In-Context In-Context Learning with Transformer Neural Processes

Matthew Ashman MCA39@CAM.AC.UK
University of Cambridge

Cristiana Diaconu CDD43@CAM.AC.UK
University of Cambridge

Adrian Weller AW665@CAM.AC.UK
University of Cambridge
The Alan Turing Institute

Richard E. Turner RET26@CAM.AC.UK
University of Cambridge
Microsoft Research AI for Science

July 2024

1. Neural Processes

Meta Learning Setting

- Seek to approximate the posterior predictive map of a ground-truth $\mathcal{Y}$-valued stochastic process on input space $\mathcal{X}$

Meta Learning Setting

- Seek to approximate the posterior predictive map of a ground-truth $\mathcal{Y}$-valued stochastic process on input space $\mathcal{X}$
- Given a dataset $\mathcal{D}_c = (\mathbf{X}_c, \mathbf{Y}_c)$ of N_c context datapoints, where

$$\mathbf{X}_c \in \mathbb{R}^{N_c \times D_x} \qquad \mathbf{Y}_c \in \mathbb{R}^{N_c \times D_y}$$

approximate $p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c)$ for N_t target inputs $\mathbf{X}_t \in \mathbb{R}^{N_t \times D_x}$

Meta Learning Setting

- Seek to approximate the posterior predictive map of a ground-truth $\mathcal{Y}$-valued stochastic process on input space $\mathcal{X}$
- Given a dataset $\mathcal{D}_c = (\mathbf{X}_c, \mathbf{Y}_c)$ of N_c context datapoints, where

$$\mathbf{X}_c \in \mathbb{R}^{N_c \times D_x} \qquad \mathbf{Y}_c \in \mathbb{R}^{N_c \times D_y}$$

approximate $p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c)$ for N_t target inputs $\mathbf{X}_t \in \mathbb{R}^{N_t \times D_x}$

We want our model to be

- Invariant to permutations of $\mathcal{D}_c$ and $\mathcal{D}_t$

Meta Learning Setting

- Seek to approximate the posterior predictive map of a ground-truth $\mathcal{Y}$-valued stochastic process on input space $\mathcal{X}$
- Given a dataset $\mathcal{D}_c = (\mathbf{X}_c, \mathbf{Y}_c)$ of N_c context datapoints, where

$$\mathbf{X}_c \in \mathbb{R}^{N_c \times D_x} \qquad \mathbf{Y}_c \in \mathbb{R}^{N_c \times D_y}$$

approximate $p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c)$ for N_t target inputs $\mathbf{X}_t \in \mathbb{R}^{N_t \times D_x}$

We want our model to be

- Invariant to permutations of $\mathcal{D}_c$ and $\mathcal{D}_t$
- Able to deal with an arbitrary number of datapoints N_c and N_t

Meta Learning Setting

- Seek to approximate the posterior predictive map of a ground-truth $\mathcal{Y}$-valued stochastic process on input space $\mathcal{X}$
- Given a dataset $\mathcal{D}_c = (\mathbf{X}_c, \mathbf{Y}_c)$ of N_c context datapoints, where

$$\mathbf{X}_c \in \mathbb{R}^{N_c \times D_x} \qquad \mathbf{Y}_c \in \mathbb{R}^{N_c \times D_y}$$

approximate $p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c)$ for N_t target inputs $\mathbf{X}_t \in \mathbb{R}^{N_t \times D_x}$

We want our model to be

- Invariant to permutations of $\mathcal{D}_c$ and $\mathcal{D}_t$
- Able to deal with an arbitrary number of datapoints N_c and N_t

(so we might consider a transformer-based model…)

Neural Processes (NPs)

- Model using an encoder $e : \mathcal{X} \times \mathcal{S} \to \mathcal{Z}$ and a decoder $d : \mathcal{X} \times \mathcal{Z} \to \Theta$

$$p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c) = p(\mathbf{Y}_t | d(\mathbf{X}_t, e(\mathbf{X}_t, \mathcal{D}_c))$$

where:

- $\mathcal{S}$ is the set of all finite datasets ($\mathcal{D}_c, \mathcal{D}_t \in \mathcal{S}$)
- $\mathcal{Z}$ is some latent space
- Θ is the parameter space of our predictive distribution.

Neural Processes (NPs)

- Model using an encoder $e : \mathcal{X} \times \mathcal{S} \rightarrow \mathcal{Z}$ and a decoder $d : \mathcal{X} \times \mathcal{Z} \rightarrow \Theta$

$$p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c) = p(\mathbf{Y}_t | d(\mathbf{X}_t, e(\mathbf{X}_t, \mathcal{D}_c))$$

where:

- $\mathcal{S}$ is the set of all finite datasets ($\mathcal{D}_c, \mathcal{D}_t \in \mathcal{S}$)
- $\mathcal{Z}$ is some latent space
- Θ is the parameter space of our predictive distribution.

Train by maximising the posterior predictive likelihood over tasks $\tau = (\mathcal{D}_c, \mathcal{D}_t)$

$$\mathcal{L}_{\mathrm{ML}} = \mathbb{E}_{p(\tau)} \left[\sum_{n=1}^{N_t} \log p(\mathbf{y}_{t,n} | d(\mathbf{x}_{t,n}, e(\mathbf{x}_{t,n}, \mathcal{D}_c)) \right]$$

Neural Processes (NPs)

- Model using an encoder $e : \mathcal{X} \times \mathcal{S} \to \mathcal{Z}$ and a decoder $d : \mathcal{X} \times \mathcal{Z} \to \Theta$

$$p(\mathbf{Y}_t | \mathbf{X}_t, \mathcal{D}_c) = p(\mathbf{Y}_t | d(\mathbf{X}_t, e(\mathbf{X}_t, \mathcal{D}_c))$$

where:

- $\mathcal{S}$ is the set of all finite datasets ($\mathcal{D}_c, \mathcal{D}_t \in \mathcal{S}$)
- $\mathcal{Z}$ is some latent space
- Θ is the parameter space of our predictive distribution.

