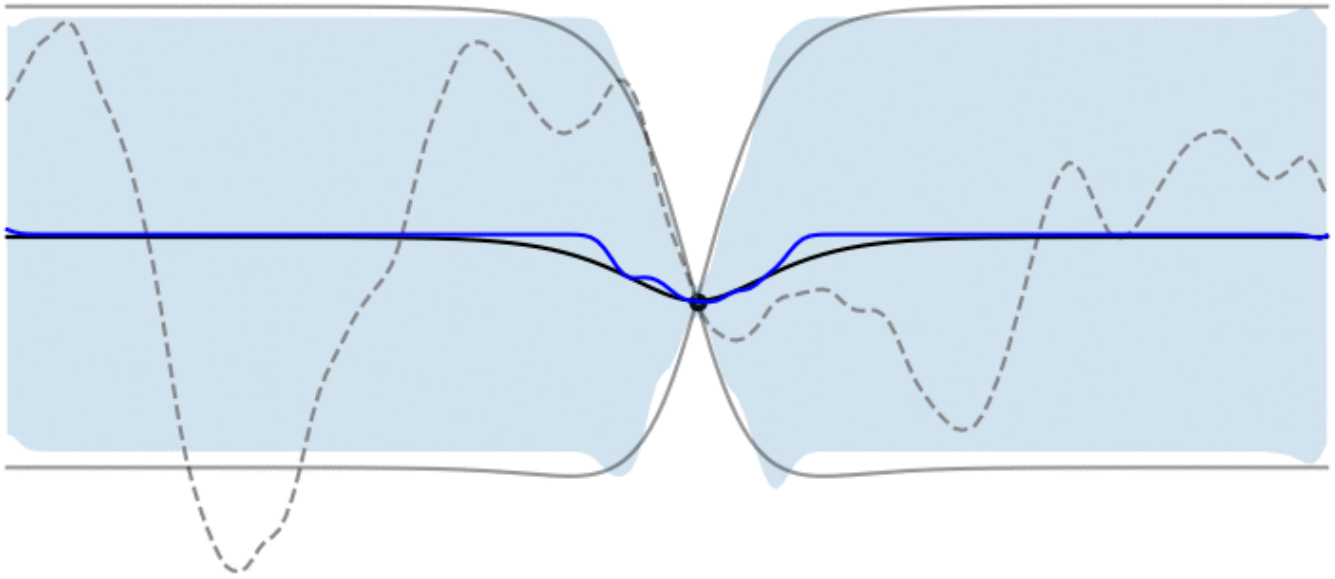
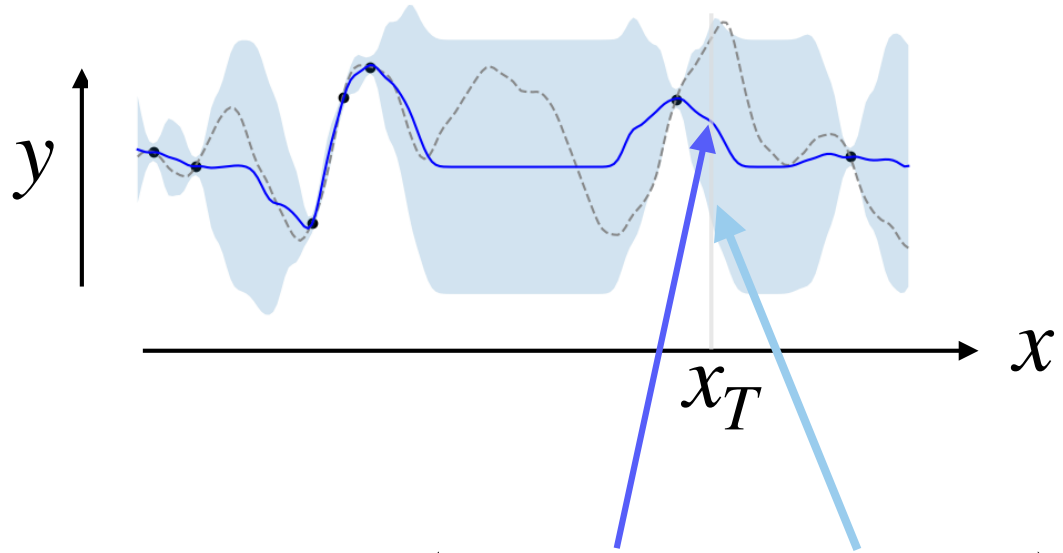


Neural Processes for Data Efficient Machine Learning

Prof. Richard E. Turner
University of Cambridge

Standard approach to regression



training data

$D_c = \{x_n, y_n\}_{n=1}^N$

$$p(y_T | x_T, \theta) = \mathcal{N}\left(y_T;\, \mu_\theta(x_T), \sigma_\theta^2(x_T)\right)$$

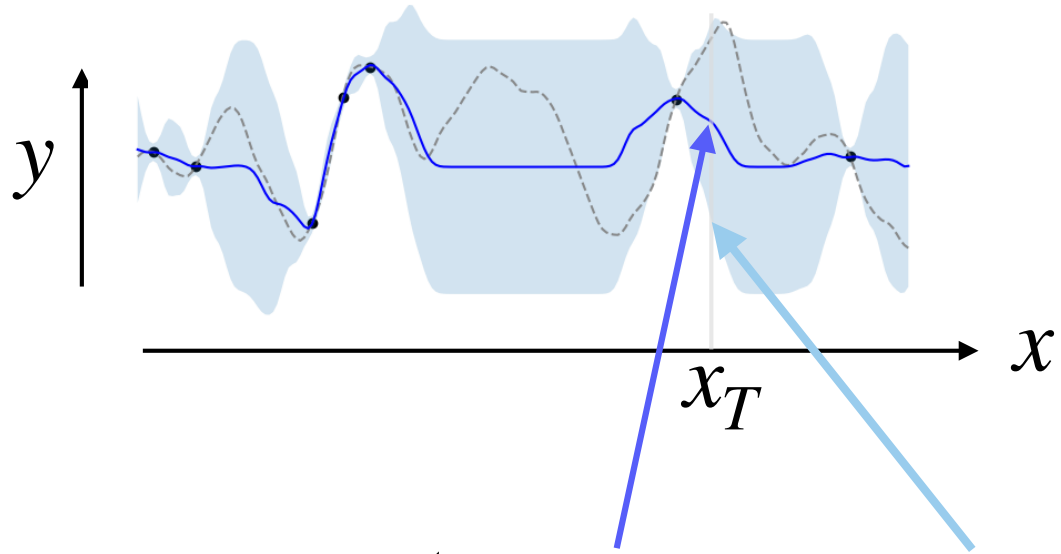
fit parameters θ to the training data set $D_c = \{x_n, y_n\}_{n=1}^N$

e.g. via maximum-likelihood

$$\theta^* = \underset{\theta \in \Theta}{\textbf{arg max}} \sum_{n=1}^{N} \log p(y_n | x_n, \theta)$$

after training, parameters depend on the data: $\theta(D_c) \Longrightarrow \mu(x_T, D_c)$, $\sigma^2(x_T, D_c)$

Conditional Neural Processes



training data
$D_c = \{x_n, y_n\}_{n=1}^N$

Zaheer et al.
Deep Sets. 2017

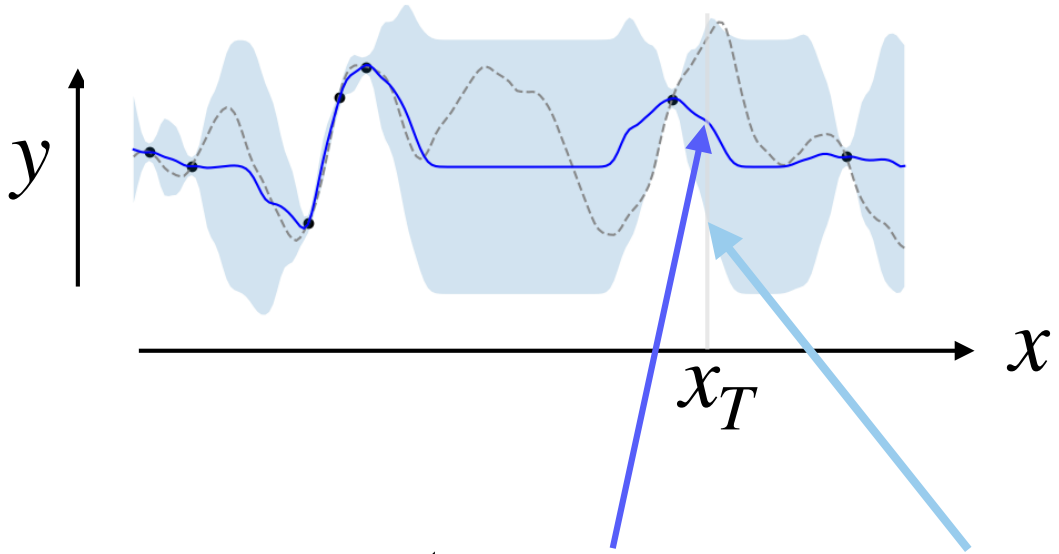
$$p(y_T | x_T, \theta) = \mathcal{N}\left(y_T;\ \mu_\theta(x_T, D_c),\ \sigma_\theta^2(x_T, D_c)\right)$$

Neural Processes model the mapping from datasets to parameters **directly** using a neural network

Theorem. A function operating on a set $f(S)$ is a valid set function iff it can be decomposed in the form:

$$\rho\left(\sum_{s \in S} \phi(s)\right)$$

Conditional Neural Processes



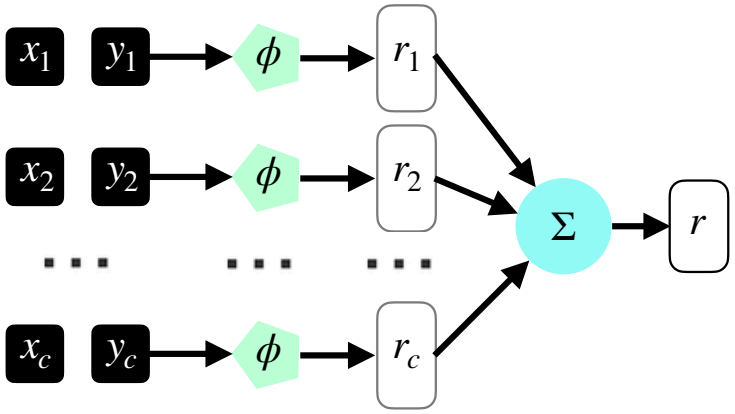
training data

$$D_c = \{x_n, y_n\}_{n=1}^{N}$$

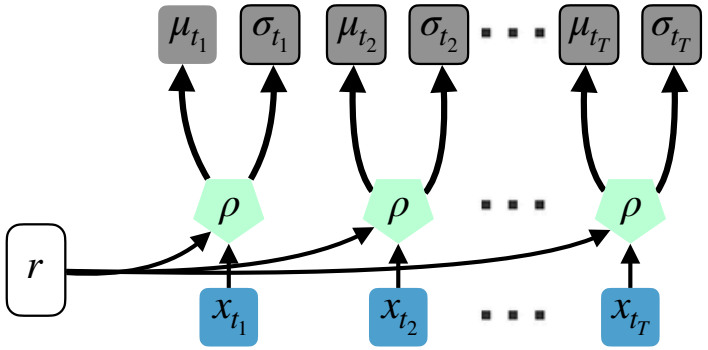
Zaheer et al.
Deep Sets. 2017

$$p(y_T \mid x_T, \theta) = \mathcal{N}\left(y_T;\ \mu_\theta(x_T, D_c),\ \sigma_\theta^2(x_T, D_c)\right)$$

