

# Advances in Conditional Neural Processes

Daolang Huang

*Mathematical Perspective on Machine Learning Seminar*

21.11.2023

# What is Amortized Inference?

- Amortization refers to the ability of a conditional distribution approximation to condition on **arbitrary instances** of the conditioning variable, rather than being specialized to a particular instance.

Too abstract!

# Example: Planning for a trip

Traditional trip planning



- Starts from scratch every time.
- Time-consuming and complex.
- Each trip planned individually.

With a smart travel assistant



- Learns from past trips.
- Quick, personalized planning.
- Efficient use of past data for new trips.

# Example: Inference for simulators

- Simulator-based model  $\mathcal{P}_\Theta = \{\mathbb{P}_\theta : \theta \in \Theta\}$ .
  - A climate simulator: given conditions (temperature, humidity, and wind speed, etc.), simulate states and behaviors of the atmosphere, oceans, or land.
  - Input: **parameters**. Output: **simulated data**.
- $\mathbb{P}_\theta$  is **intractable**, but simulating data  $\mathbf{x}_{1:n} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \sim \mathbb{P}_\theta$  is straightforward.
  - Likelihood function is unknown.
- Observed data  $\mathbf{x}^* \subseteq \mathbb{R}^d$  denoted by empirical distribution  $\mathbb{Q}$ .
  - Real-world states and behaviors of the atmosphere, oceans, or land.
- Aim: Estimate  $\theta^*$  given data  $\mathbf{x}^*$  such that  $\mathbb{P}_{\theta^*} = \mathbb{Q}$ .

# Example: Inference for simulators

Traditional inference method: *Approximate Bayesian Computation (ABC)*

Repeat until  $m$  samples accepted:

1. Sample  $\tilde{\theta}$  from prior  $p(\theta)$ .
2. Simulate data  $\mathbf{x} \sim \mathbb{P}_{\tilde{\theta}}$ .
3. If  $d(\mathbf{x}, \mathbf{x}^*) < \epsilon$ , accept  $\tilde{\theta}$ .

# Example: Inference for simulators

Traditional inference method: *Approximate Bayesian Computation (ABC)*

Assume  $\mathbf{x}^* = 2$ ,  $\theta^* = 1$ ,  $\epsilon = 0.05$ .

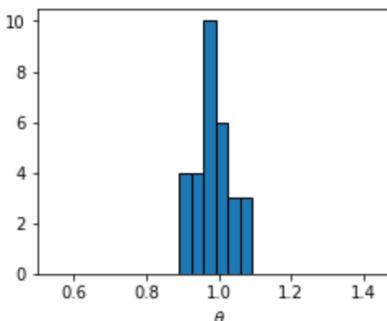
1<sup>st</sup> round:

1. Sample from prior  $\tilde{\theta} = 1.05$ .
2. Simulate data  $\mathbf{x} = 2.03$ .
3.  $d(\mathbf{x}, \mathbf{x}^*) = 0.03 < \epsilon$ , accept  $\tilde{\theta}$ .

2<sup>nd</sup> round:

1. Sample from prior  $\tilde{\theta} = 1.4$ .
2. Simulate data  $\mathbf{x} = 2.97$ .
3.  $d(\mathbf{x}, \mathbf{x}^*) = 0.97 > \epsilon$ , reject  $\tilde{\theta}$ .

...



What about a new piece of observed data  $\mathbf{x}^* = 2.001$ ?

1<sup>st</sup> round: ...

2<sup>nd</sup> round: ...

...

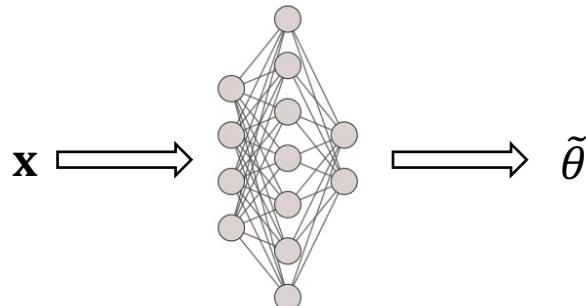
*Not efficient!*

# Example: Inference for simulators

Amortized inference method: we can build a direct mapping -> **Neural Posterior Estimation (NPE)**

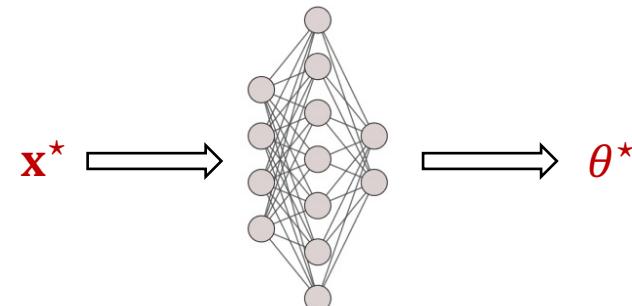
Training a neural network:

1. Sample  $\{\tilde{\theta}_i\}_{i=1}^n$  from prior.
2. Simulate corresponding data  $\{x_i\}_{i=1}^n$ .
3. Learn a map from  $x$  to  $\tilde{\theta}$ .



Given **any** observed data  $x^*$ :

1. Pass  $x^*$  to the trained neural network.
2. Get the estimated posterior.



# Amortized Inference

$$P(\theta^* | \mathbf{x}^*)$$

- Amortization refers to the ability of a conditional distribution approximation to condition on **arbitrary instances** of the conditioning variable, rather than being specialized to a particular instance.
- A plain definition: Training a deep neural networks as a surrogate model for a lengthy computational process.
- The amortization consists of the fact that we typically need to spend a lot of computations upfront to train the neural network, once it's trained it acts as an oracle that gives us the answer with a single forward pass.

# From Amortized Inference to Meta-learning

- Core idea: Allow models to efficiently make predictions on new data by leveraging patterns learned from past data.



# Meta-Learning over Sets

- Context set (black points):

$$(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$$

- Target set (red points):

$$\mathbf{X}^* = (\mathbf{x}_1^*, \dots, \mathbf{x}_M^*)$$

- A set of context sets:

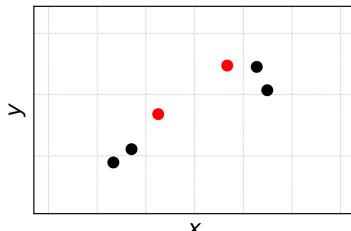
$$\{(\mathbf{X}_k, \mathbf{Y}_k)\}_{k=1}^K = \{(\mathbf{X}_1, \mathbf{Y}_1), \dots, (\mathbf{X}_K, \mathbf{Y}_K)\}$$

- A set of target sets:

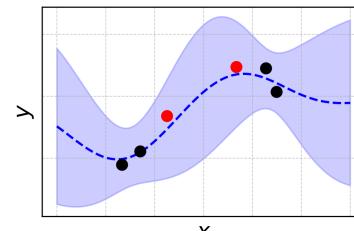
$$\{\mathbf{X}_k^*\}_{k=1}^K = \{\mathbf{x}_1^*, \dots, \mathbf{x}_K^*\}$$

- Task: build a single model to make instant predictions for  $\mathbf{Y}^*$  given any new context set and target set.
- Applications: personalized medicine, recommendation systems...

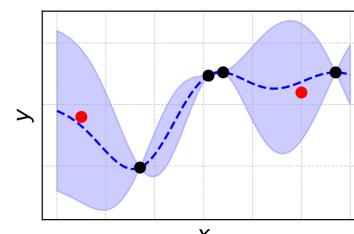
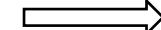
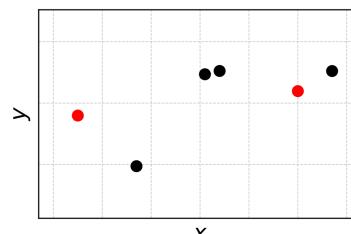
Data sets



Predictions



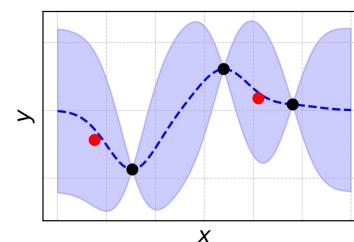
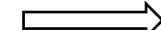
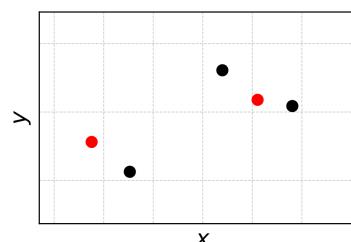
⋮



training

---

test



# Prediction Maps

- A **prediction map**  $\pi$  maps a context set and a set of target inputs to a distribution over the corresponding target outputs :

$$\pi(\mathbf{Y}^* | (\mathbf{X}, \mathbf{Y}), \mathbf{X}^*) = p(\mathbf{Y}^* | \mathbf{r}),$$

where  $\mathbf{r} = r((\mathbf{X}, \mathbf{Y}), \mathbf{X}^*)$  is a representation vector which parameterizes the distribution over  $\mathbf{Y}^*$ .

# Conditional Neural Processes

Conditional Neural Processes (CNPs)<sup>1</sup> are a class of meta-learning models.

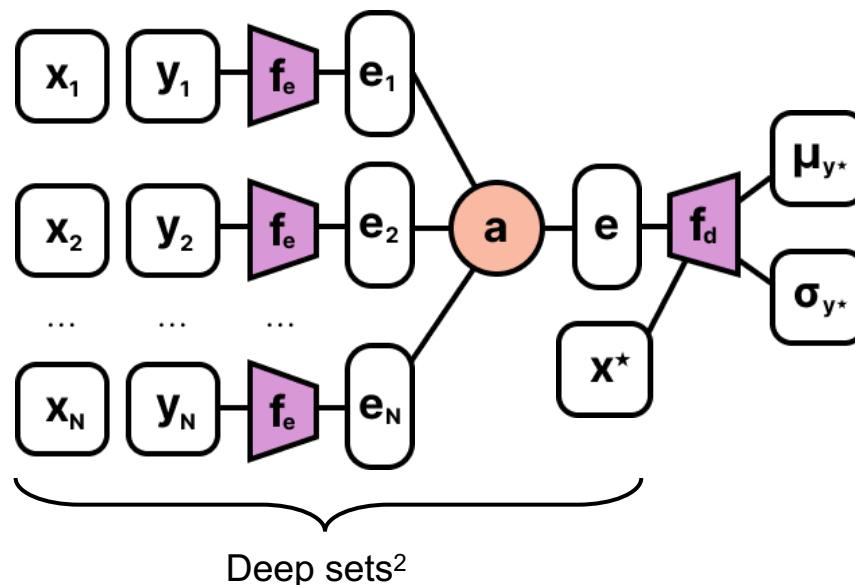
- Leverage the concept of amortized inference to deal with diverse data efficiently.
- Provide well-calibrated uncertainty estimation.