Train by maximising the posterior predictive likelihood over tasks $\tau = (\mathcal{D}_c, \mathcal{D}_t)$

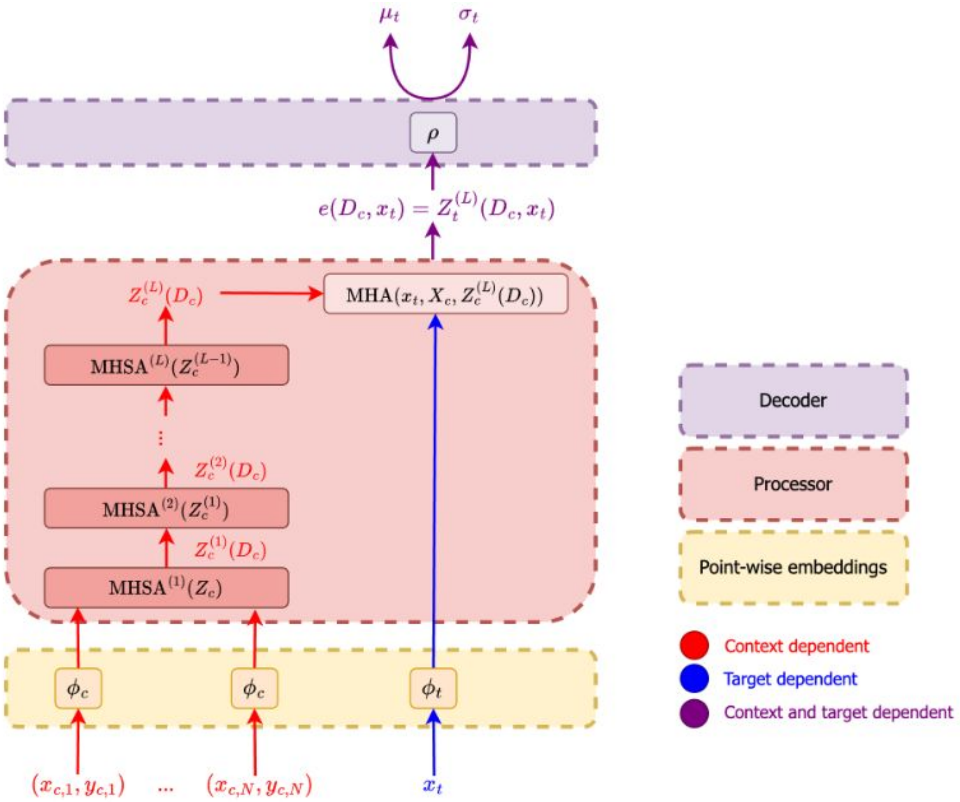
$$\mathcal{L}_{\mathrm{ML}} = \mathbb{E}_{p(\tau)} \left[\sum_{n=1}^{N_t} \log p(\mathbf{y}_{t,n} | d(\mathbf{x}_{t,n}, e(\mathbf{x}_{t,n}, \mathcal{D}_c)) \right]$$

(Use a Monte-Carlo estimate of this expectation using the tasks you actually have access to.)

2. Transformer Neural Processes

Transformer Neural Processes (TNPs)

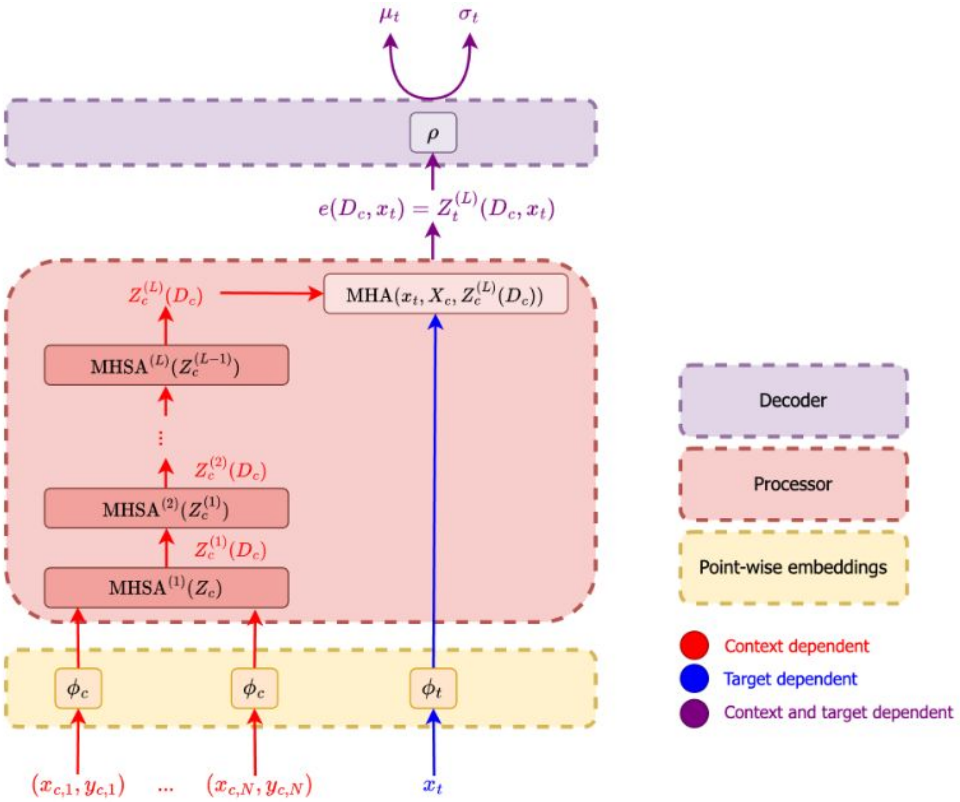
- Obtain initial token representation for context points via pointwise embeddings $\mathbf{Z}_c^0 = \phi_c(\mathcal{D}_c) \in \mathbb{R}^{N_c \times D_z}$