Set Encoder $E(D_c) = r$

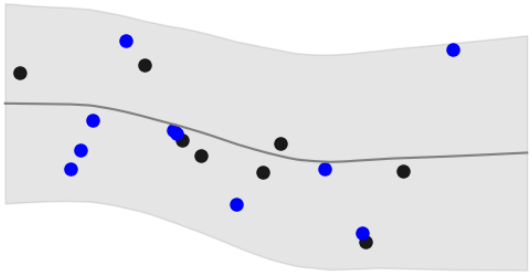


Decoder $p(y \mid x, r)$



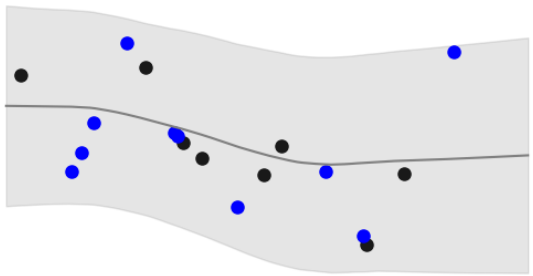
Training CNPs: maximum likelihood is meta-learning

$$\theta^* = \underset{\theta \in \Theta}{\textbf{arg max}} \; \mathop{\mathbb{E}}_{(D_c,\, D_t) \sim P} \left[\log p(D_t \,|\, D_c, \theta) \right]$$

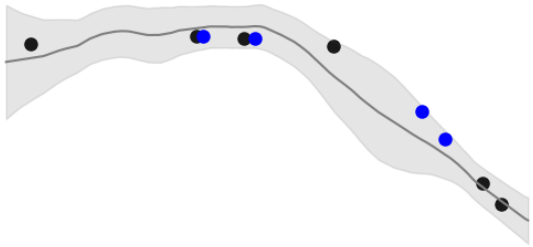


$$\theta^* = \underset{\theta \in \Theta}{\textbf{arg max}} \; \mathop{\mathbb{E}}_{(D_c,\, D_t) \sim P} \left[\sum_{(x,y) \in D_t} \log \mathcal{N}\left(y; \mu_\theta(x, D_c), \sigma^2_\theta(x, D_c)\right) \right]$$

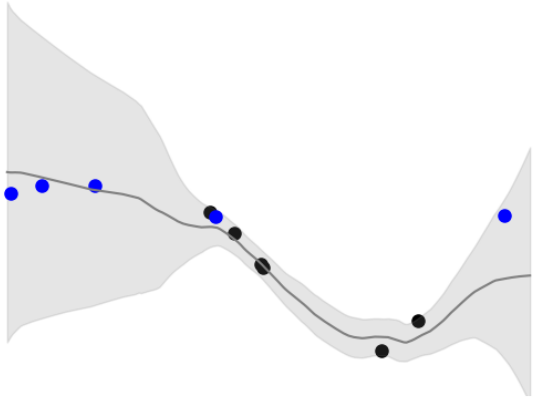
Epoch 0



Epoch 40

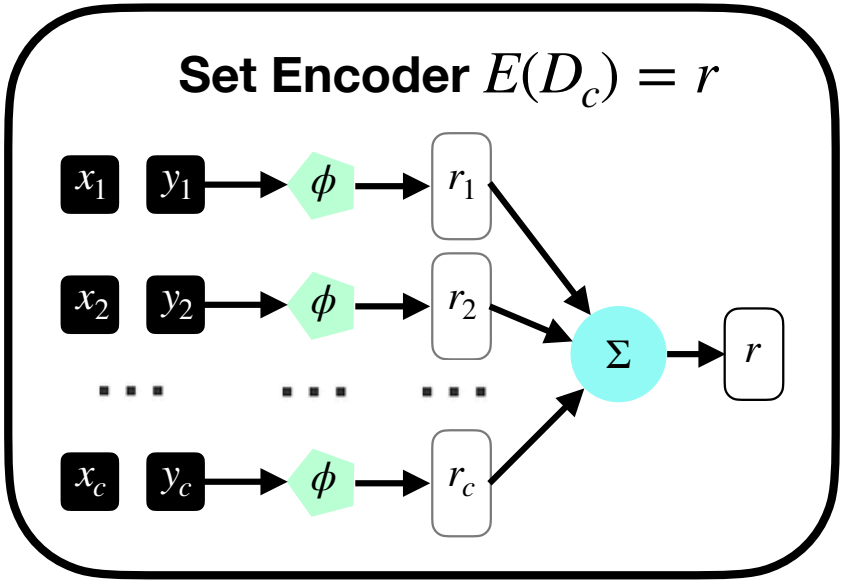


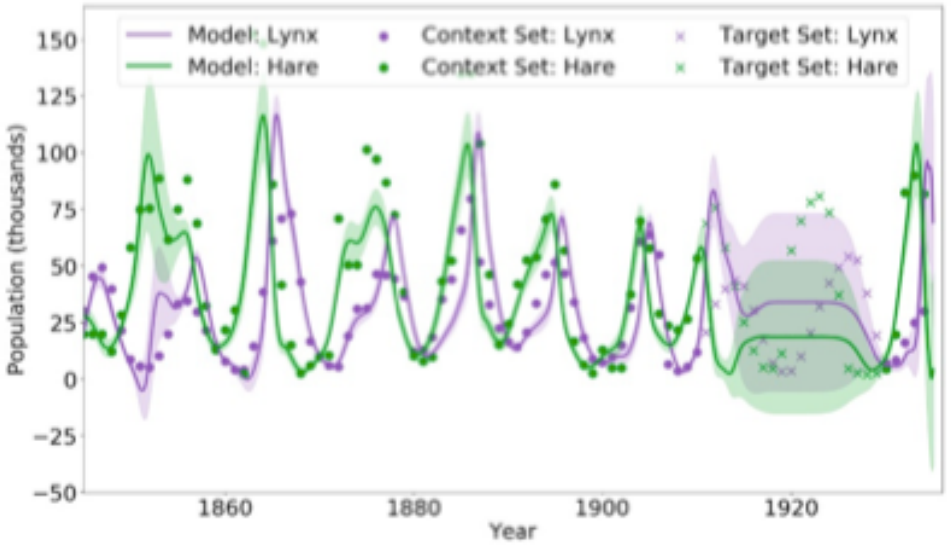
Epoch 80



Uses of Neural Processes

- Sets of datasets with
  - small size (where avoiding overfitting important)
  - variable and complex patterns of missing data
  - irregularly sampled spatio-temporal data
- Continual learning (naturally supports incremental updates)
- Sim2Real (train on data from simulator; deploy on real data)
- On device learning (does not require derivatives)

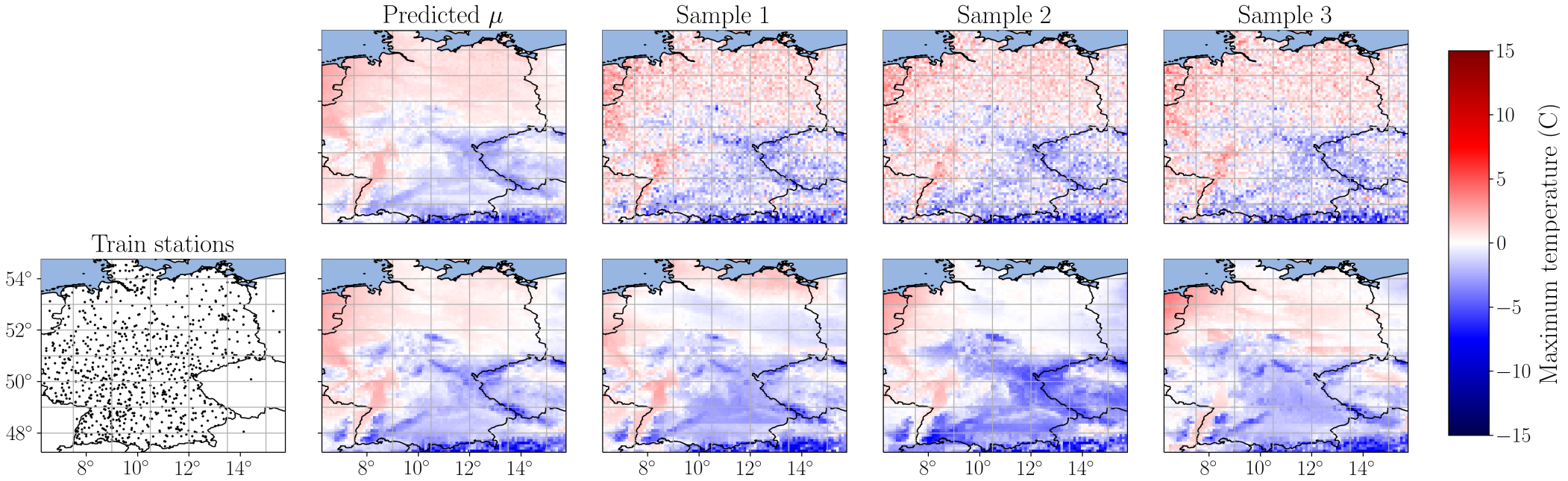




Drawbacks of current CNPs

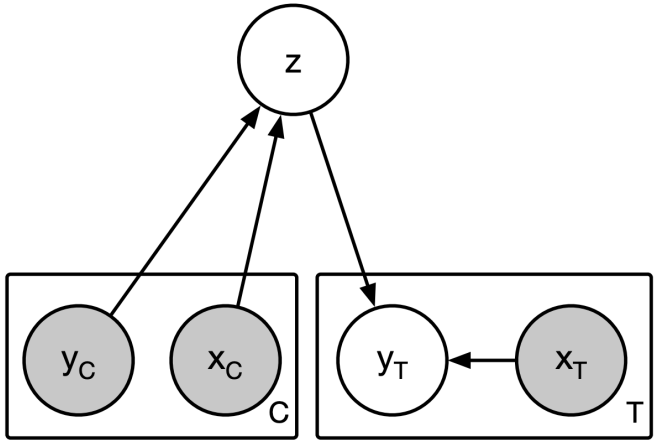
Factorised predictive distribution

$$p(y_T | x_T, \theta) = \mathcal{N}\left(y_T;\ \mu_\theta(x_T, D_c),\ \sigma^2_\theta(x_T, D_c)\right)$$



Extension: Latent Variable Model

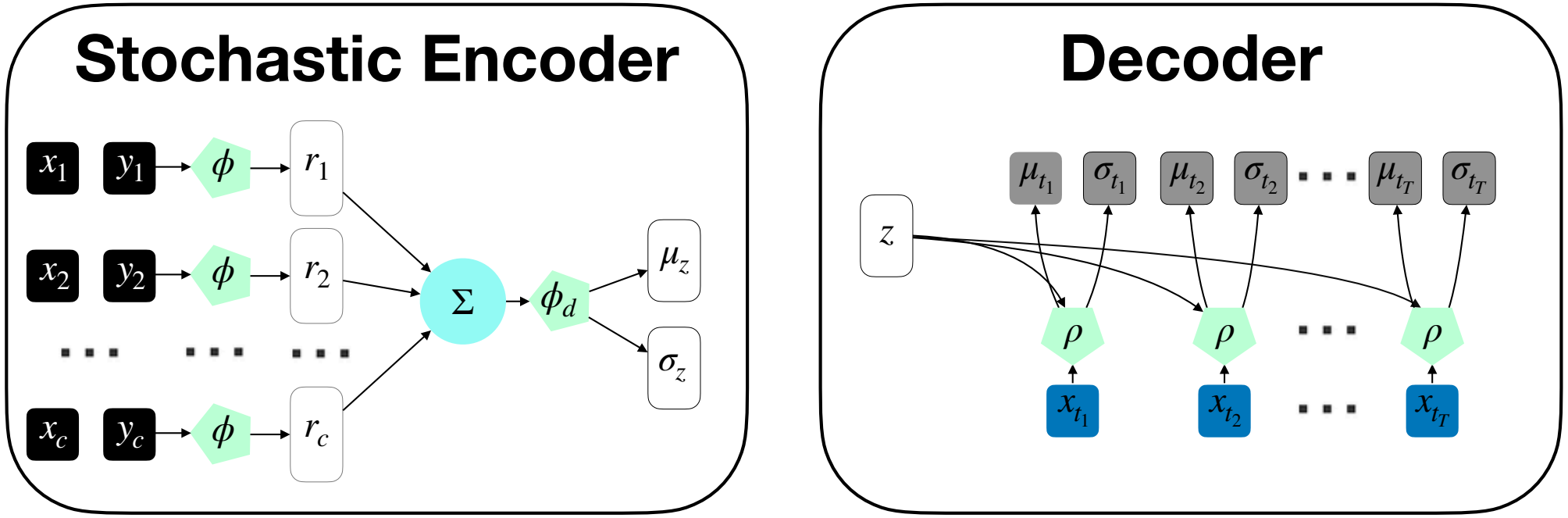
Neural Processes



$$p(y_{1:T} \mid x_{1:T}, D_c) = \int p(z \mid D_c) \prod_{t=1}^{T} p(y_t \mid x_t, z) \mathrm{d}z$$