<sup>1</sup> Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M. & Eslami, S. A. (2018, July). Conditional neural processes. In *International conference on machine learning* (pp. 1704-1713). PMLR.

# Conditional Neural Processes

## Components

- **Encoder** ( $f_e$ ): a **shared** neural network among all input-output pairs.
- **Aggregator** ( $a$ ): a sum or mean operator to ensure **permutation invariance**.
- **Decoder** ( $f_d$ ): a neural network which makes the prediction over the target outputs.



<sup>2</sup> Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., & Smola, A. J. (2017). Deep sets. *Advances in neural information processing systems*, 30.

# Conditional Neural Processes

CNP is a valid prediction map:

$$\pi(\mathbf{Y}^* | (\mathbf{X}, \mathbf{Y}), \mathbf{X}^*) = \prod_{m=1}^M p(y_m^* | r_m),$$

where each  $p(y_m^* | r_m)$  is an independent Gaussian.

## Training

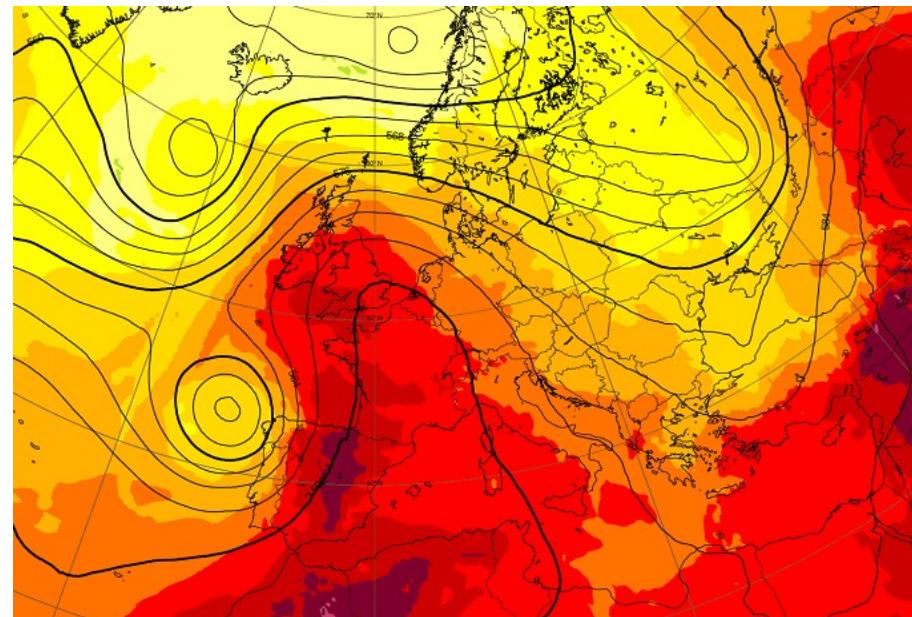
- Given closed-form Gaussian likelihood, we can optimize the parameters via maximum likelihood:

$$\theta^* = \arg \max_{\theta \in \Theta} \mathbb{E}_D [p(\mathbf{Y}^* | \mathbf{r})]$$

$$\theta^* = \arg \max_{\theta \in \Theta} \mathbb{E}_D \left[ \sum_{\mathbf{y}^* \in \mathbf{Y}^*} \log \mathcal{N}(\mathbf{y}^*; \mu_{\mathbf{y}^*}, \sigma_{\mathbf{y}^*}) \right]$$

# Dependent predictions

- Not all outputs are **independent**
  - E.g., Predicting heat waves or floods requires modelling dependencies in temperature or precipitation over time and space.
- CNPs are factored models which do **not** produce **correlated predictions**.



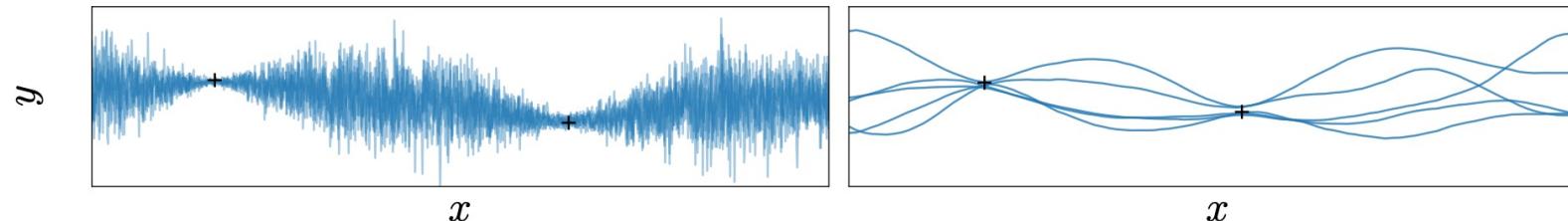
# Gaussian Neural Processes (GNPs)<sup>3</sup>

- Directly parameterize the mean the **covariance** over the output variables:

$$\pi(\mathbf{Y}^* | (\mathbf{X}, \mathbf{Y}), \mathbf{X}^*) = \mathcal{N}(\mathbf{Y}^*; \mathbf{m}, \mathbf{K}).$$

- **m** and **K** are directly parameterized by neural networks:

$$\mathbf{m} = f_m(\mathbf{e}, \mathbf{X}^*), \quad \mathbf{K}_{ij} = k(f_{cov}(\mathbf{x}_i^*, \mathbf{e}), f_{cov}(\mathbf{x}_j^*, \mathbf{e}))$$



CNP (left) makes independent predictions, GNP (right) makes dependent predictions and can be used to draw function samples which are coherent.

<sup>3</sup>Markou, S., Requeima, J., Bruinsma, W. P., Vaughan, A., & Turner, R. E. (2022). Practical conditional neural processes via tractable dependent predictions. *arXiv preprint arXiv:2203.08775*.

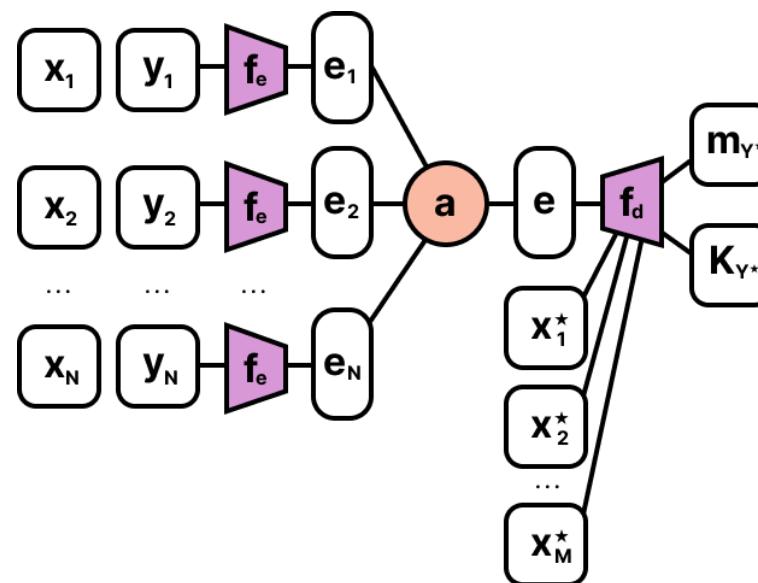
# Gaussian Neural Processes (GNPs)

- Directly parameterize the mean the **covariance** over the output variables:

$$\pi(\mathbf{Y}^* | (\mathbf{X}, \mathbf{Y}), \mathbf{X}^*) = \mathcal{N}(\mathbf{Y}^*; \mathbf{m}, \mathbf{K}).$$

- **m** and **K** are directly parameterized by neural networks:

$$\mathbf{m} = f_m(\mathbf{e}, \mathbf{X}^*), \quad \mathbf{K}_{ij} = k(f_{cov}(\mathbf{x}_i^*, \mathbf{e}), f_{cov}(\mathbf{x}_j^*, \mathbf{e}))$$



# Covariance

## Linear Covariance

$$\mathbf{K}_{ij} = \mathbf{f}_{cov}(\mathbf{x}_i^*, \mathbf{e})^T \mathbf{f}_{cov}(\mathbf{x}_j^*, \mathbf{e})$$

- The finite number of basis functions may limit its expressivity.