(Attentive Neural Process, Kim et al., 2019)

Transformer Neural Processes (TNPs)

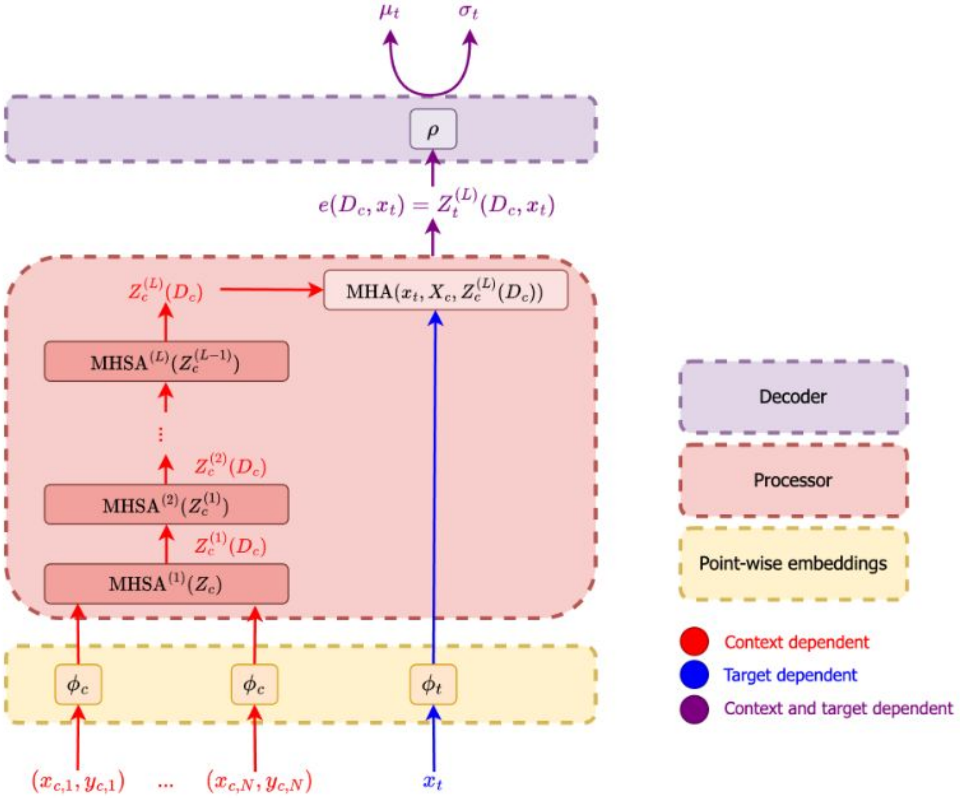
- Obtain initial token representation for context points via pointwise embeddings $\mathbf{Z}_c^0 = \phi_c(\mathcal{D}_c) \in \mathbb{R}^{N_c \times D_z}$
- Pass tokens through L multi-headed self-attention layers



(Attentive Neural Process, Kim et al., 2019)

Transformer Neural Processes (TNPs)

- Obtain initial token representation for context points via pointwise embeddings $\mathbf{Z}_c^0 = \phi_c(\mathcal{D}_c) \in \mathbb{R}^{N_c \times D_z}$
- Pass tokens through L multi-headed self-attention layers
- Perform cross attention with
  - Values Z_c
  - Keys X_c
  - Queries $\phi_t(x_t)$

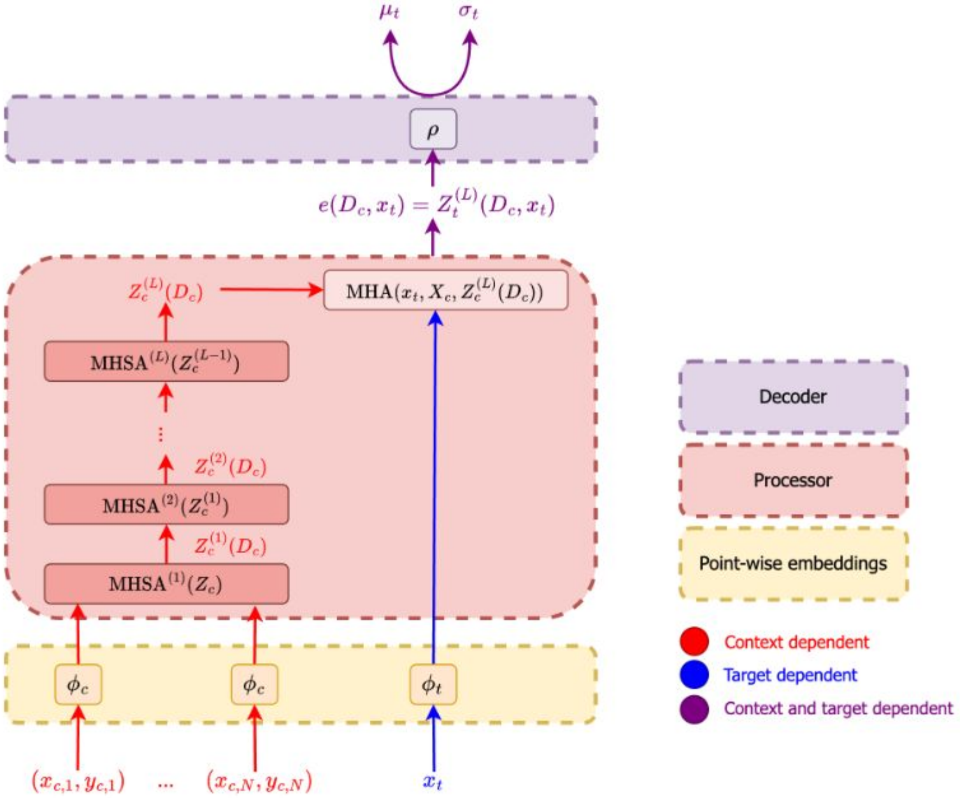


(Attentive Neural Process, Kim et al., 2019)