Neural Processes Computational Graph

$$p(y_{1:T}|x_{1:T}, D_c) = \int p(z|D_c) \prod_{t=1}^{T} p(y_t|x_t, z) \mathrm{d}z$$



Variational Training Objectives

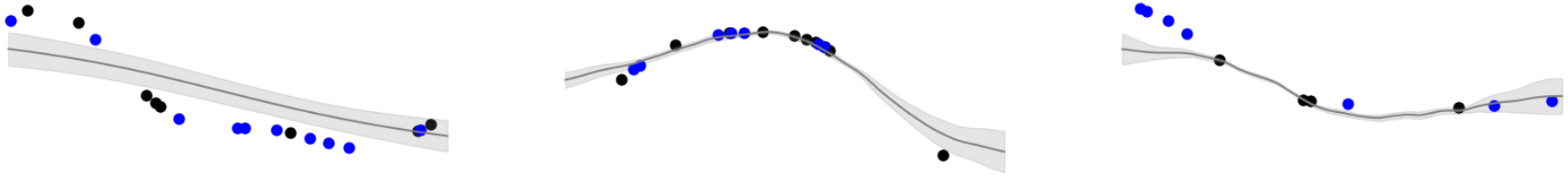
$\underbrace{D_c = \{(x_i, y_i)\}_{i=1}^{m}}_{\textbf{Context set}};$ $\underbrace{D_t = \{(x_i, y_i)\}_{i=m+1}^{T}}_{\textbf{Target set}};$ $\underbrace{D = D_c \cup D_t}_{\textbf{Context and Target}}$

$$\log p(y_{m+1:T} \,|\, x_{m+1:T}, D_c) \quad \geq \mathop{\mathbb{E}}_{z \sim q(z|D)} \left[\sum_{i=m+1}^{T} \log p(y_i \,|\, x_i, z) - \log \frac{q(z \,|\, D)}{p(z \,|\, D_c)} \right]$$

Meta-Training

$$\theta, \psi = \textbf{arg max} \underset{z \sim q_\psi(z|D)}{\mathbb{E}} \left[\sum_{i=m+1}^{N} \log p(y_i \,|\, x_i, z) - \log \frac{q_\psi(z \,|\, D)}{q_\psi(z \,|\, D_c)} \right]$$