# Covariance

$k_{vv}$  Covariance

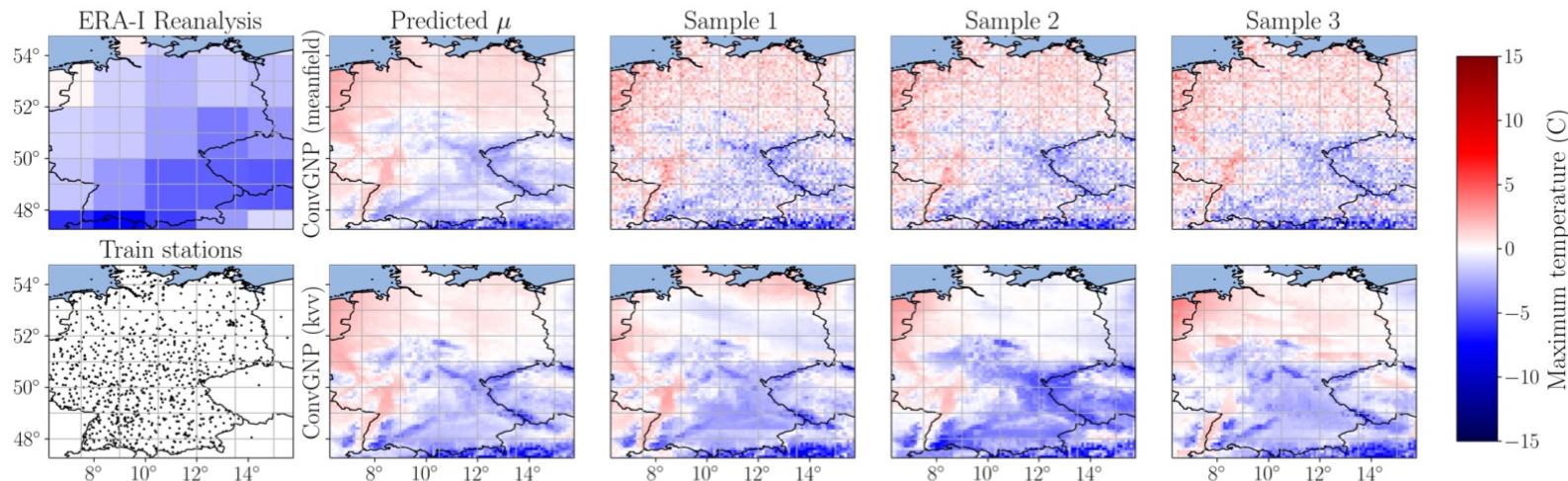
$$K_{ij} = k \left( f_{cov}(\mathbf{x}_i^*, \mathbf{e}), f_{cov}(\mathbf{x}_j^*, \mathbf{e}) \right) f_v(\mathbf{x}_i^*, \mathbf{e}) f_v(\mathbf{x}_j^*, \mathbf{e})$$

- $k$  is the **Exponentiated Quadratic (EQ)** covariance with unit length-scale and  $f_v$  is a **scalar-output** neural network which modulate the **magnitude of the covariance**.

# Experiments

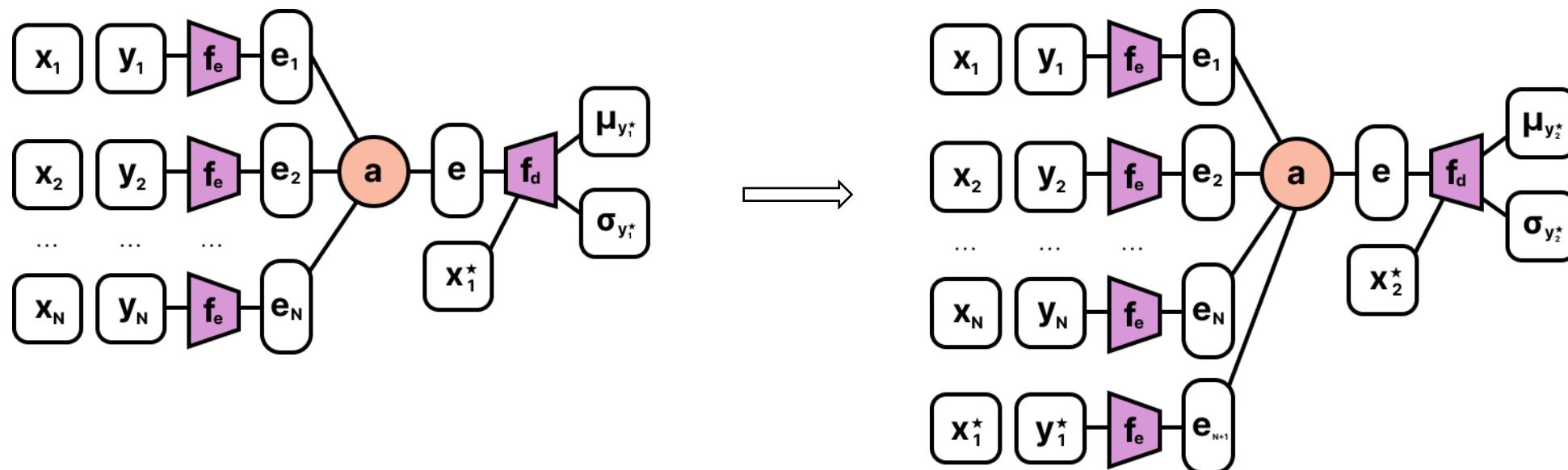
## Temperature modeling

- In climate modelling, future projections are obtained by simulating the atmospheric equations of motion on a spatial-temporal grid.
- computational constraints typically limit spatial resolution to around 100-200km, which is insufficient to resolve extreme events and produce local projections.



# Autoregressive Conditional Neural Processes (AR-CNPs)<sup>4</sup>

- Define a joint predictive distribution autoregressively, leveraging the chain rule of probability.
- AR-CNPs feed earlier output predictions back into the model autoregressively to predict new points.



<sup>4</sup> Bruinsma, W. P., Markou, S., Requima, J., Foong, A. Y., Andersson, T. R., Vaughan, A., ... & Turner, R. E. (2023). Autoregressive conditional neural processes. arXiv preprint arXiv:2303.14468.

# Autoregressive Conditional Neural Processes (AR-CNPs)

Advantages:

- Correlated and non-Gaussian predictions.
- No modifications to model or training procedure.

Disadvantages:

- Predictions depend on number and order of data (no longer consistent).
- Requires multiple forward passes of CNP.

# Other Conditional Neural Processes

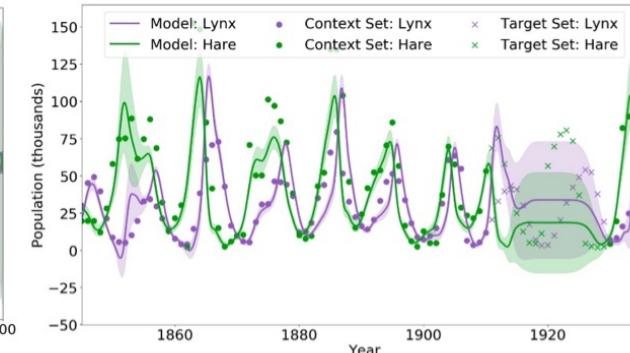
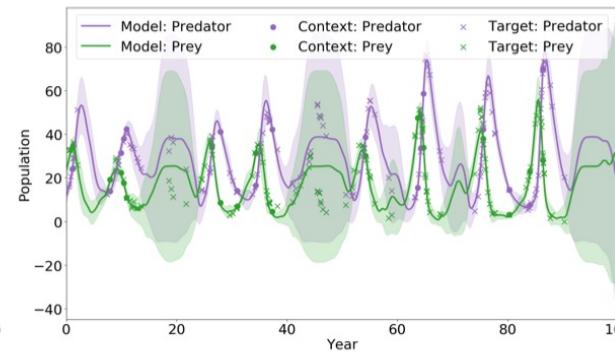
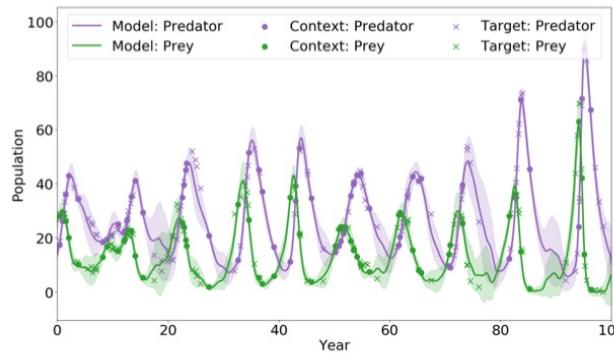
- Attentive CNPs (ACNPs)<sup>5</sup>: Add attention mechanism to CNPs.
  - Task: Make the prediction for  $x^* = 1.1$ .
  - Our context set includes various observed points, one of which is very close to our point of interest  $x = 1.0$ .
  - In ACNPs, **not all context points contribute equally to the prediction.**
  - The model 'pays more attention' to  $x = 1.0$ , as it's closer to our point of interest.
- Transformer CNPs (TCNPs)<sup>6</sup>: Replace the linear layers with Transformer layers.

<sup>5</sup> Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., ... & Teh, Y. W. (2019). Attentive neural processes. arXiv preprint arXiv:1901.05761.

<sup>6</sup> Nguyen, T., & Grover, A. (2022). Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. arXiv preprint arXiv:2207.04179.

# Uses of CNPs

1. Sets of datasets with
  - Small size
  - Irregular and missing data
2. Continual learning (naturally supports incremental updates)
3. Sim2Real (train on data from simulator; deploy on real data)