Transformer Neural Processes (TNPs)

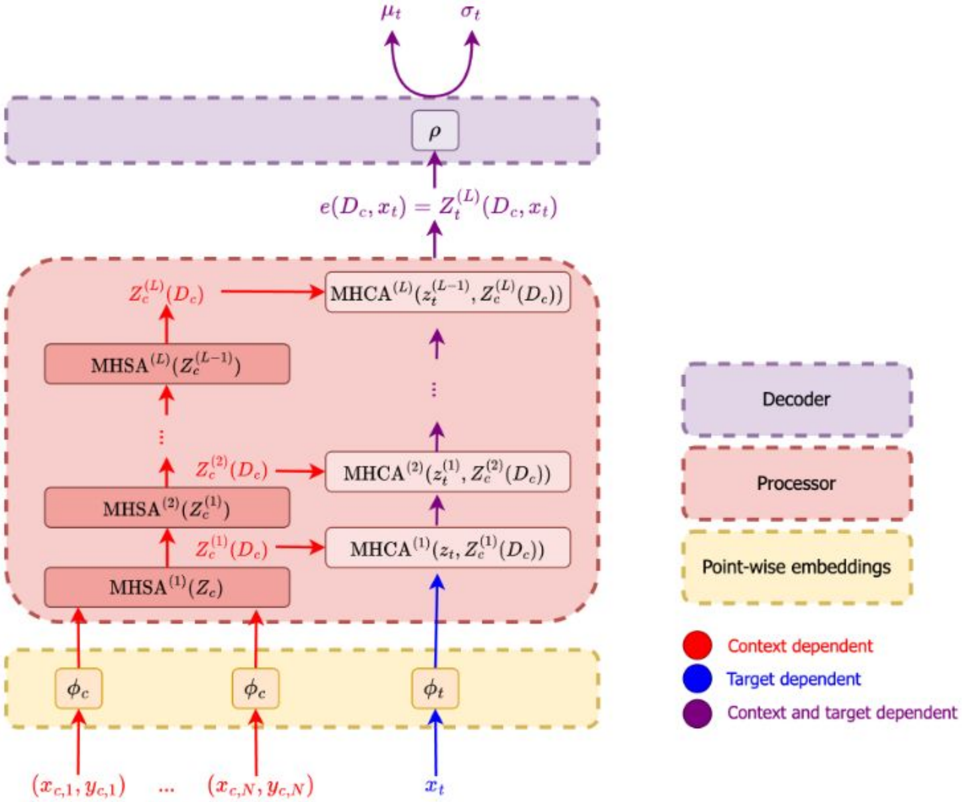
- Obtain initial token representation for context points via pointwise embeddings $\mathbf{Z}_c^0 = \phi_c(\mathcal{D}_c) \in \mathbb{R}^{N_c \times D_z}$
- Pass tokens through L multi-headed self-attention layers
- Perform cross attention with
  - Values Z_c
  - Keys X_c
  - Queries $\phi_t(x_t)$
- Pass output through decoder ρ (a 2-layer, 128-width MLP) to get predictive distribution (mean and variance of a Gaussian).

(Attentive Neural Process, Kim et al., 2019)



Transformer Neural Processes (TNPs)

- Better idea: add in cross-attention between context tokens Z_c and updated target token z_t at each layer

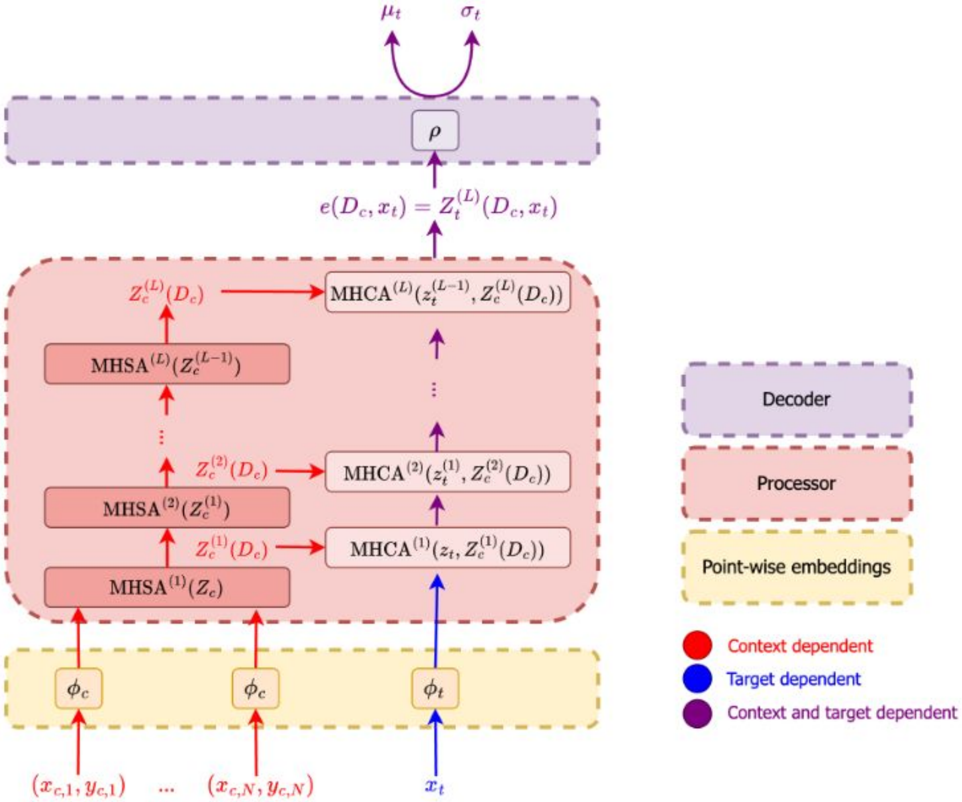


(Transformer Neural Process, Nguyen and Grover, 2022)