Epoch 0 Epoch 40 Epoch 80

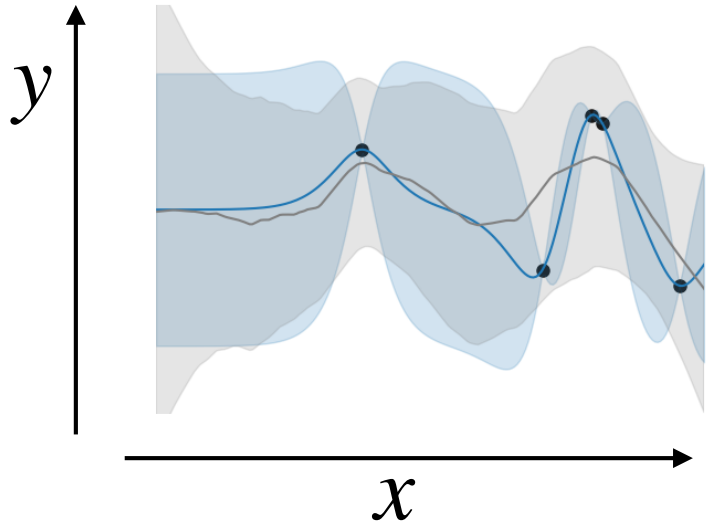


Sampling from NPs

Drawbacks of current CNPs

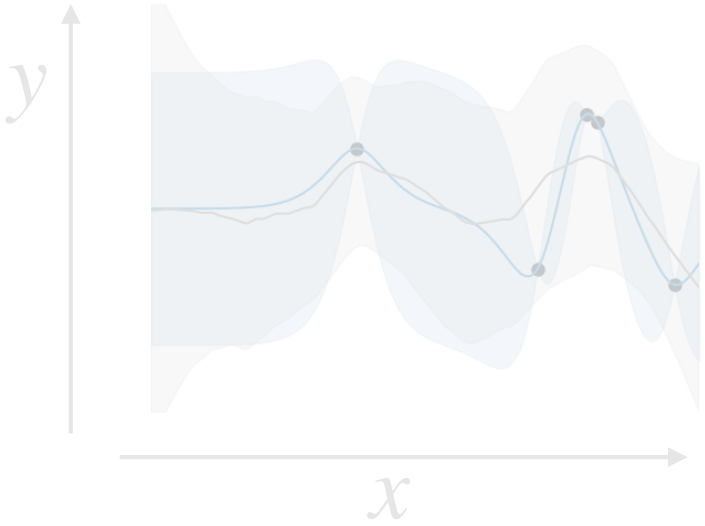
Severe under-fitting

Failure to Extrapolate

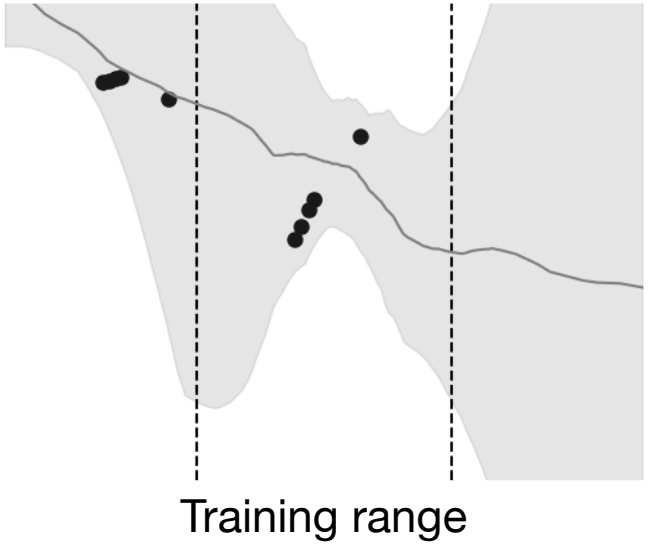


Drawbacks of current CNPs

Severe under-fitting

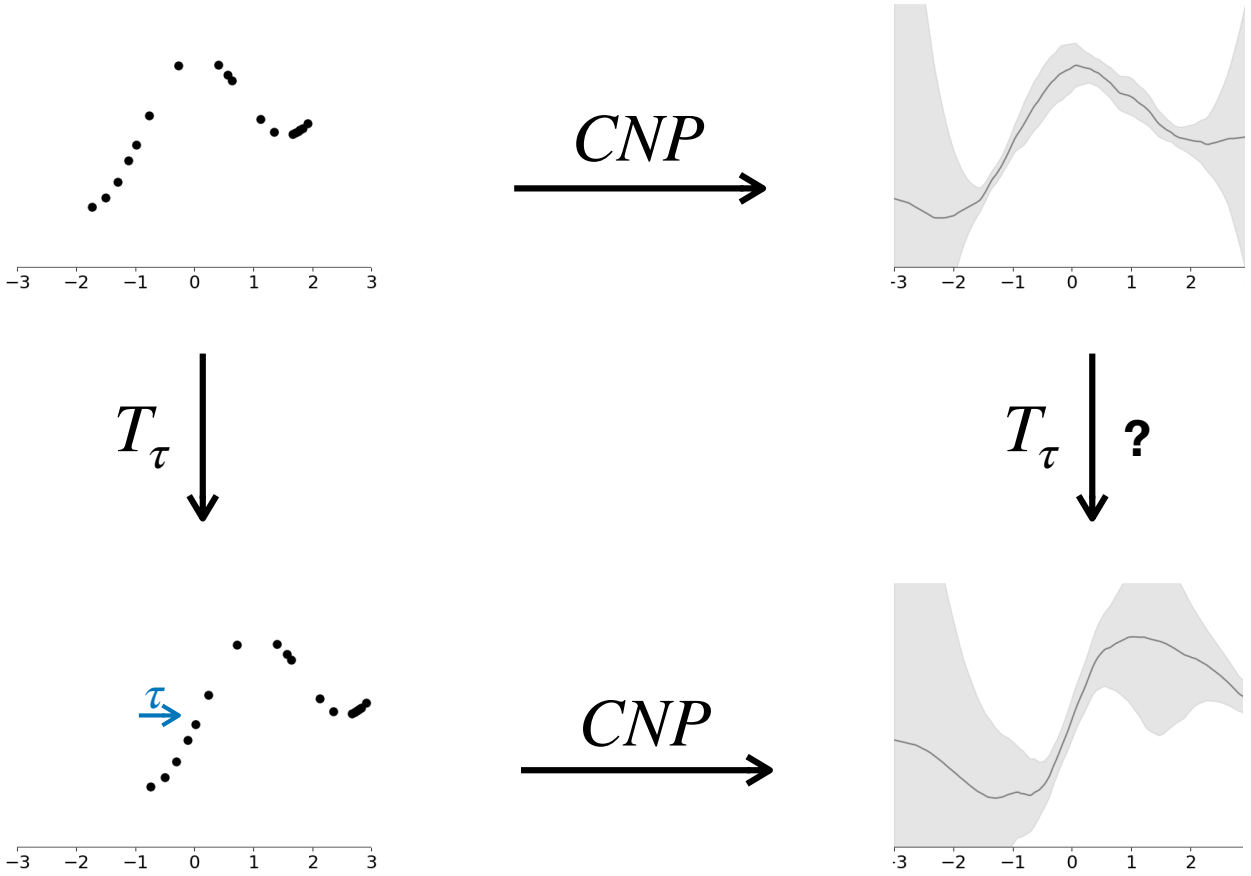


Failure to Extrapolate

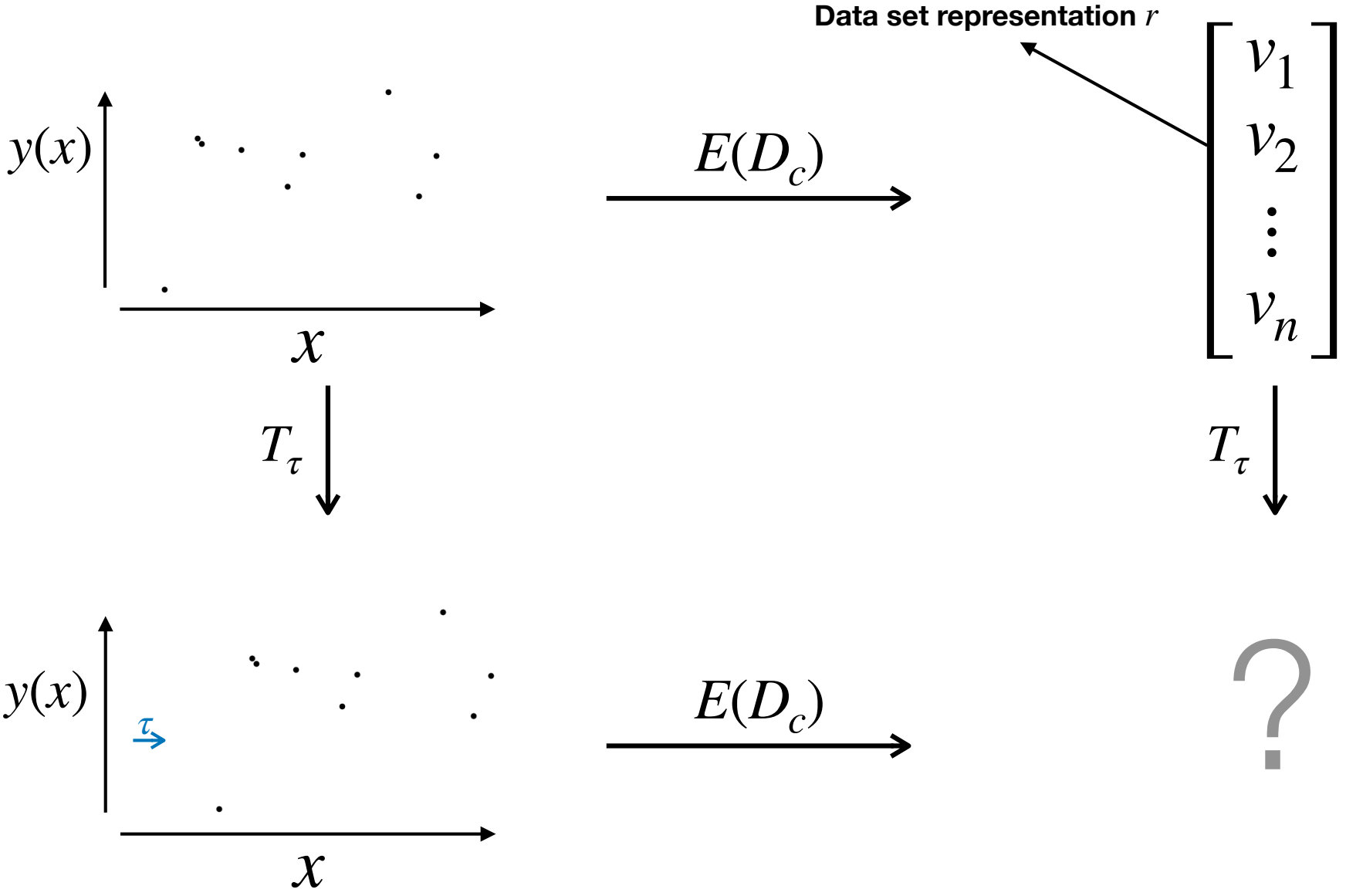


Solution: Translation Equivariant CNPs

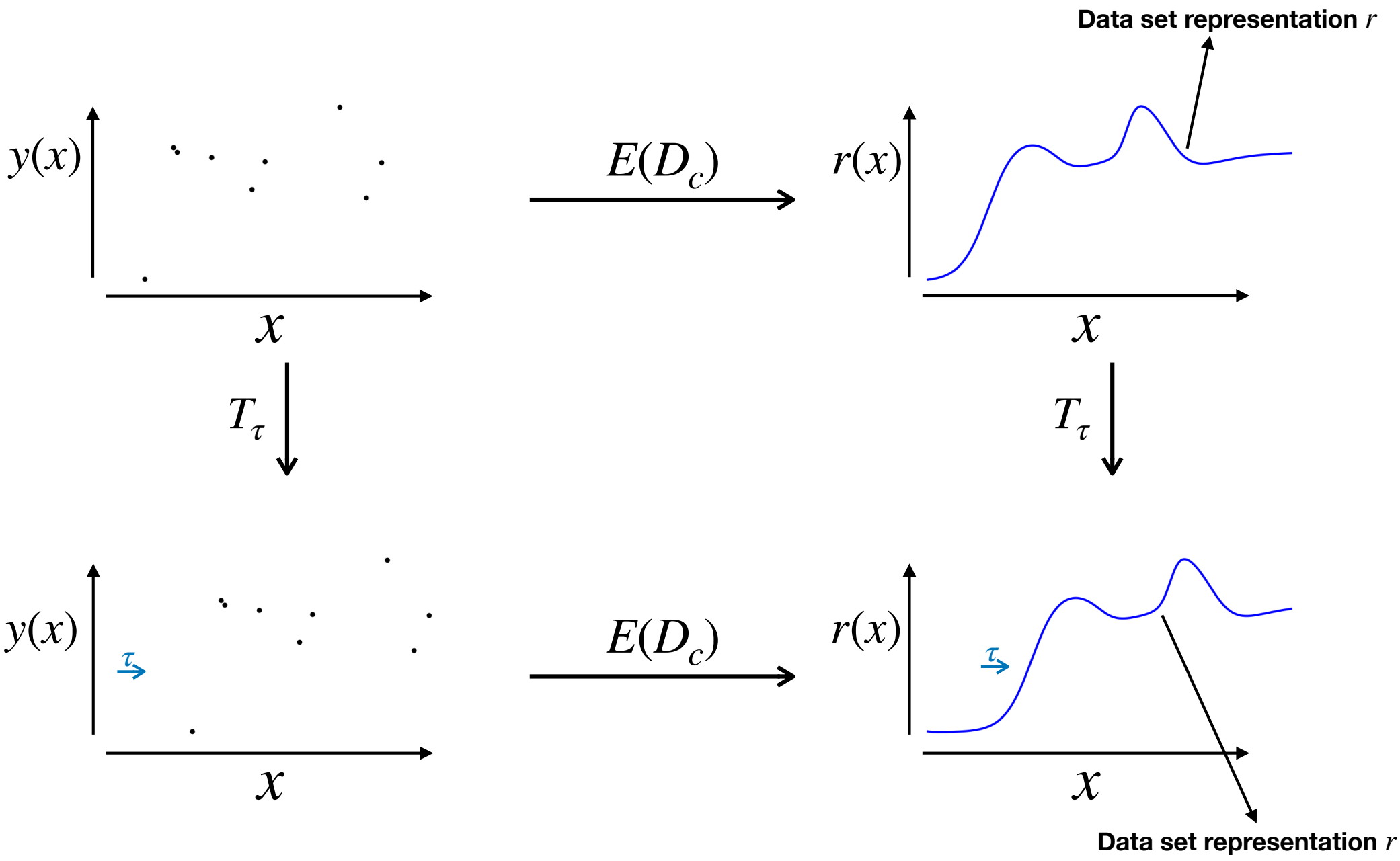
CNPs and Translation Equivariance



Embedding Sets into Function Space



Embedding Sets into Function Space



Main Result (Informal Statement)

Deep Sets

$$f(D_c) = \mathrm{MLP}\Big(E\Big(\underbrace{\{(x_n, y_n)\}_{n=1}^{N}}_{D_c}\Big)\Big)$$

$$E(D_c) = \sum_{(x,y)\in D_c} \phi_{xy}([x; y])$$

$\Longleftrightarrow$

$f(D_c)$ is **permutation invariant**

Convolutional Deep Sets

$$f(D_c) = \mathrm{CNN}\Big(E\Big(\underbrace{\{(x_n, y_n)\}_{n=1}^{N}}_{D_c}\Big)\Big)$$

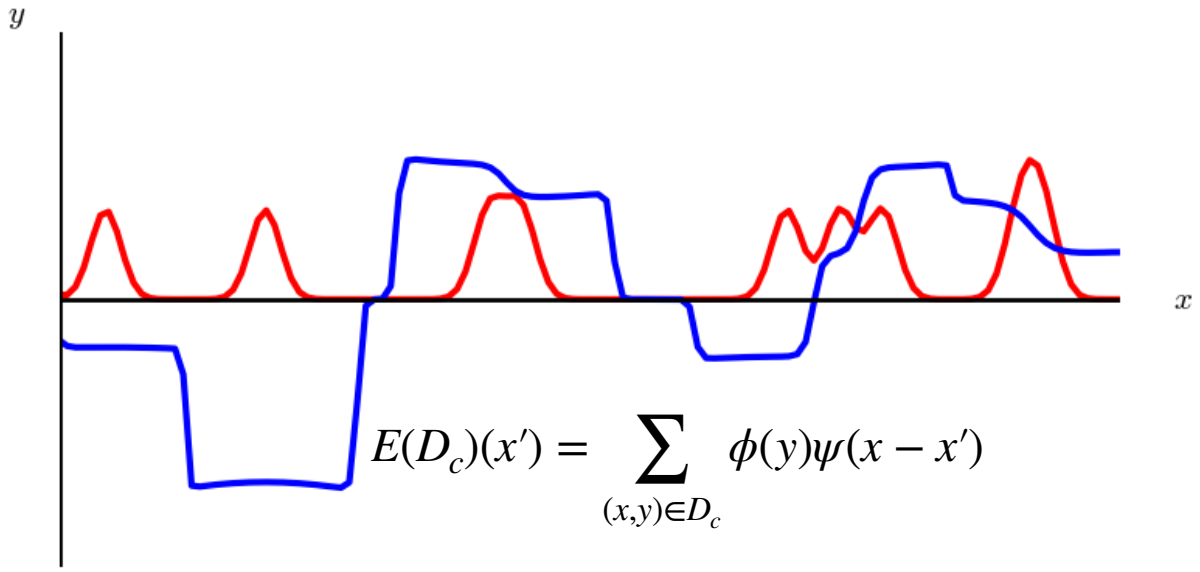
$$E(D_c)(x') = \sum_{(x,y)\in D_c} \phi_y(y)\psi(x - x')$$

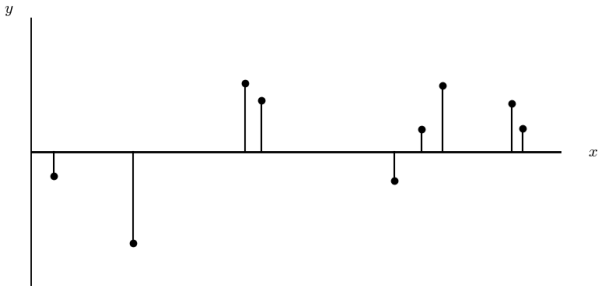
$\Longleftrightarrow$

$f(D_c)$ is **permutation invariant** and **translation equivariant**

Convolutional Conditional Neural Process

$$ConvCNP = CNP + TE$$

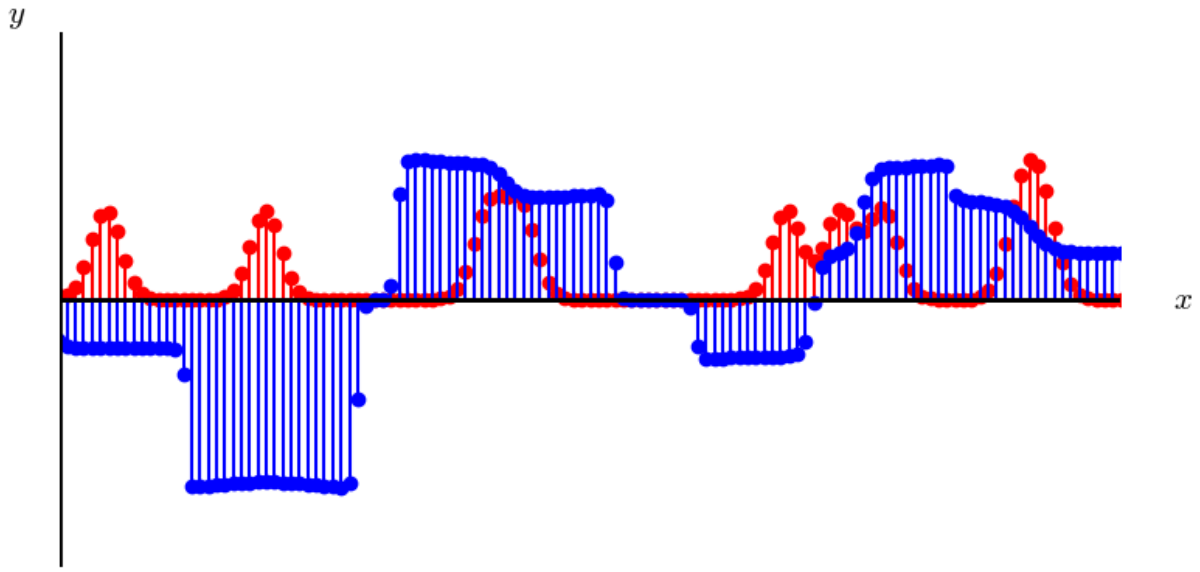


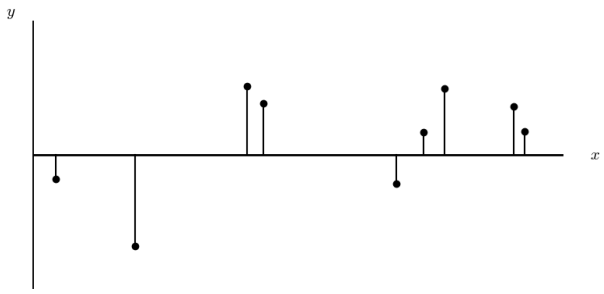


Context set (D_c)