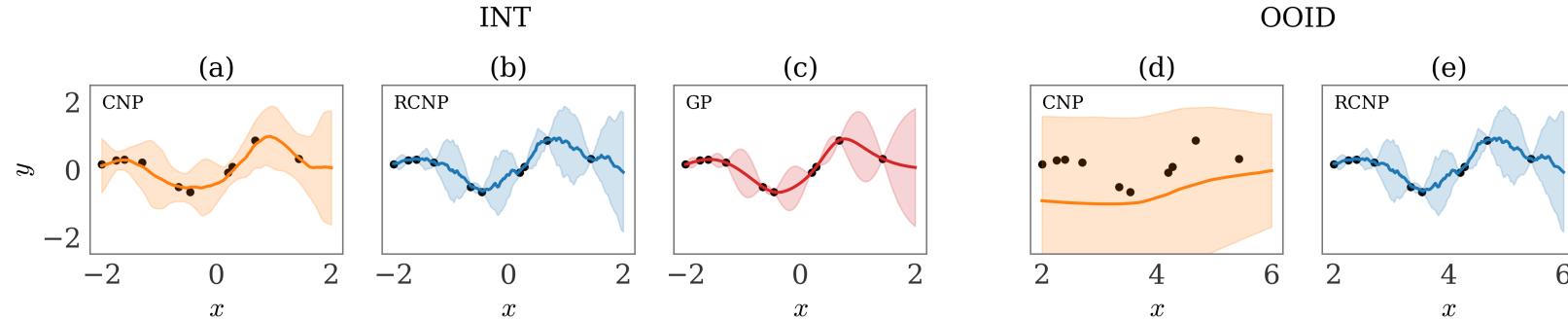


# Practical Equivariances via Relational Conditional Neural Processes

Daolang Huang Manuel Haussmann Ulpu Remes ST John

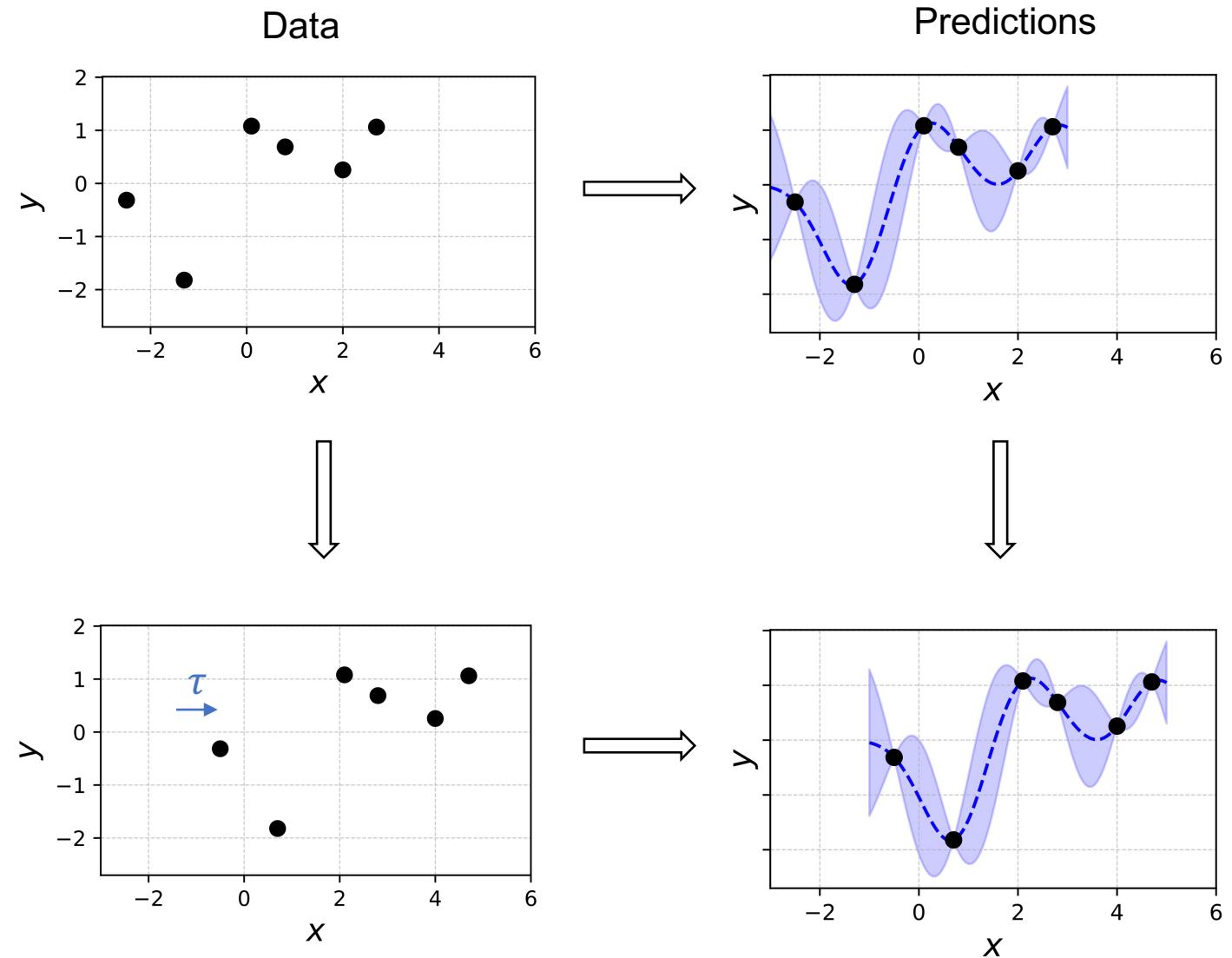
Grégoire Clarté Kevin Sebastian Luck Samuel Kaski Luigi Acerbi

NeurIPS 2023



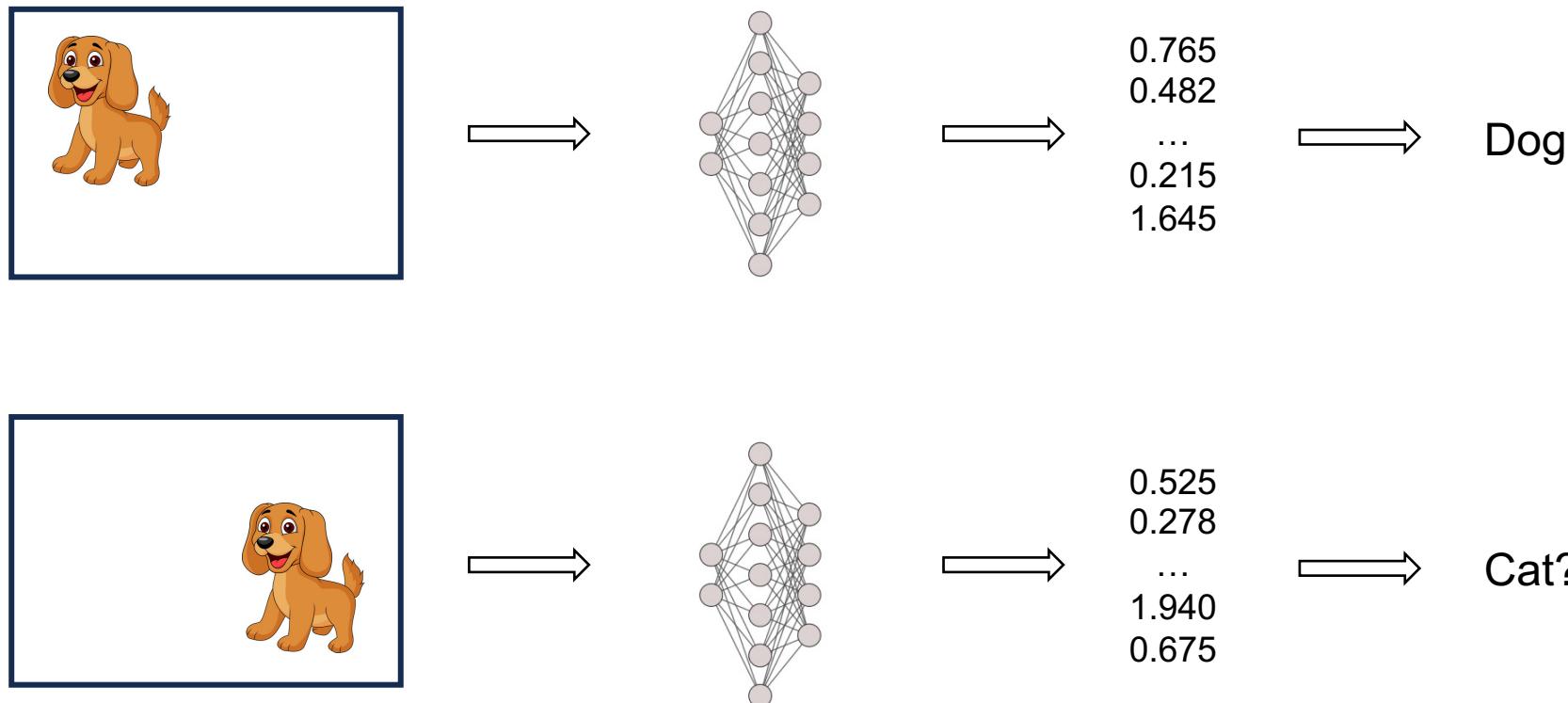
# Equivariances

- Definition:  
$$f(\tau\mathbf{x}) = \tau f(\mathbf{x})$$
- Spatial-temporal modeling, image recognition...



# Equivariances

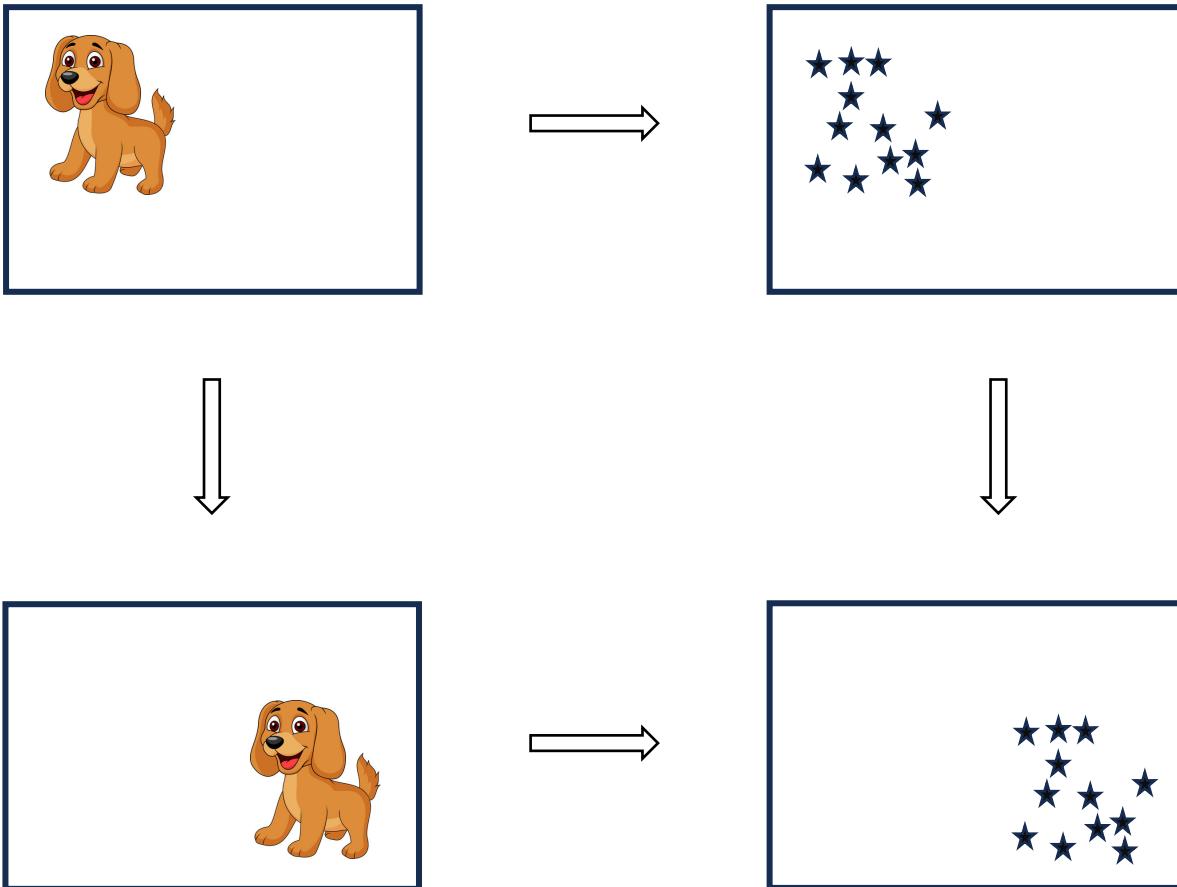
- Neural network can learn equivariance from the data, but inefficient.