Transformer Neural Processes (TNPs)

- Better idea: add in cross-attention between context tokens Z_c and updated target token z_t at each layer

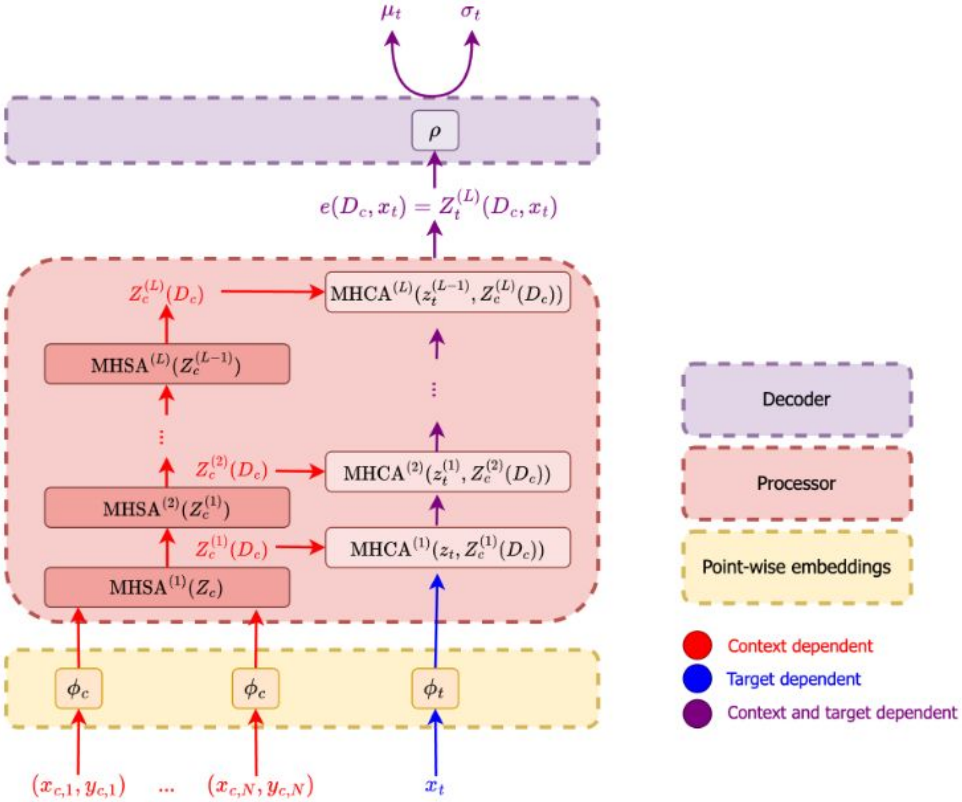
- Problem: $\mathcal{O}\left(N_c^2 + N_c N_t\right)$



(Transformer Neural Process, Nguyen and Grover, 2022)

Transformer Neural Processes (TNPs)

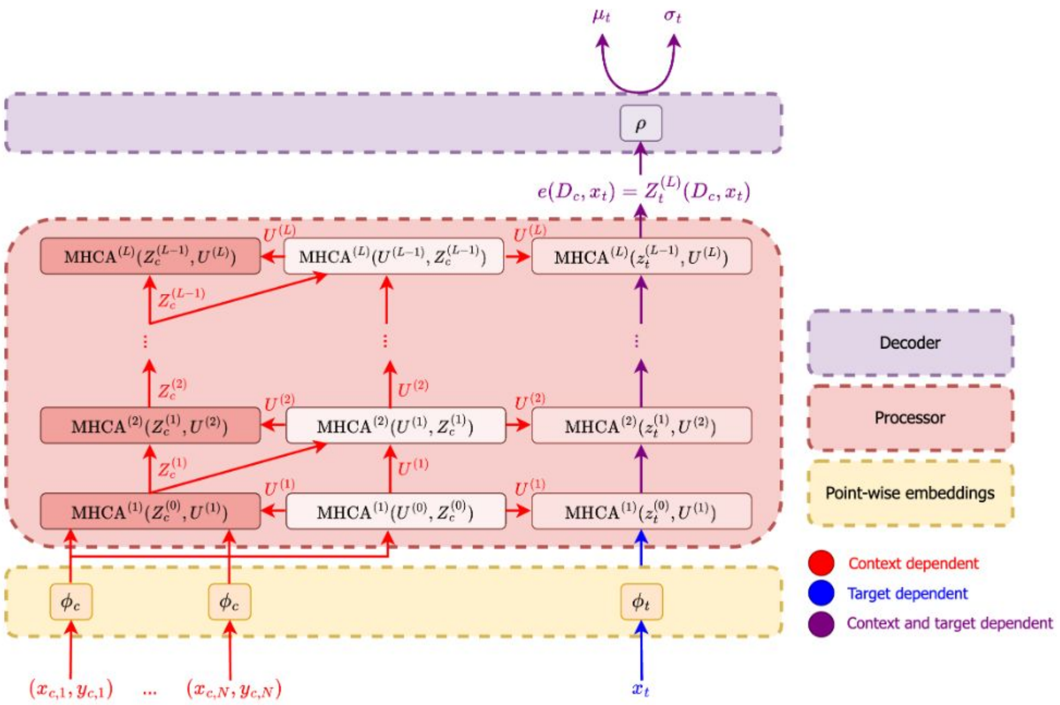
- Better idea: add in cross-attention between context tokens Z_c and updated target token z_t at each layer
- Problem: $\mathcal{O}\left(N_c^2 + N_c N_t\right)$
- Solution: pseudo tokens



(Transformer Neural Process, Nguyen and Grover, 2022)

Pseudo Token TNP (PT-TNP)

- Add in $M \ll N_c$ ‘pseudo tokens’, $\mathrm{U} \in \mathbb{R}^{M \times D_z}$, to distill/summarise the latent context tokens, Z_c .