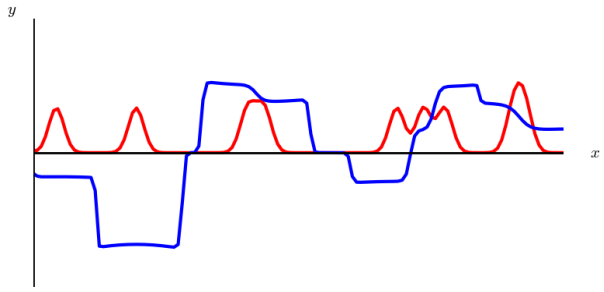
Convolutional Conditional Neural Process

$$ConvCNP = CNP + TE$$





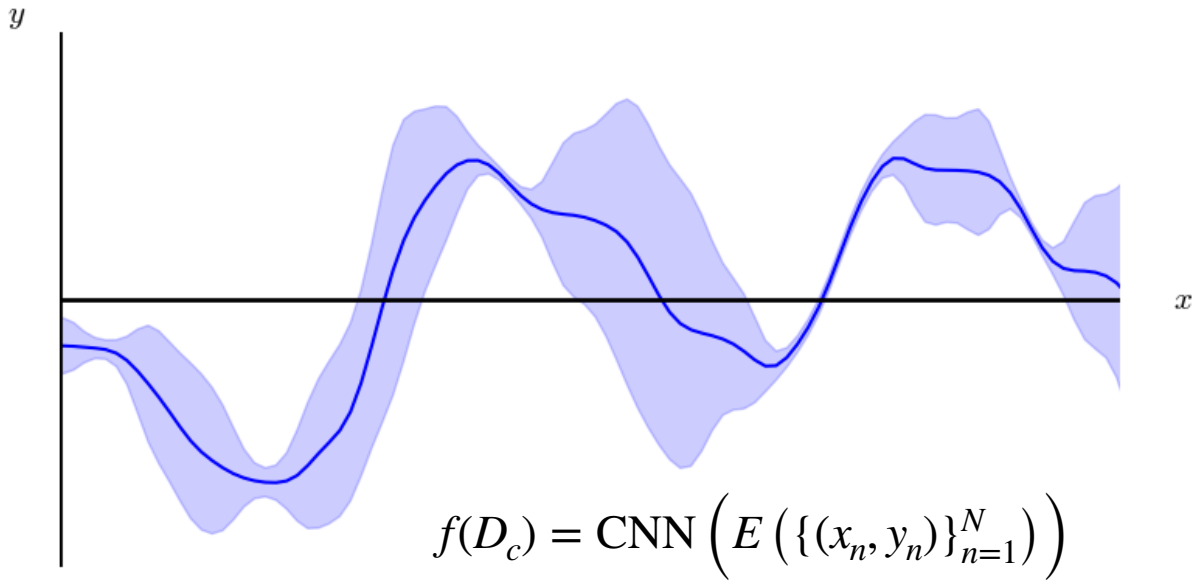
Context set (D_c)



Embedding (r)

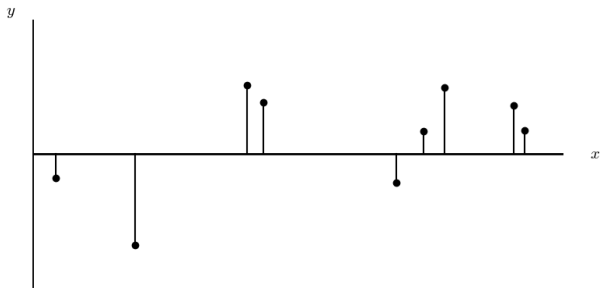
Convolutional Conditional Neural Process

$$ConvCNP = CNP + TE$$

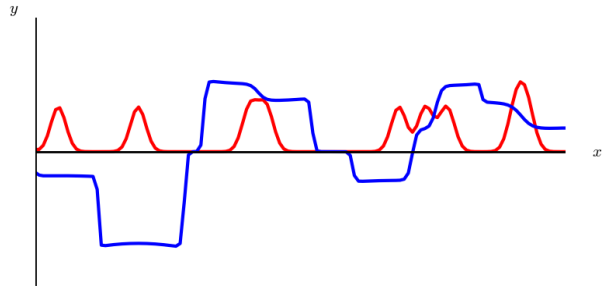


$$f(D_c) = \text{CNN}\left(E\left(\{(x_n, y_n)\}_{n=1}^{N}\right)\right)$$

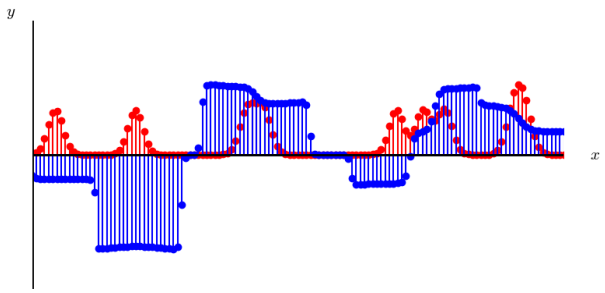
Predictive ($p(y_T | x_T, D_c)$)



Context set (D_c)

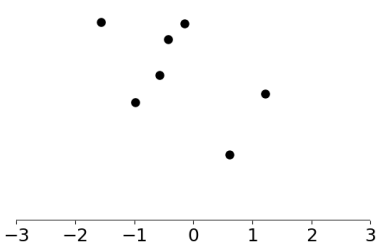


Embedding (r)

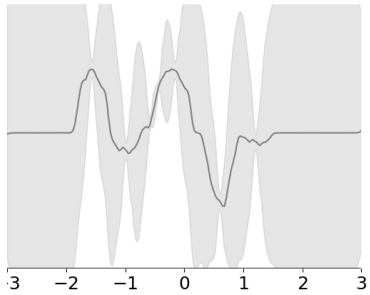


Discretization

Translation Equivariance in Practice

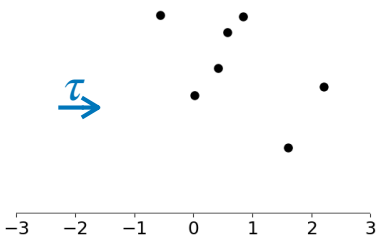


ConvCNP

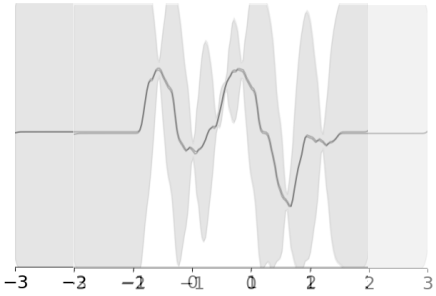


T_τ

T_τ

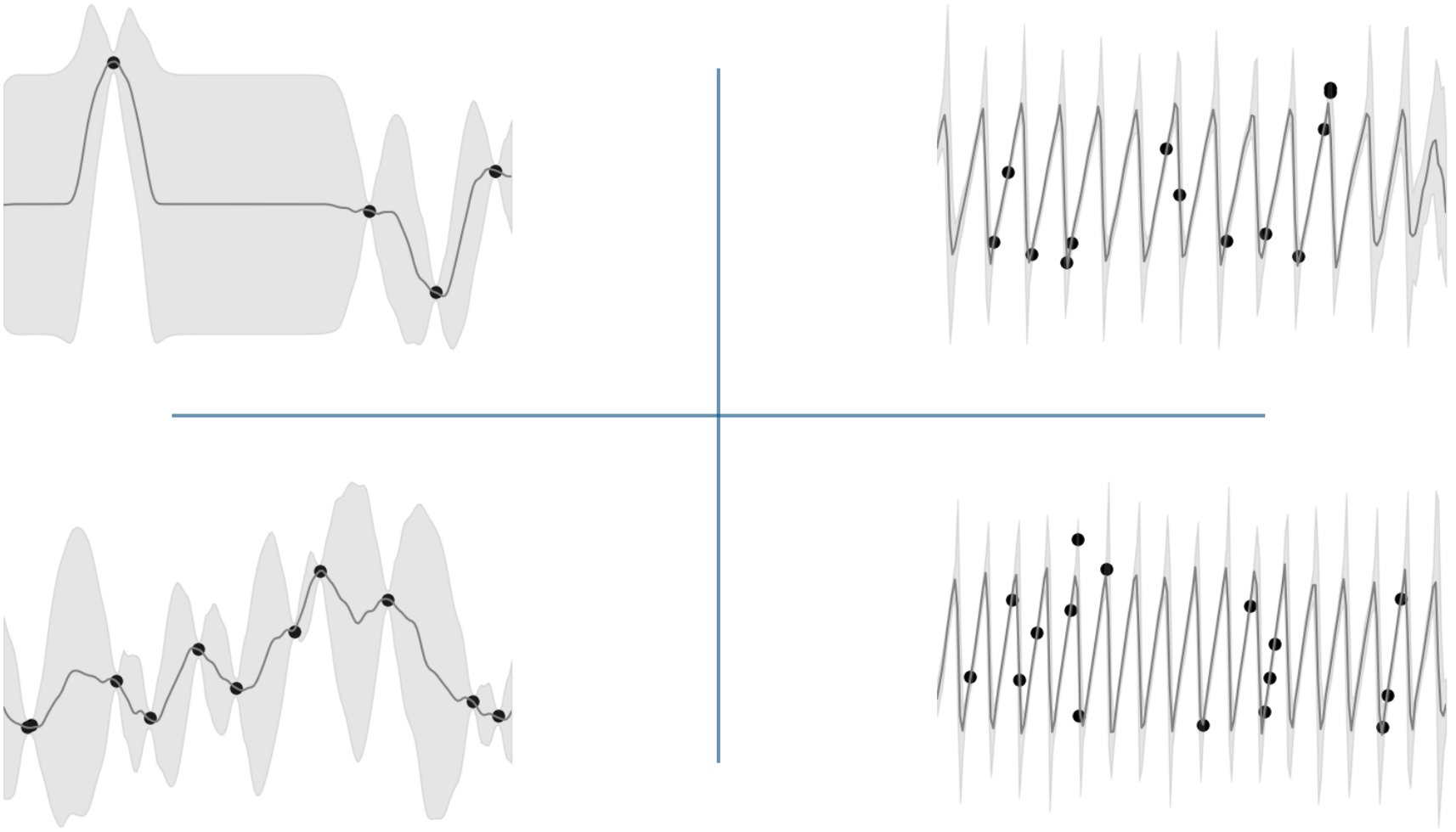


ConvCNP



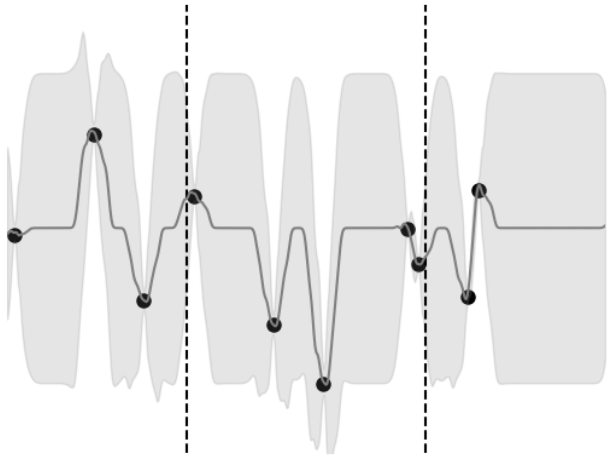
The Benefits of Translation Equivariance