# Equivariances

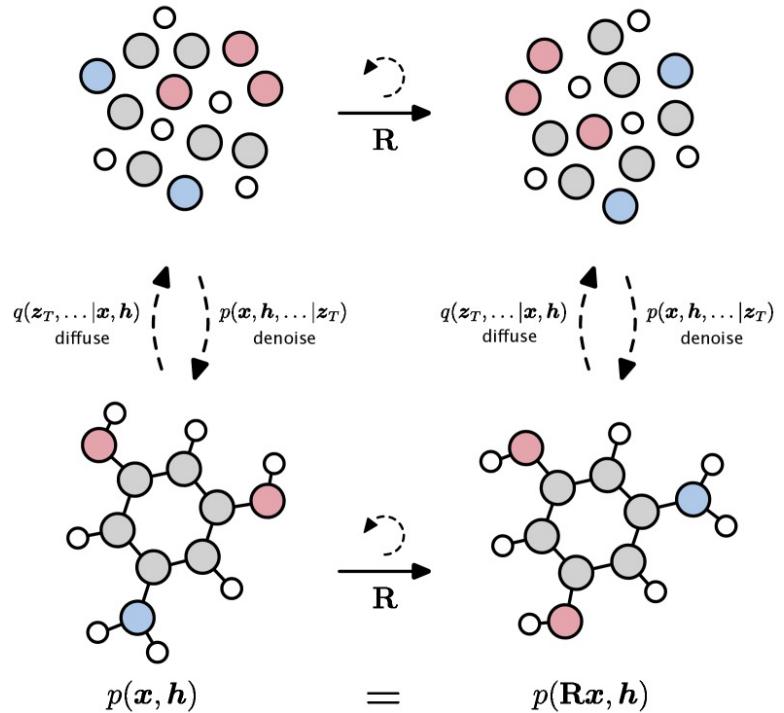
- How about directly build equivariances into NN?

Convolutional neural network  
vs  
Deep neural network

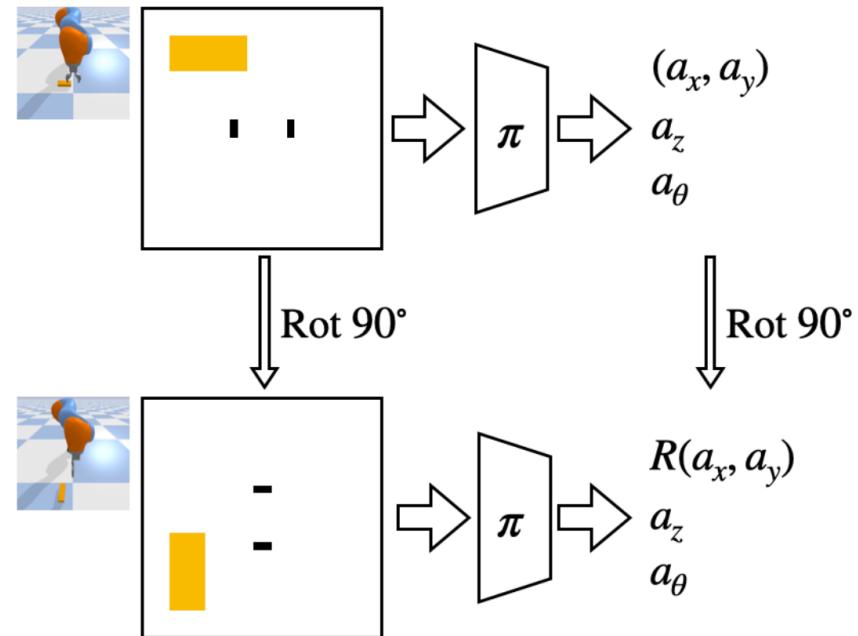


# Equivariances in Our Life

## Molecular sciences



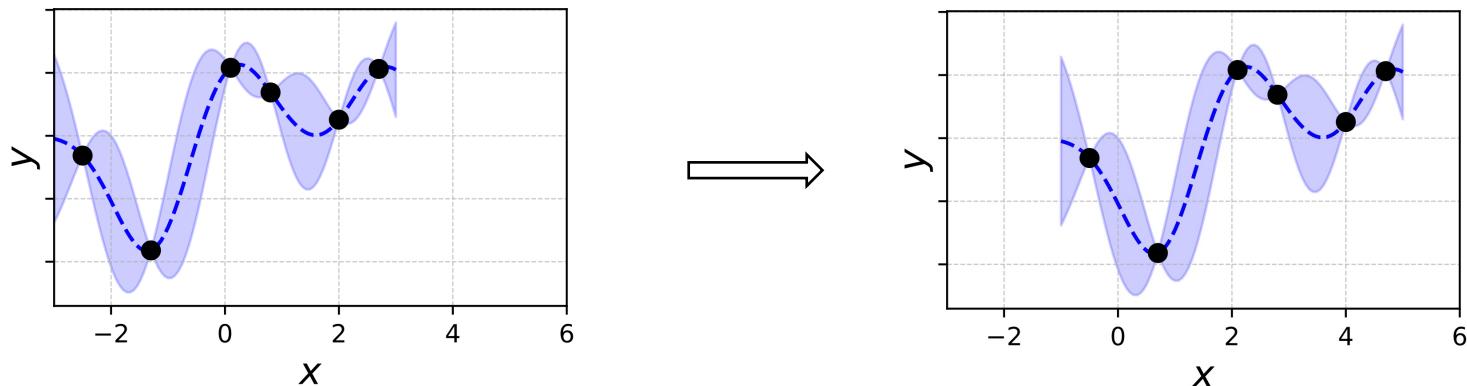
## Robotics



# Equivariance for a prediction map

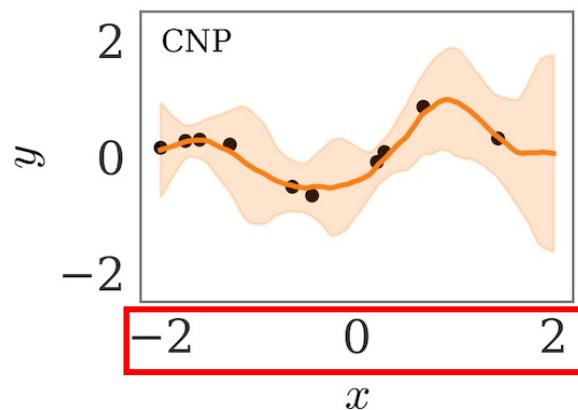
- A prediction map  $\pi$  with representation function  $r$  is  $\mathcal{T}$ -equivariant if and only if for all  $\tau \in \mathcal{T}$ :

$$r((\mathbf{X}, \mathbf{Y}), \mathbf{X}^*) = r((\tau \mathbf{X}, \mathbf{Y}), \tau \mathbf{X}^*).$$

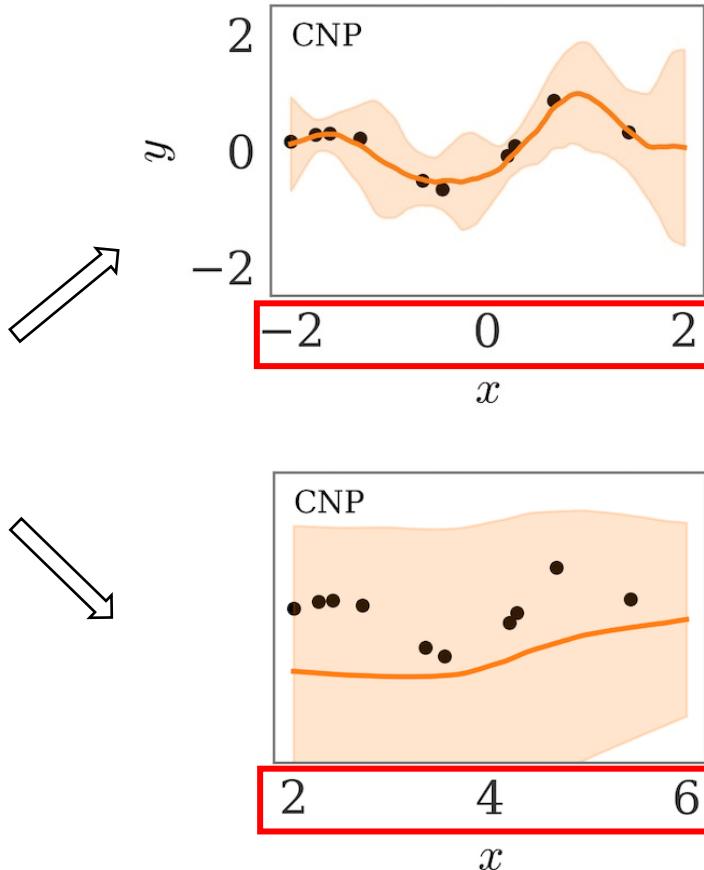


# Equivariance in CNPs

- CNPs do not model equivariances.