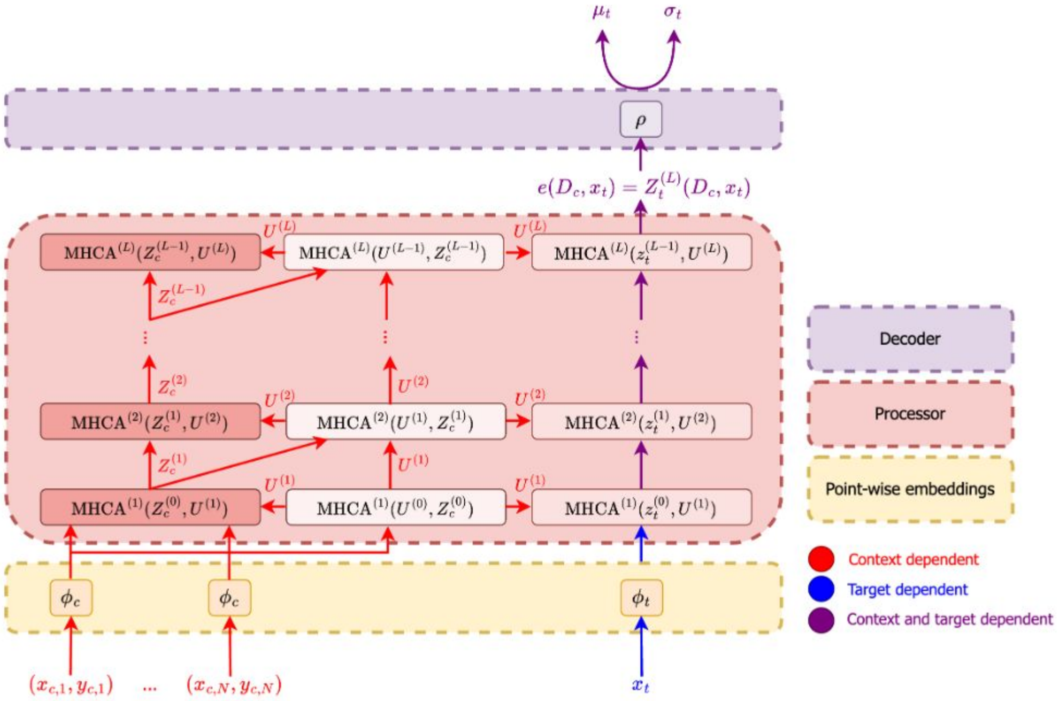


(Set Transformer, Lee et al., 2019)

Pseudo Token TNP (PT-TNP)

- Add in $M \ll N_c$ 'pseudo tokens', $\mathrm{U} \in \mathbb{R}^{M \times D_z}$, to distill/summarise the latent context tokens, Z_c .
- Remove the quadratic cost of performing attention between context and target

$$\mathcal{O}\left(MN_c + MN_t + M^2\right)$$



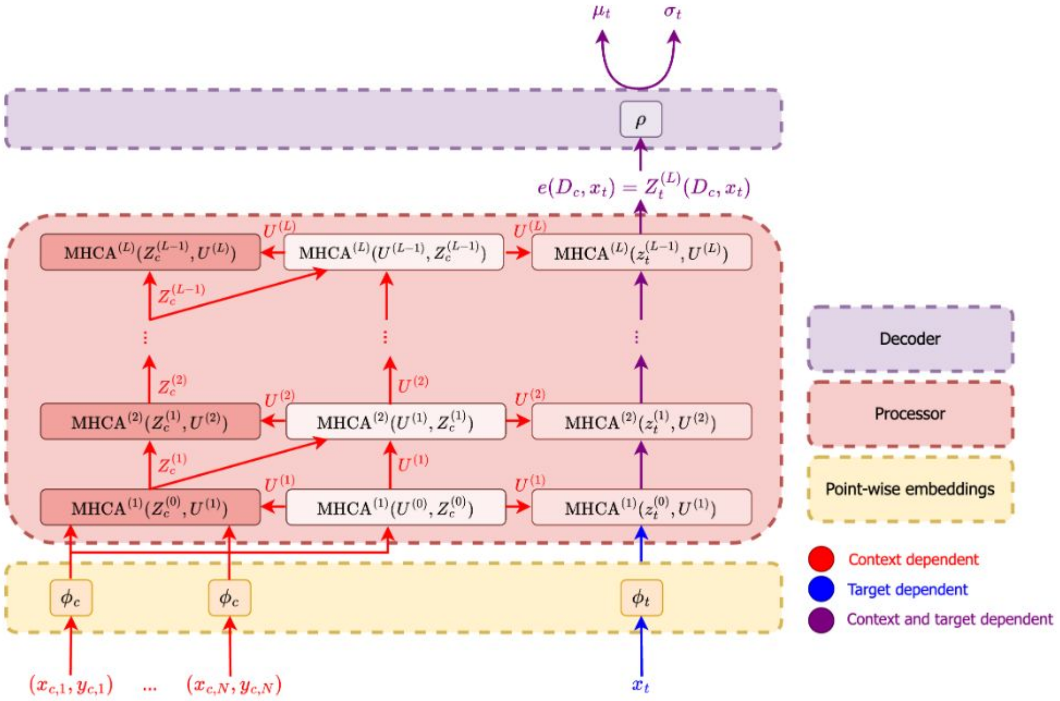
(Set Transformer, Lee et al., 2019)

Pseudo Token TNP (PT-TNP)

- Add in $M \ll N_c$ 'pseudo tokens', $\mathrm{U} \in \mathbb{R}^{M \times D_z}$, to distill/summarise the latent context tokens, Z_c .
- Remove the quadratic cost of performing attention between context and target

$$\mathcal{O}\left(MN_c + MN_t + M^2\right)$$

- Now we're linear in dataset size



(Set Transformer, Lee et al., 2019)

3. In-Context In-Context Learning

What if we have related (‘in-context’) data?