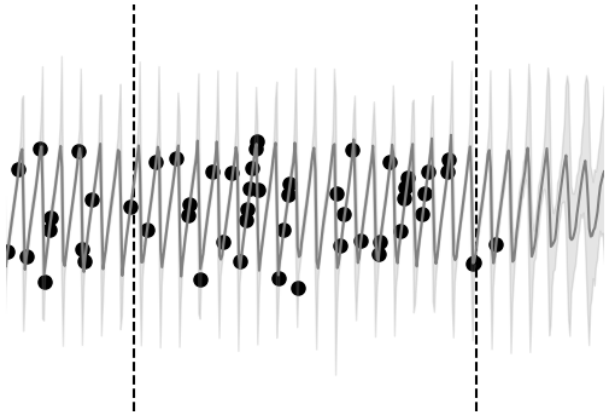
1D Regression Tasks



The Benefits of Translation Equivariance

Extrapolation





Simple 1D Regression Problems

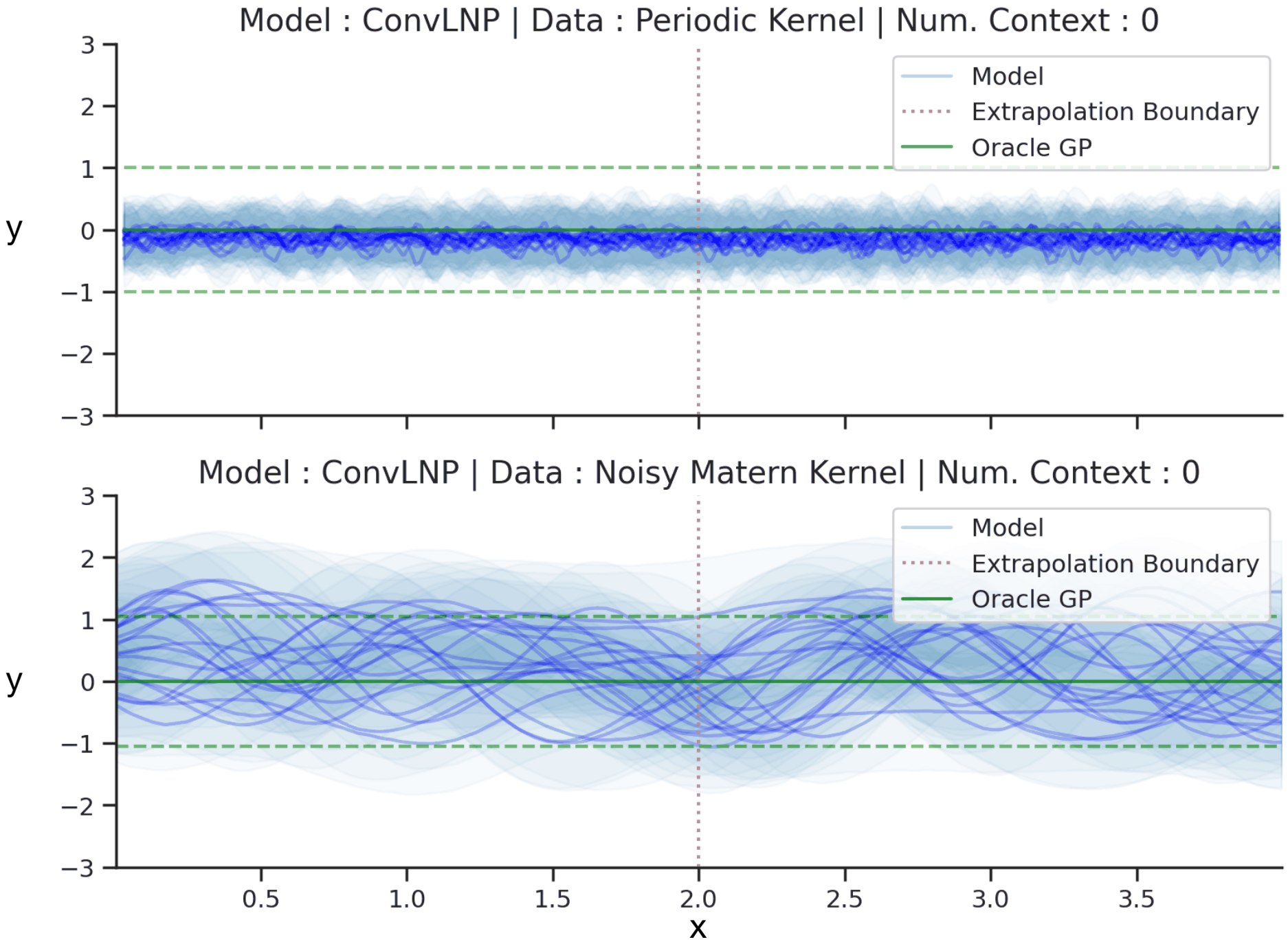
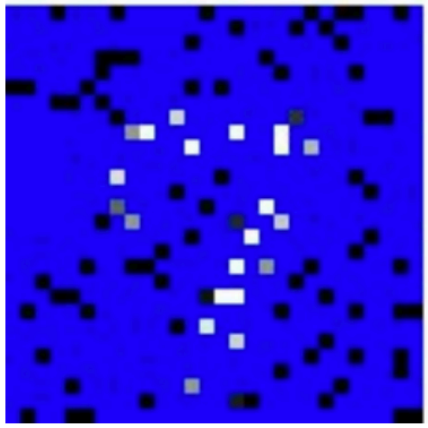


Image Reconstruction Experiments

$\mathrm{Im}\colon \mathbb{Z}^2 \to \mathbb{R}$

Original Image



Context set

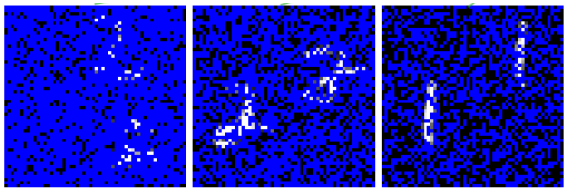
Reconstruction

Generalization with Image Data

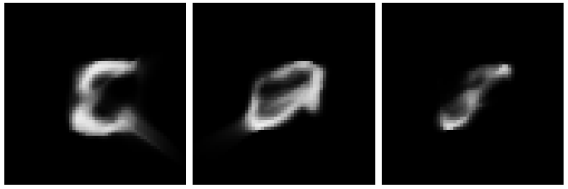
Train images

Test images

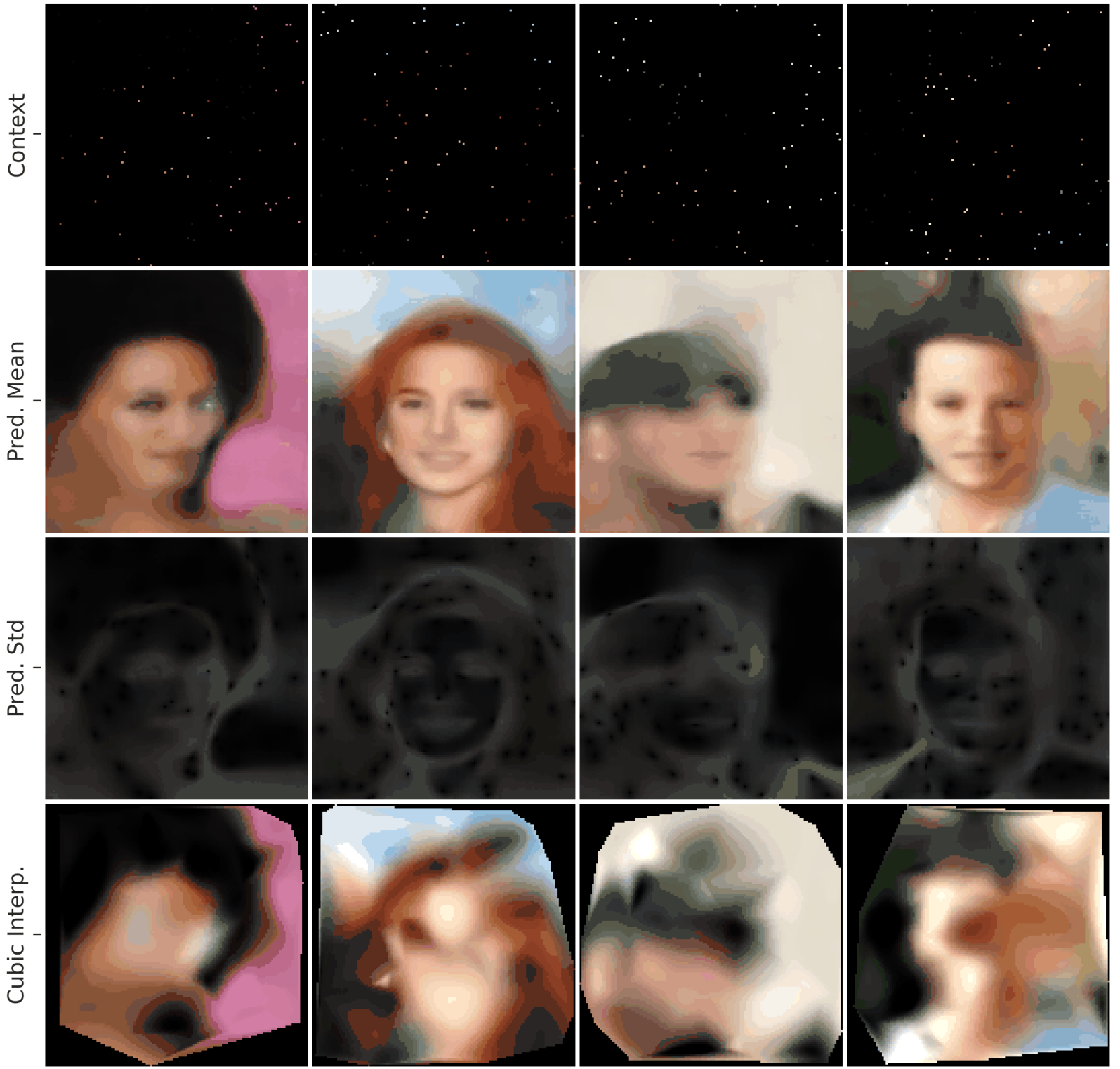
Context set



Attentive CNP

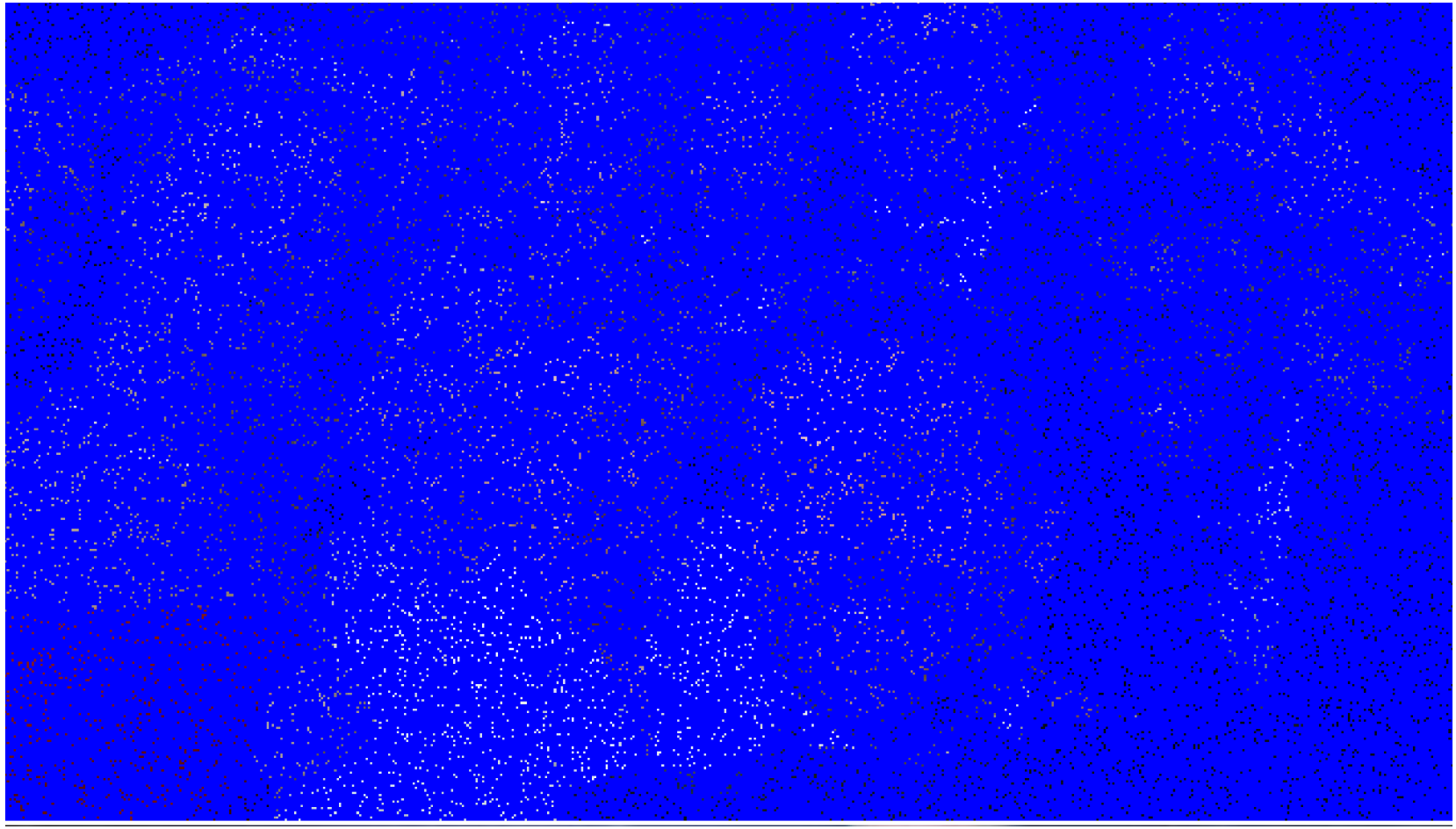


convCNP

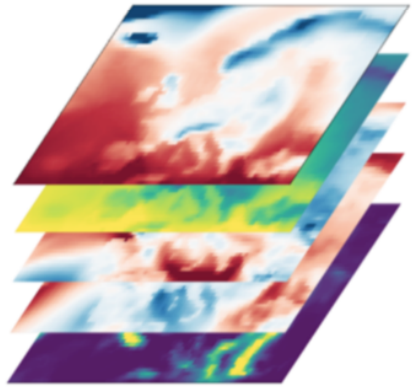

ConvCNPxl | Celeba128 | C=0.5%
Context
Pred. Mean
Pred. Std
Cubic Interp.

