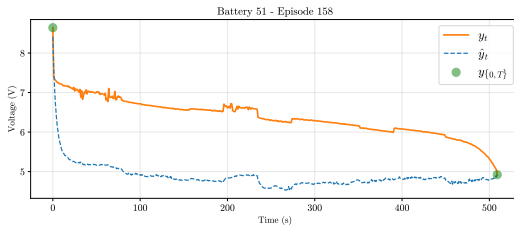




✖ Sensitivity to simulation coverage → **Expert Knowledge**



✗ Sensitivity to simulation coverage → Expert Knowledge



✗ Lack of quantitative evaluation



So What?

- ✗ Sensitivity to simulation coverage → **Expert Knowledge**
- ✗ Lack of quantitative evaluation
- ✓ **The results are promising!** we addressed a very difficult inverse problem **compensating the lack of supervision with the knowledge gained from learning a surrogate**



So What?

- ✗ Sensitivity to simulation coverage → **Expert Knowledge**
- ✗ Lack of quantitative evaluation
- ✓ **The results are promising!** we addressed a very difficult inverse problem **compensating the lack of supervision with the knowledge gained from learning a surrogate**
- ✓ Continuous representations are interesting in Sim-to-Real Transfer



So What?

- ✗ Sensitivity to simulation coverage → **Expert Knowledge**
- ✗ Lack of quantitative evaluation
- ✓ **The results are promising!** we addressed a very difficult inverse problem **compensating the lack of supervision with the knowledge gained from learning a surrogate**
- ✓ Continuous representations are interesting in Sim-to-Real Transfer
- ✓ Our main contribution are open questions:



So What?

- ✗ Sensitivity to simulation coverage → **Expert Knowledge**
- ✗ Lack of quantitative evaluation
- ✓ **The results are promising!** we addressed a very difficult inverse problem **compensating the lack of supervision with the knowledge gained from learning a surrogate**
- ✓ Continuous representations are interesting in Sim-to-Real Transfer
- ✓ Our main contribution are open questions:
 - How much Expert Knowledge is enough?



So What?

- ✗ Sensitivity to simulation coverage → **Expert Knowledge**
- ✗ Lack of quantitative evaluation
- ✓ **The results are promising!** we addressed a very difficult inverse problem **compensating the lack of supervision with the knowledge gained from learning a surrogate**
- ✓ Continuous representations are interesting in Sim-to-Real Transfer
- ✓ Our main contribution are open questions:
 - How much Expert Knowledge is enough?
 - How can we build metrics?



So What?

- ✗ Sensitivity to simulation coverage → **Expert Knowledge**
- ✗ Lack of quantitative evaluation
- ✓ **The results are promising!** we addressed a very difficult inverse problem **compensating the lack of supervision with the knowledge gained from learning a surrogate**
- ✓ Continuous representations are interesting in Sim-to-Real Transfer
- ✓ Our main contribution are open questions:
 - How much Expert Knowledge is enough?
 - How can we build metrics?
 - Is our task feasible?



WHEN IT'S BETTER TO CALL YOURSELF A...

	"STUDENT"	"RESEARCHER"
GETTING A STUDENT DISCOUNT	✓	✗
REGISTERING FOR A CONFERENCE	✓	✗
USING STUDENT FACILITIES (HEALTH CENTER, GYM, HOUSING)	✓	✗
TALKING TO FRIENDS AND FAMILY	✗	✓
CRASHING DEPARTMENT PARTIES WITH SIGNS THAT SAY "NO STUDENTS ALLOWED"	✗	✓
FINDING PARKING ON CAMPUS	✗	✓
IMPRESSING SOMEONE ON A FIRST DATE	✗	✗

JORGE CHAM © 2016

WWW.PHDCOMICS.COM



-  Daigle, Matthew and Chetan S Kulkarni (2013). “Electrochemistry-based Battery Modeling for Prognostics”. en. In.
-  Fricke, Kajetan et al. (2023). “An Accelerated Life Testing Data set for Lithium-Ion Batteries with Constant and Variable Loading Conditions”. en. In.
-  Ha, David, Andrew Dai, and Quoc V. Le (2016). *HyperNetworks*. arXiv: 1609.09106 [cs.LG]. URL: <https://arxiv.org/abs/1609.09106>.
-  Jin, Xiaoyong et al. (2022). *Domain Adaptation for Time Series Forecasting via Attention Sharing*. arXiv: 2102.06828 [cs.LG]. URL: <https://arxiv.org/abs/2102.06828>.
-  Liu, Xiaofeng et al. (2022). *Deep Unsupervised Domain Adaptation: A Review of Recent Advances and Perspectives*. arXiv: 2208.07422 [cs.CV]. URL: <https://arxiv.org/abs/2208.07422>.



-  Mildenhall, Ben et al. (Aug. 2020). *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. en. arXiv:2003.08934 [cs]. DOI: 10.48550/arXiv.2003.08934. URL: <http://arxiv.org/abs/2003.08934> (visited on 03/25/2026).
-  Naour, Etienne Le et al. (2024). *Time Series Continuous Modeling for Imputation and Forecasting with Implicit Neural Representations*. arXiv: 2306.05880 [cs.LG]. URL: <https://arxiv.org/abs/2306.05880>.
-  Ozyurt, Yilmazcan, Stefan Feuerriegel, and Ce Zhang (2023). “Contrastive Learning for Unsupervised Domain Adaptation of Time Series”. In: *ICLR*.
-  Pan, Sinno Jialin and Qiang Yang (Oct. 2010). “A Survey on Transfer Learning”. en. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, pp. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. URL: <http://ieeexplore.ieee.org/document/5288526/> (visited on 03/23/2026).
-  Perez, Ethan et al. (2017). *FiLM: Visual Reasoning with a General Conditioning Layer*. arXiv: 1709.07871 [cs.CV]. URL: <https://arxiv.org/abs/1709.07871>.



-  Sun, Baochen and Kate Saenko (2016). *Deep CORAL: Correlation Alignment for Deep Domain Adaptation*. arXiv: 1607.01719 [cs.CV]. URL: <https://arxiv.org/abs/1607.01719>.
-  Zhong, Ellen D. et al. (Feb. 2020). *Reconstructing continuous distributions of 3D protein structure from cryo-EM images*. en. arXiv:1909.05215 [q-bio]. DOI: 10.48550/arXiv.1909.05215. URL: <http://arxiv.org/abs/1909.05215> (visited on 03/25/2026).