- We may have multiple related datasets from the target stochastic process
- E.g. a meta-dataset of PDE-simulated Navier-Stokes equations*
    - Could be partitioned into multiple sets based on the Reynold’s number (dimensionless parameter describing fluid flow)
    - Infeasible: train one NP per Reynold’s number (from 10^-6 to 10^12)
    - Idea: learn a single NP that can condition on each partition of the meta-dataset
- “*In-context in-context learning*”
    - “*amortizing the learning of stochastic-process specific NPs*”

(*) Note: training a single NP on the whole non-partitioned meta-dataset would give the predictive distribution of the marginal stochastic process

Theorem: More data is good

Theorem 1 (In-context in-context learning) *Let* $\xi_i \sim p(\xi)$, $\mathcal{D}_i, \{\mathcal{D}_j\} \sim P(\xi_i)$. *Let* $p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \xi_i)$ *be the marginal posterior distribution of* $P(\xi_i)$ *given* $\mathcal{D}_i$, $p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \{\mathcal{D}_j\})$ *be the marginal posterior distribution of the stochastic process* P *given* $\mathcal{D}_i$ *and* $\{\mathcal{D}_j\}$, *and* $p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i)$ *be the marginal posterior distribution of the stochastic process* P *given* $\mathcal{D}_i$. *Then,*

$$\mathbb{E}_{\mathcal{D}_i,\{\mathcal{D}_j\},\xi_i}\left[\mathrm{KL}\left[p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \xi_i)||p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \{\mathcal{D}_j\})\right]\right] \leq \mathbb{E}_{\mathcal{D}_i,\xi_i}\left[\mathrm{KL}\left[p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \xi_i)||p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i)\right]\right].$$

Theorem: More data is good

Theorem 1 (In-context in-context learning) *Let* $\xi_i \sim p(\xi)$, $\mathcal{D}_i, \{\mathcal{D}_j\} \sim P(\xi_i)$. *Let* $p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \xi_i)$ *be the marginal posterior distribution of* $P(\xi_i)$ *given* $\mathcal{D}_i$, $p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \{\mathcal{D}_j\})$ *be the marginal posterior distribution of the stochastic process* P *given* $\mathcal{D}_i$ *and* $\{\mathcal{D}_j\}$, *and* $p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i)$ *be the marginal posterior distribution of the stochastic process* P *given* $\mathcal{D}_i$. *Then,*

$$\mathbb{E}_{\mathcal{D}_i, \{\mathcal{D}_j\}, \xi_i} \left[\mathrm{KL} \left[p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \xi_i) || p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \{\mathcal{D}_j\}) \right] \right] \leq \mathbb{E}_{\mathcal{D}_i, \xi_i} \left[\mathrm{KL} \left[p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \xi_i) || p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i) \right] \right].$$

If we have extra datasets $\{\mathcal{D}_j\}$from the stochastic process $P(\xi_i)$ (on top of $\mathcal{D}_i$), then it can't hurt to condition on those too.

Theorem: More data is good

Theorem 1 (In-context in-context learning) *Let $\xi_i \sim p(\xi)$, $\mathcal{D}_i, \{\mathcal{D}_j\} \sim P(\xi_i)$. Let $p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \xi_i)$ be the marginal posterior distribution of $P(\xi_i)$ given $\mathcal{D}_i$, $p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \{\mathcal{D}_j\})$ be the marginal posterior distribution of the stochastic process P given $\mathcal{D}_i$ and $\{\mathcal{D}_j\}$, and $p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i)$ be the marginal posterior distribution of the stochastic process P given $\mathcal{D}_i$. Then,*