Generalization with Image Data

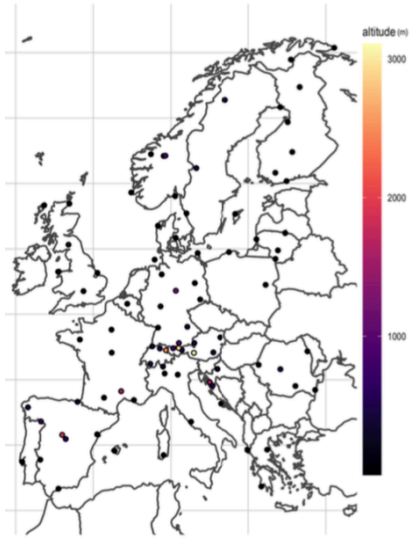
Train images



Climate Science: Statistical Downscaling



context set:
output of climate simulators
(coarse)

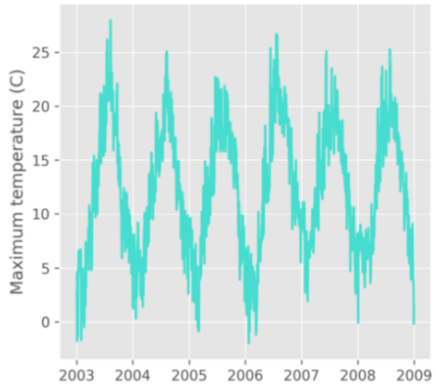


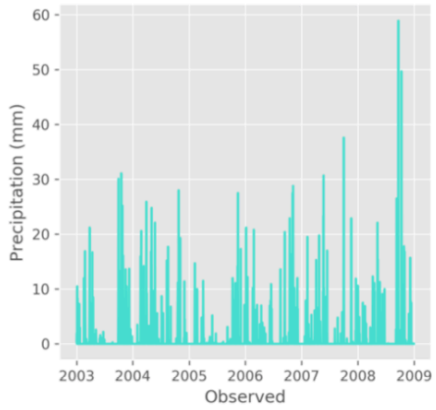
target inputs:
latitude, longitude, elevation

D_c

x_T

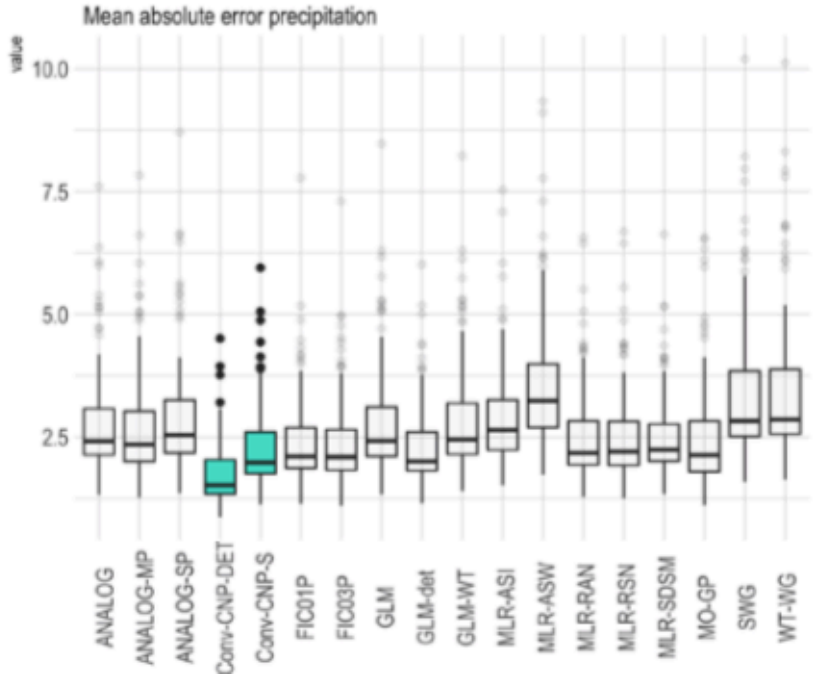
ConvCNP $\rightarrow y_T$

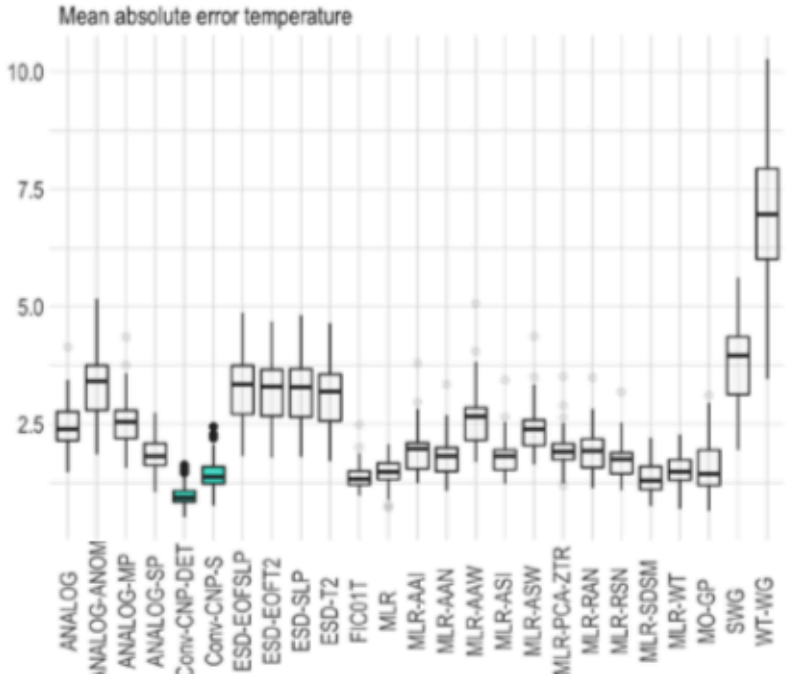


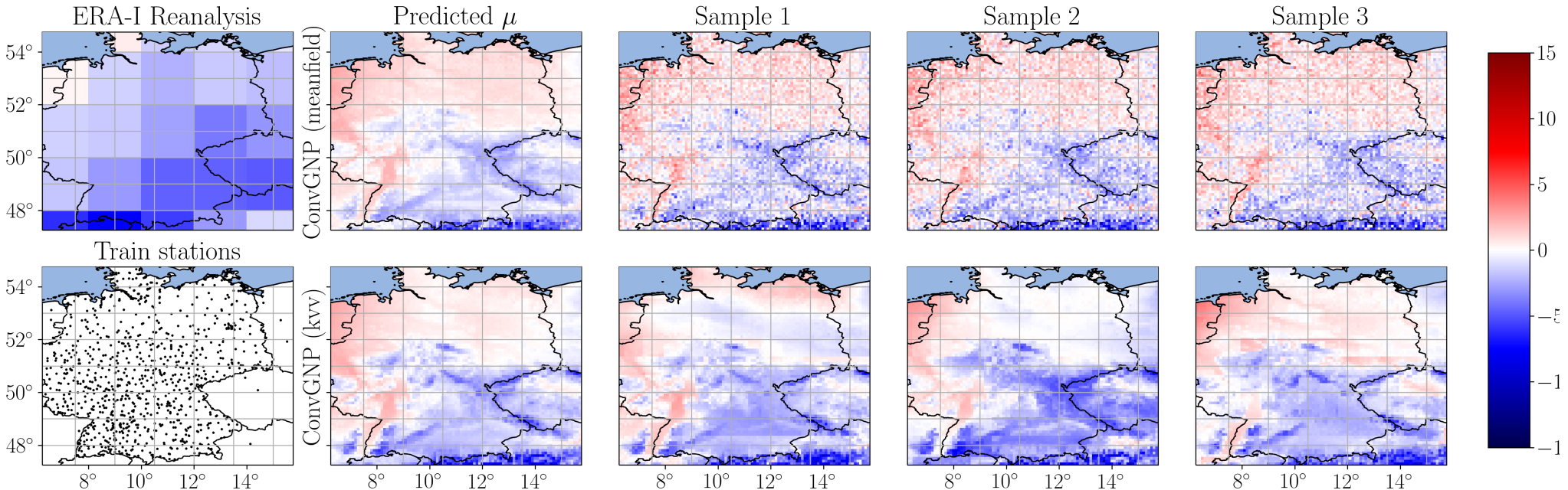


target outputs:
temperature and precipitation
(fine grained)