**training**



**evaluation**

# Equivariance in CNPs

Previous solution:

## *Convolutional CNPs*<sup>2</sup>

- Leverage a convolutional deep sets to induce translational-equivariant embeddings.
- **Can only apply to very low dimensional equivariant inputs.**
  - Building a grid is infeasible in high dimension.

<sup>2</sup> Gordon, J., Bruinsma, W. P., Foong, A. Y., Requeima, J., Dubois, Y., & Turner, R. E. (2019, September). Convolutional Conditional Neural Processes. In *International Conference on Learning Representations*.

# Equivariance in CNPs

## *Convolutional CNPs*

- **Can only apply to very low dimensional equivariant inputs.**
  - Convolutional operations in high dimension is not well-supported.

### Conv2d:

#### Search Results

Search finished, found 53 page(s) matching the search query.

- [conv2d](#)  
conv2d class torch.ao.nn.quantized.functional.conv2d(input, weight, bias, stride=1, padding=0, dilation=1, groups=1, padding\_mode='zeros', scale=1.0, zero\_point=0, dtype=torch.quint8)[source] Applies a 2D...
- [Conv2d](#)  
Conv2d class torch.ao.nn.qat.Conv2d(in\_channels, out\_channels, kernel\_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding\_mode='zeros', qconfig=None, device=None, dtype=None)[source] A Conv...
- [Conv2d](#)  
Conv2d class torch.ao.nn.quantized.Conv2d(in\_channels, out\_channels, kernel\_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding\_mode='zeros', device=None, dtype=None)[source] Applies a 2D c...

### Conv4d:

#### Search Results

Your search did not match any documents. Please make sure that all words are spelled correctly and that you've selected enough categories.

© Copyright 2023, PyTorch Contributors.  
Built with [Sphinx](#) using a [theme](#) provided by [Read the Docs](#).

# Equivariance in CNPs

Our contribution:

We propose **Relational CNPs (RCNPs)**<sup>4</sup>, a new family of CNPs, to efficiently **scale different equivariances to any input dimensions**.

<sup>4</sup> Huang, D., Haussmann, M., Remes, U., John, S. T., Clarté, G., Luck, K. S., ... & Acerbi, L. (2023). Practical Equivariances via Relational Conditional Neural Processes. *Neurips 2023*.

# Relational Encoding

We only encode **relative information** of the data, ensuring desired equivariances while **discarding non-essential absolute information**.

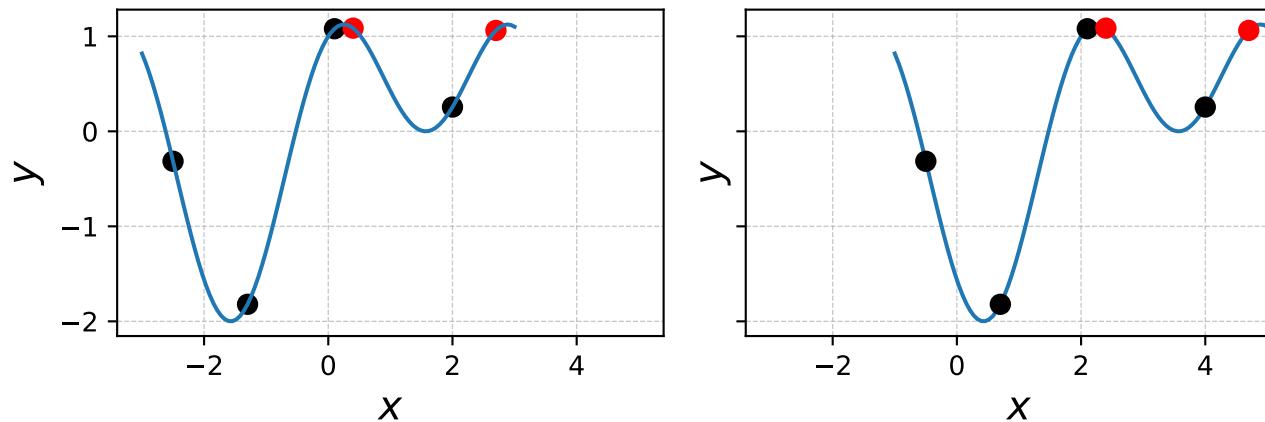
- We process data through a **comparison function**  $g(\mathbf{x}_n, \mathbf{x}_m^*)$ .



Remove all input information that does not matter to impose the desired equivariance.

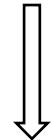
# Relational Encoding

- Translational equivariance  $\rightarrow g_{\text{diff}}(\mathbf{x}_n, \mathbf{x}_m^*) = \mathbf{x}_m^* - \mathbf{x}_n$
- Isotropy (equivariance to rigid transformations)  $\rightarrow g_{\text{dist}}(\mathbf{x}_n, \mathbf{x}_m^*) = \|\mathbf{x}_m^* - \mathbf{x}_n\|_2$



# Simple RCNP

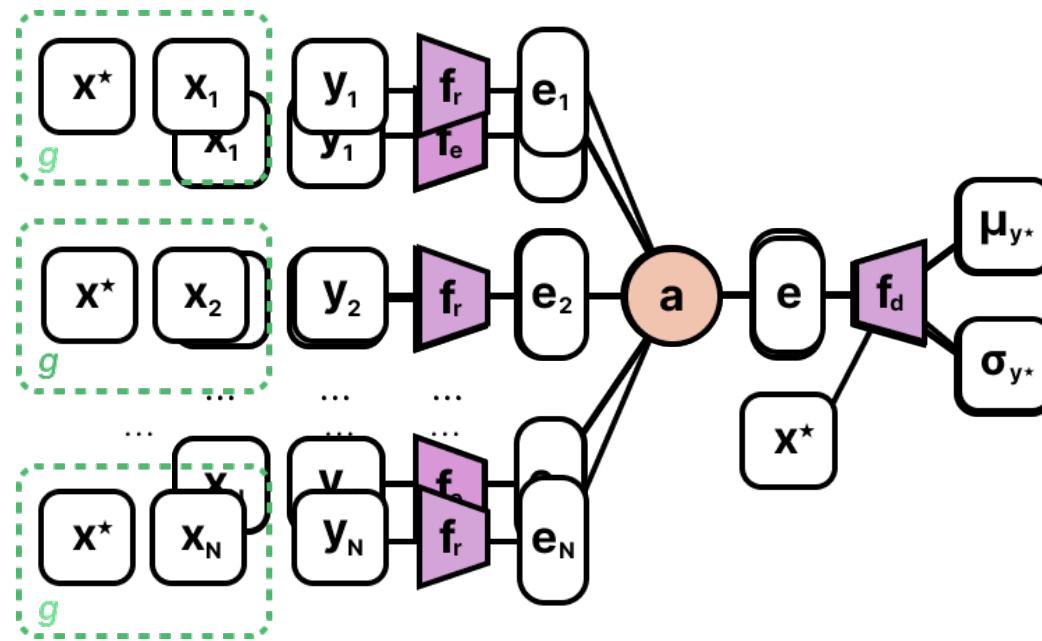
Traditional CNP       $r(\mathbf{x}_m^*, (\mathbf{X}, \mathbf{Y})) = \left( \bigoplus_{n=1}^N \textcolor{violet}{f}_{\textcolor{teal}{e}}(\mathbf{x}_n, \mathbf{y}_n), \mathbf{x}_m^* \right).$



Simple RCNP       $\rho_{\text{diag}}(\mathbf{x}_m^*, (\mathbf{X}, \mathbf{Y})) = \bigoplus_{n=1}^N \textcolor{violet}{f}_{\textcolor{teal}{r}}(\textcolor{blue}{g}(\mathbf{x}_n, \mathbf{x}_m^*), \mathbf{y}_n).$

# Simple RCNP Architecture

If we want to keep the values of the prediction map unchanged (according to the definition of equivariance), we need to ensure that the neural network never receives absolute information.



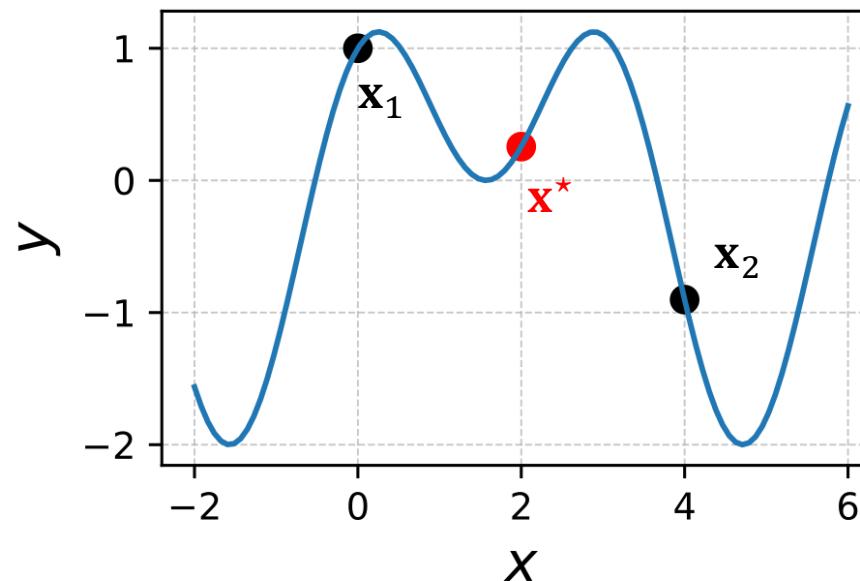
We shift the  $X^*$  from “input of the decoder” to “input of the comparison function”.

# Simple RCNP is not context-preserving for $g_{\text{dist}}$

Relative information between  $x_1$  and  $\mathbf{x}^*$ :  $g_{\text{dist}}(x_1, \mathbf{x}^*) = 2$

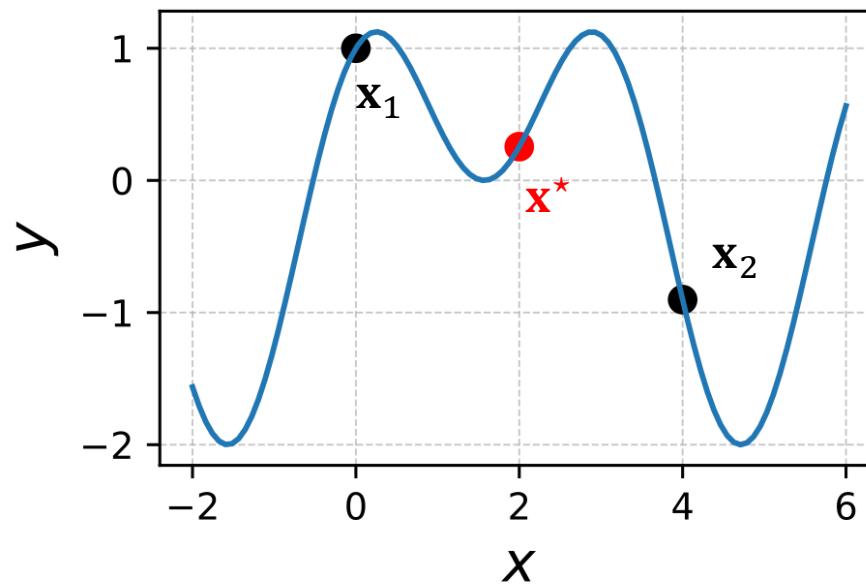
Relative information between  $x_2$  and  $\mathbf{x}^*$ :  $g_{\text{dist}}(x_2, \mathbf{x}^*) = 2$

**But  $x_1 \neq x_2$ !**



# Simple RCNP is not context-preserving for $g_{\text{dist}}$

We should also encode the relative information between  $\mathbf{x}_1$  and  $\mathbf{x}_2$ !



# Full RCNP is context-preserving for $g_{\text{dist}}$

- The full relational encoding of a target point  $\mathbf{x}_m^*$  respect to the context set  $(\mathbf{X}, \mathbf{Y})$ :

$$\rho_{\text{full}}(\mathbf{x}_m^*, (\mathbf{X}, \mathbf{Y})) = \bigoplus_{n,n'=1}^N \textcolor{violet}{f}_r(\textcolor{green}{g}(\mathbf{x}_n, \mathbf{x}_m^*), \mathbf{R}_{nn'}) , \quad \mathbf{R}_{nn'} \equiv (\textcolor{green}{g}(\mathbf{x}_n, \mathbf{x}_{n'}), \mathbf{y}_n, \mathbf{y}_{n'}) ,$$

$\mathbf{R}$  is the relational matrix, comparing all pairs of the context set.

- The simple relational encoding is a diagonal version of the full relational encoding:

$$\rho_{\text{diag}}(\mathbf{x}_m^*, (\mathbf{X}, \mathbf{Y})) = \bigoplus_{n=1}^N \textcolor{violet}{f}_r(\textcolor{green}{g}(\mathbf{x}_n, \mathbf{x}_m^*), \textcolor{green}{g}(\mathbf{x}_n, \mathbf{x}_n), \mathbf{y}_n) = \bigoplus_{n=1}^N \textcolor{violet}{f}_r(\textcolor{green}{g}(\mathbf{x}_n, \mathbf{x}_m^*), \mathbf{y}_n) .$$

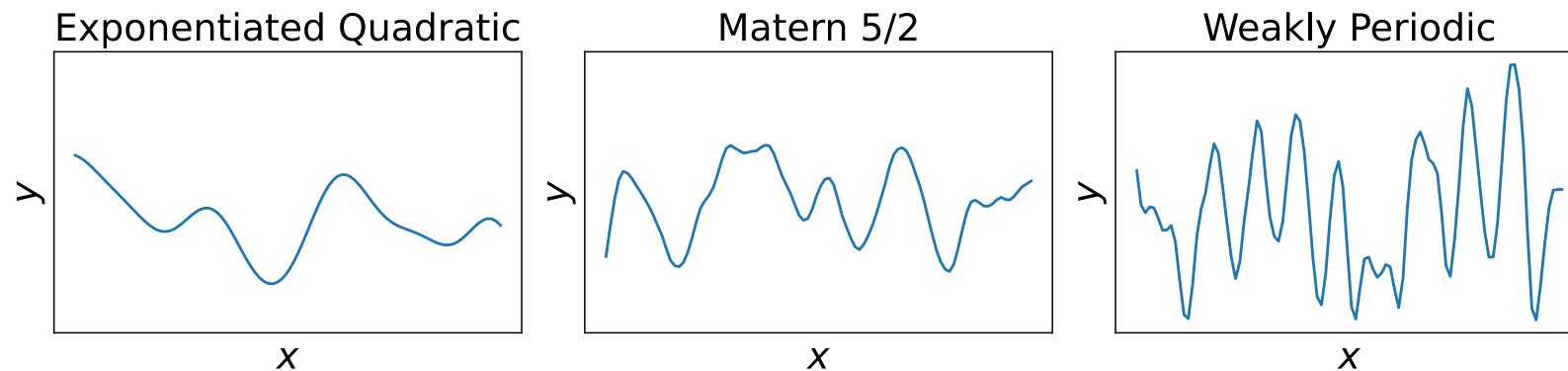
# Experiments

- **Synthetic regression**
- **Bayesian optimization**
- Lotka–Volterra model
- Reaction-diffusion model
- Image completion

# Experiments

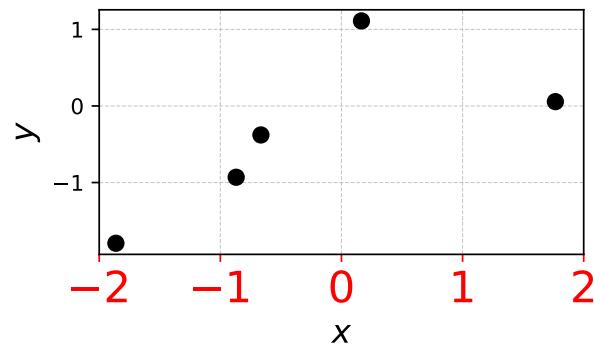
## Synthetic regression

- RCNPs vs CNP models across different synthetic regression tasks.

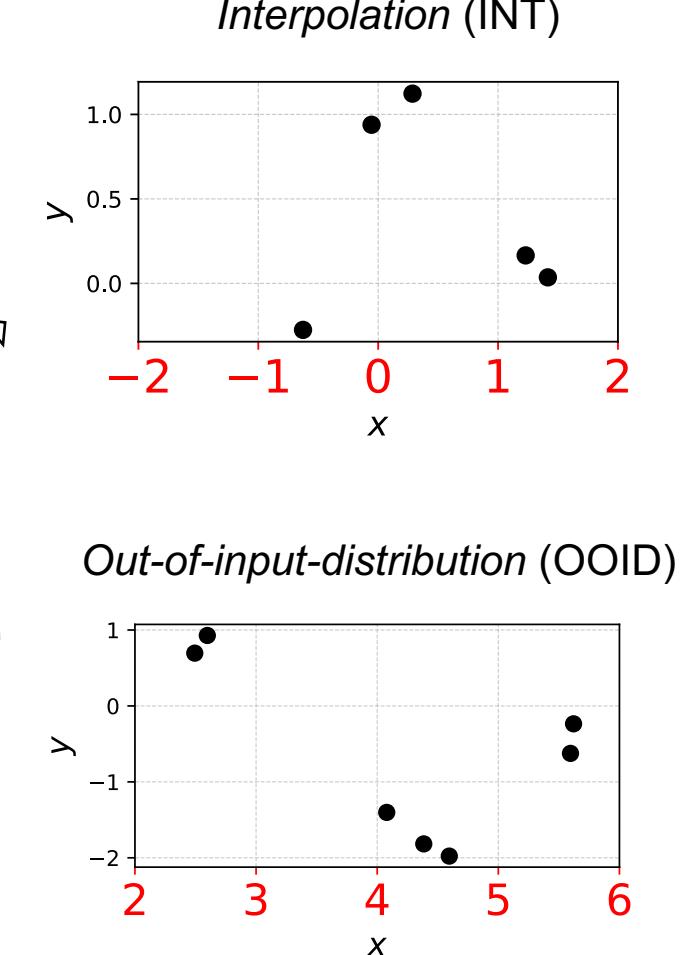


# Experiments

## Synthetic regression



training

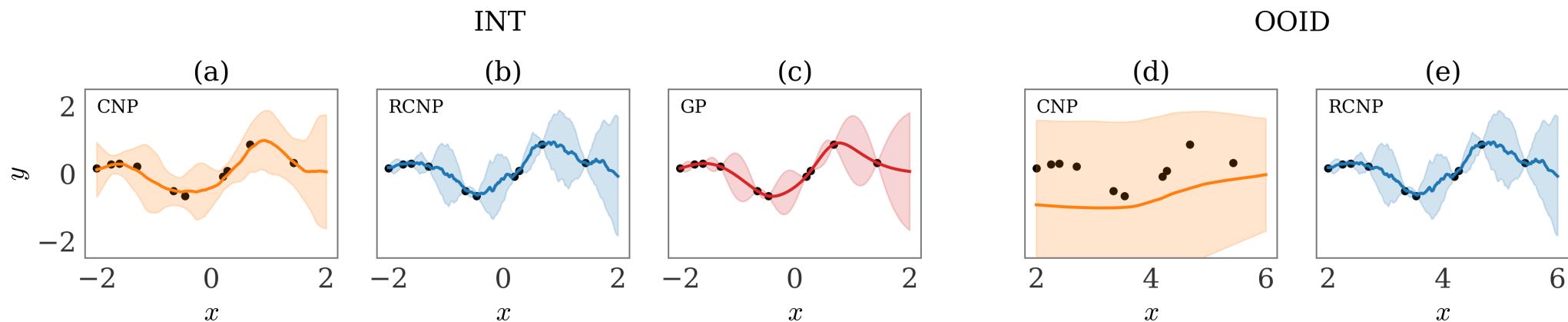


evaluation

# Experiments

## Synthetic regression

- INT: The CNP **underfits** the context data, while the RCNP **yields better predictions**.
- OOID: The CNP **fails** to make prediction; whereas the RCNP generalizes by means of translational equivariance.