Climate Science: Statistical Downscaling

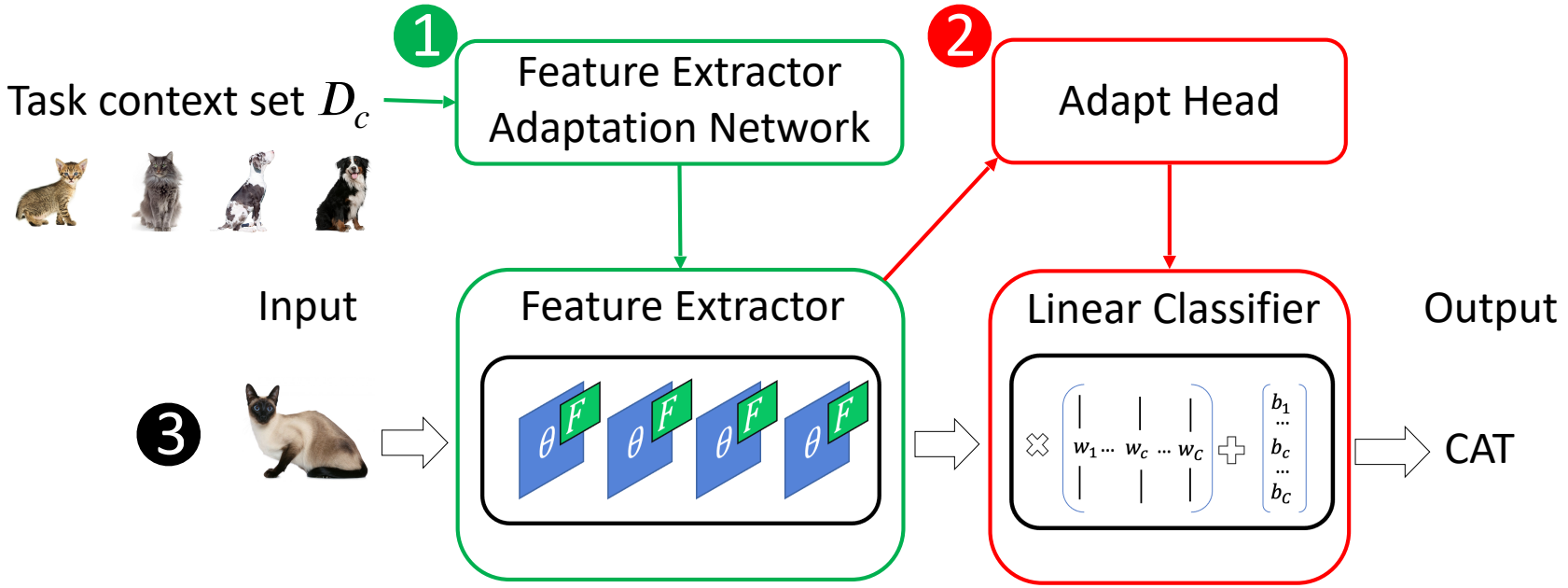




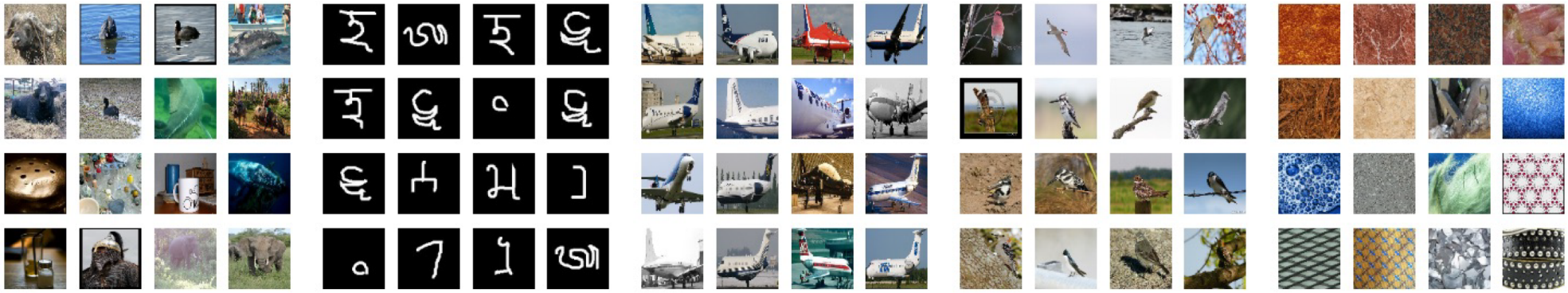


Application to classification

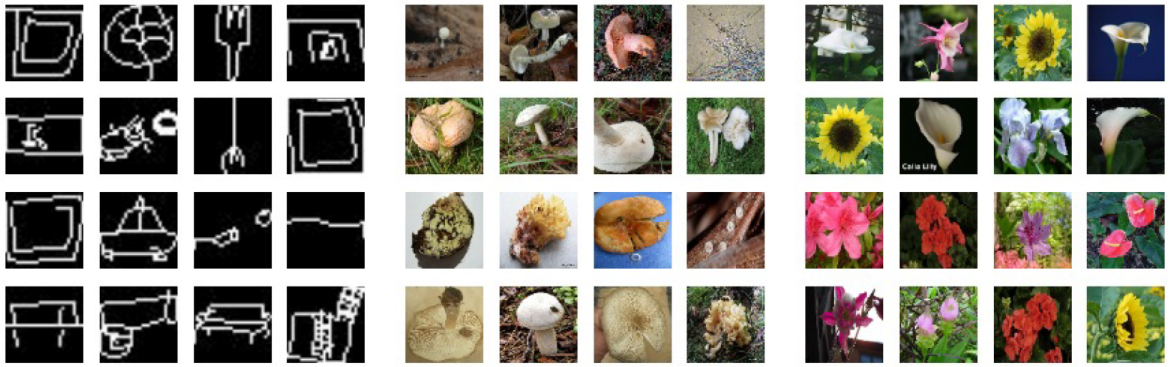
1. Use context set to adapt feature extractor (FiLM layers)
2. Pass context set class by class through adapted feature extractor and generate head
3. Apply adapted network to test examples



Meta-Dataset[1] Multi-task, Few-shot Benchmark



ImageNet OmniGlot Aircraft Birds DTD



Quick Draw Fungi VGG Flower Traffic Signs MSCOCO

[1] Eleni, et al. "Meta-dataset: A dataset of datasets for learning to learn from few examples."

Training Images

Stop Watch


Digital Clock



Digital Watch

Parking Meter

0.69 ✓	0.02	0.07	0.24
0.04	0.13	0.11	0.32
0.27	0.85 ✗	0.80 ✓	0.35 ✗
0	0.01	0.02	0.10

Testing Images




Training Images

Stop Watch


Digital Clock



Digital Watch

Parking Meter


0.63 ✓	0.02	0.04	0.11
0.04	0.09	0.13	0.11
0.29	0.70 ✗	0.61 ✓	0.24
0.04	0.19	0.22	0.54 ✓

Testing Images




Training Images

Stop Watch


Digital Clock



Digital Watch

Parking Meter


Sundial

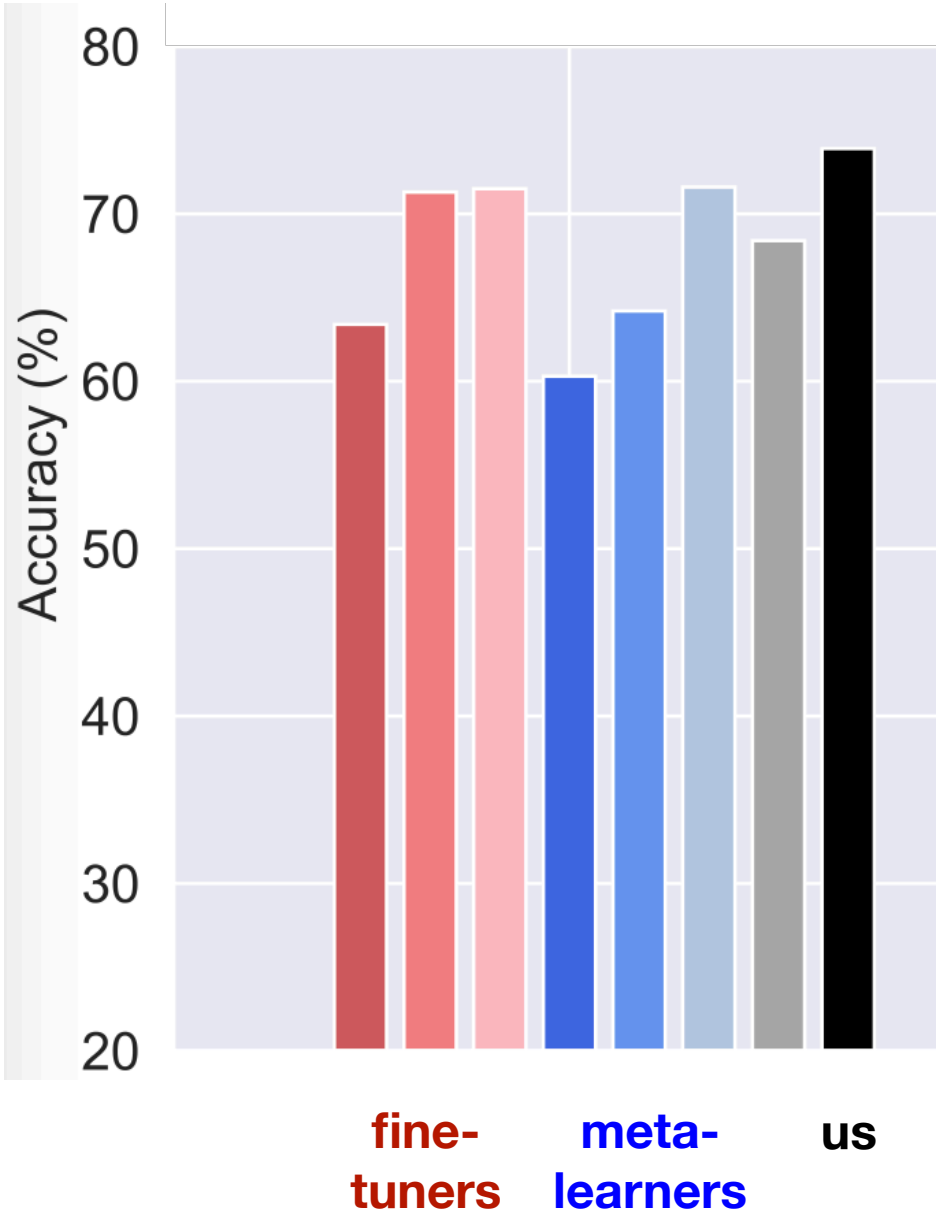
0.81 ✓	0.04	0.07	0.11	0.11
0.01	0.12	0.09	0.12	0.16
0.08	0.68 ✗	0.72 ✓	0.12	0.13
0.02	0.16	0.12	0.54 ✓	0.12
0.07	0	0	0.12	0.47 ✓

Testing Images





Meta-dataset results

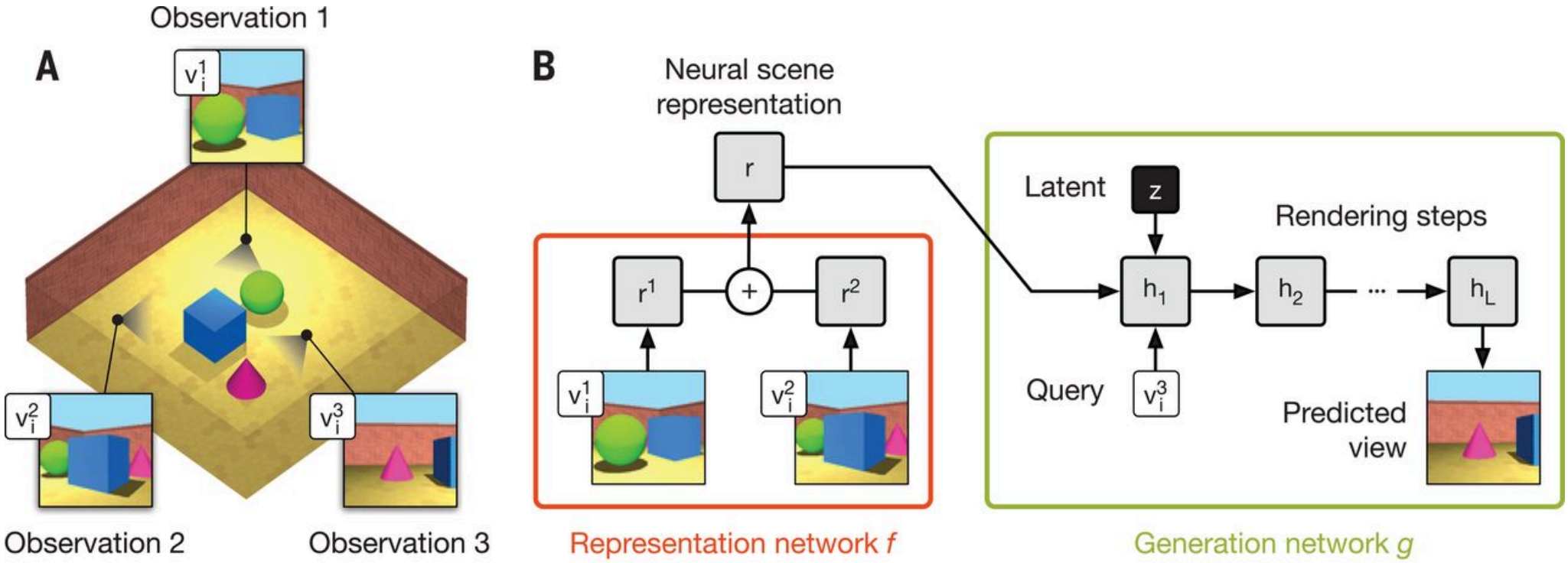


fine-tuners
MD-Transfer
SUR
BiT

meta-learners
ProtoNets
ProtoMAML
CTX

simple CNAPs
40x more efficient
at test time

Generative Query Network



Summary

- **Conditional Neural Processes** (**CNPs**): data-efficient meta-learning
  - Drawbacks: **severe under-fitting** and **failure to generalise**
- **ConvCNP endow CNPs with translation equivariance** (**TE**): powerful inductive bias
  - Significant improvements over previous CNP models
- **Conditional Neural Adaptive Processes** (CNAPs) enable data-efficient classification

Papers and other resources:

- Convolutional CNPs, ICLR 2020 (code https://github.com/cambridge-mlg/convcnp)
- Convolutional Neural Processes, NeurIPS 2020 (improved samples)
- Conditional Neural Adaptive Processes, NeurIPS 2019
  - applications to classification, continual learning (good for on device learning)
- Jupyter Book introduction to Neural Processes:
  - https://yanndubs.github.io/Neural-Process-Family/text/Intro.html
- Julia implementation (composition of neural processes and model building):
  - https://github.com/wesselb/NeuralProcesses.jl