# Experiments

## Synthetic regression

- The OOID results show significant improvement of our models, as they can leverage translational equivariance to generalize outside the training range.
- ConvCNPs excel in low dimensions, but RCNPs are often competitive and are applicable for  $d_x \gg 2$ .

		Weakly-periodic KL divergence( $\downarrow$ )			Sawtooth log-likelihood( $\uparrow$ )			Mixture log-likelihood( $\uparrow$ )		
		$d_x = 1$	$d_x = 3$	$d_x = 5$	$d_x = 1$	$d_x = 3$	$d_x = 5$	$d_x = 1$	$d_x = 3$	$d_x = 5$
INT	RCNP (sta)	0.24 (0.00)	0.28 (0.00)	0.31 (0.00)	3.03 (0.06)	0.85 (0.01)	0.44 (0.00)	0.20 (0.01)	-0.10 (0.00)	-0.31 (0.03)
	RGNP (sta)	0.03 (0.00)	<b>0.05</b> (0.00)	<b>0.08</b> (0.00)	3.90 (0.09)	<b>1.09</b> (0.01)	<b>1.13</b> (0.05)	0.34 (0.03)	<b>0.37</b> (0.01)	<b>0.04</b> (0.02)
	ConvCNP	0.21 (0.00)	-	-	3.64 (0.04)	-	-	0.38 (0.02)	-	-
	ConvGNP	<b>0.01</b> (0.00)	-	-	<b>3.94</b> (0.11)	-	-	<b>0.49</b> (0.15)	-	-
	CNP	0.31 (0.00)	0.39 (0.00)	0.42 (0.00)	2.25 (0.02)	0.36 (0.28)	-0.03 (0.10)	0.01 (0.01)	-0.57 (0.11)	-0.72 (0.08)
	GNP	0.06 (0.00)	0.08 (0.01)	0.11 (0.01)	0.83 (0.04)	0.23 (0.13)	0.02 (0.05)	0.17 (0.01)	-0.17 (0.00)	-0.32 (0.00)
OOD	RCNP (sta)	0.24 (0.00)	<b>0.28</b> (0.01)	<b>0.31</b> (0.00)	3.04 (0.06)	0.85 (0.01)	0.44 (0.00)	0.20 (0.01)	-0.10 (0.00)	-0.31 (0.03)
	RGNP (sta)	0.03 (0.00)	<b>0.05</b> (0.01)	<b>0.08</b> (0.00)	3.90 (0.10)	<b>1.09</b> (0.01)	<b>1.13</b> (0.05)	0.34 (0.03)	<b>0.37</b> (0.01)	<b>0.04</b> (0.02)
	ConvCNP	0.21 (0.00)	-	-	3.64 (0.04)	-	-	0.38 (0.02)	-	-
	ConvGNP	<b>0.01</b> (0.00)	-	-	<b>3.97</b> (0.08)	-	-	<b>0.49</b> (0.15)	-	-
	CNP	2.88 (0.91)	1.58 (0.50)	2.20 (0.81)	F	-0.37 (0.12)	-0.22 (0.03)	F	-2.55 (1.15)	-1.71 (0.55)
	GNP	F	1.47 (0.27)	0.62 (0.04)	F	F	F	F	-0.67 (0.05)	-0.72 (0.03)

# Experiments

## Bayesian optimization

The goal of **Bayesian optimization (BO)** is to find the global optimum  $x^*$  of  $f$  under a limited budget, defined as:

$$x^* = \arg \min_{x \in \mathcal{X}} f(x)$$

BO is equipped with two components:

- A surrogate model for  $f$ .
- An acquisition function  $\alpha$ , which is based on the surrogate.

During optimization, we conduct the following operations iteratively:

- Query the next point by maximizing  $\alpha$ .
- Augment the dataset with the newly added sample.
- Update the surrogate model with the new dataset.

# Experiments

## Bayesian optimization

- We train CNP models with synthetic data.
- The trained CNP models are used as **surrogate models** to minimize the Hartmann function.

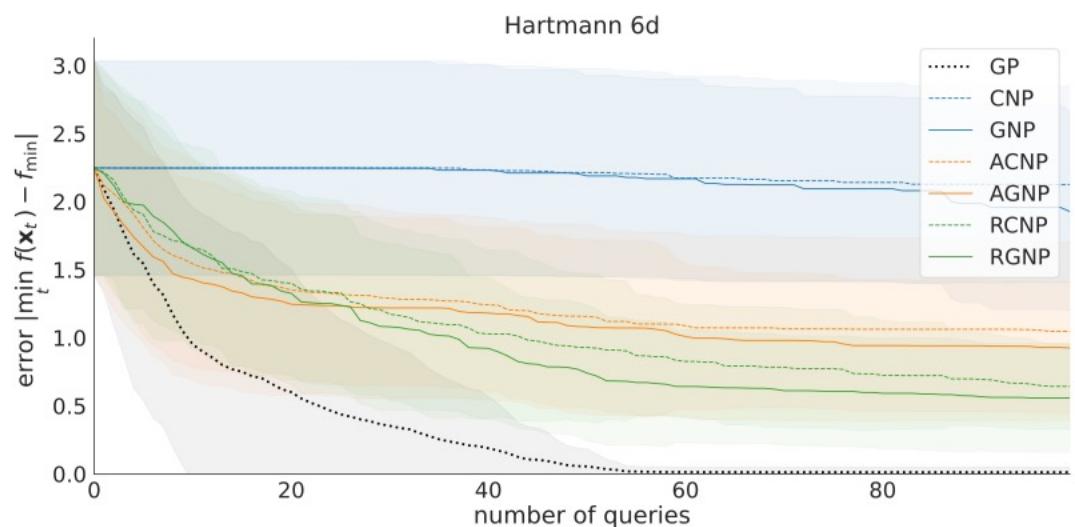
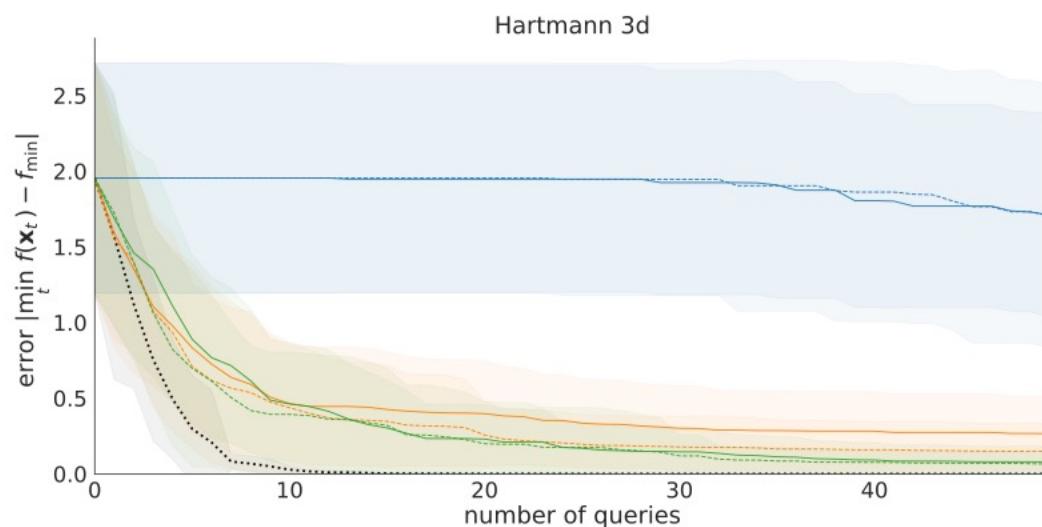
- Query the next point by maximizing  $\alpha$ .
- Augment the dataset with the newly added sample.
- Update the surrogate model with the new dataset.

- Query the next point by maximizing  $\alpha$ .
- Augment the context set with the newly added sample.
- ~~Update the surrogate model with the new dataset.~~

# Experiments

## Bayesian optimization

- CNPs and GNPs fail at the task.
- RCNPs can come close to the performance of a Gaussian Process.



# Experiments

## Bayesian optimization

### End-to-End Bayesian Optimization<sup>9</sup>

- ~~Query the next point by maximizing  $\alpha$ .~~
- Augment the context set with the newly added sample.
- ~~Update the surrogate model with the new dataset.~~

<sup>9</sup> Maraval, A., Zimmer, M., Grosnit, A., & Ammar, H. B. (2023). End-to-End Meta-Bayesian Optimisation with Transformer Neural Processes. *arXiv preprint arXiv:2305.15930*.

# Limitations

- The RCNP models have more expensive cost for training and deployment than basic CNPs.
  - We need to calculate the relative information for  $M$  test points with respect to  $N$  context points.
  - CNPs:  $O(M + N)$ , Simple RCNP:  $O(MN)$ , Full RCNP:  $O(MN^2)$ .

# Takeaways

- Exploiting equivariances intrinsic to a problem can significantly improve performance.
- RCNPs provide a simple and effective way to implement equivariances into the CNP model family.

# Thanks!



Paper: <https://arxiv.org/abs/2306.10915>.

Paper

Code implementation: <https://github.com/acerbilab relational-neural-processes>.

Code

FCAI Team Amortized Inference:



Daolang  
Huang



Manuel  
Haussmann



Ulpu Remes



Ti John



Grégoire  
Clarté



Kevin S  
Luck



Samuel Kaski



Luigi Acerbi