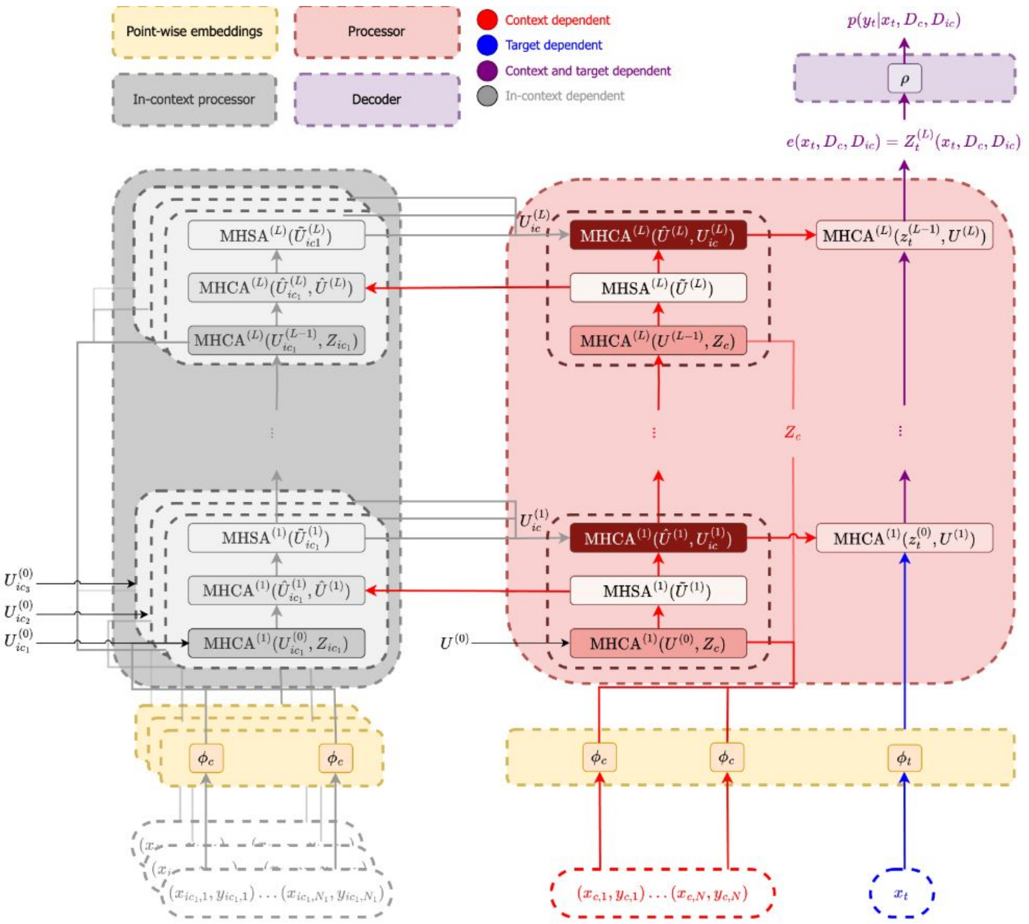
$$\mathbb{E}_{\mathcal{D}_i,\{\mathcal{D}_j\},\xi_i}\left[\mathrm{KL}\left[p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \xi_i)||p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \{\mathcal{D}_j\})\right]\right] \leq \mathbb{E}_{\mathcal{D}_i,\xi_i}\left[\mathrm{KL}\left[p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i, \xi_i)||p(\mathbf{y}|\mathbf{x}, \mathcal{D}_i)\right]\right].$$

If we have extra datasets $\{\mathcal{D}_j\}$from the stochastic process $P(\xi_i)$ (on top of $\mathcal{D}_i$), then it can't hurt to condition on those too.

We refer to the additional datasets as the *in-context datasets* $\{\mathcal{D}_{ic,j}\}_{j=1}^{N_{ic}}$

In-Context In-Context Learning with TNPs (ICICL-TNP)

Create extra pseudo-tokens for each of the in-context datasets, U_{ic_i}.

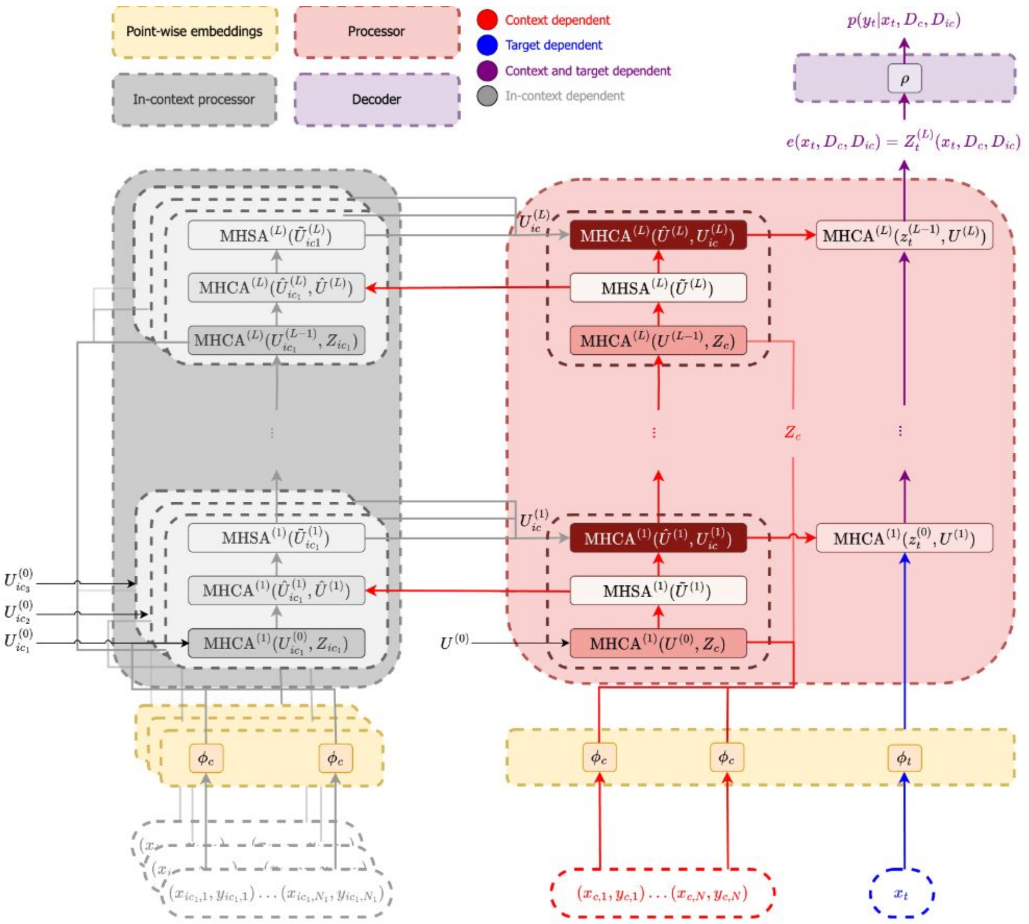


In-Context In-Context Learning with TNPs (ICICL-TNP)

Create extra pseudo-tokens for each of the in-context datasets, U_{ic_i}.

At each layer we perform cross attention:

1. Between latents Z_c (or Z_{ic_i}) and pseudo tokens U (or U_{ic_j})

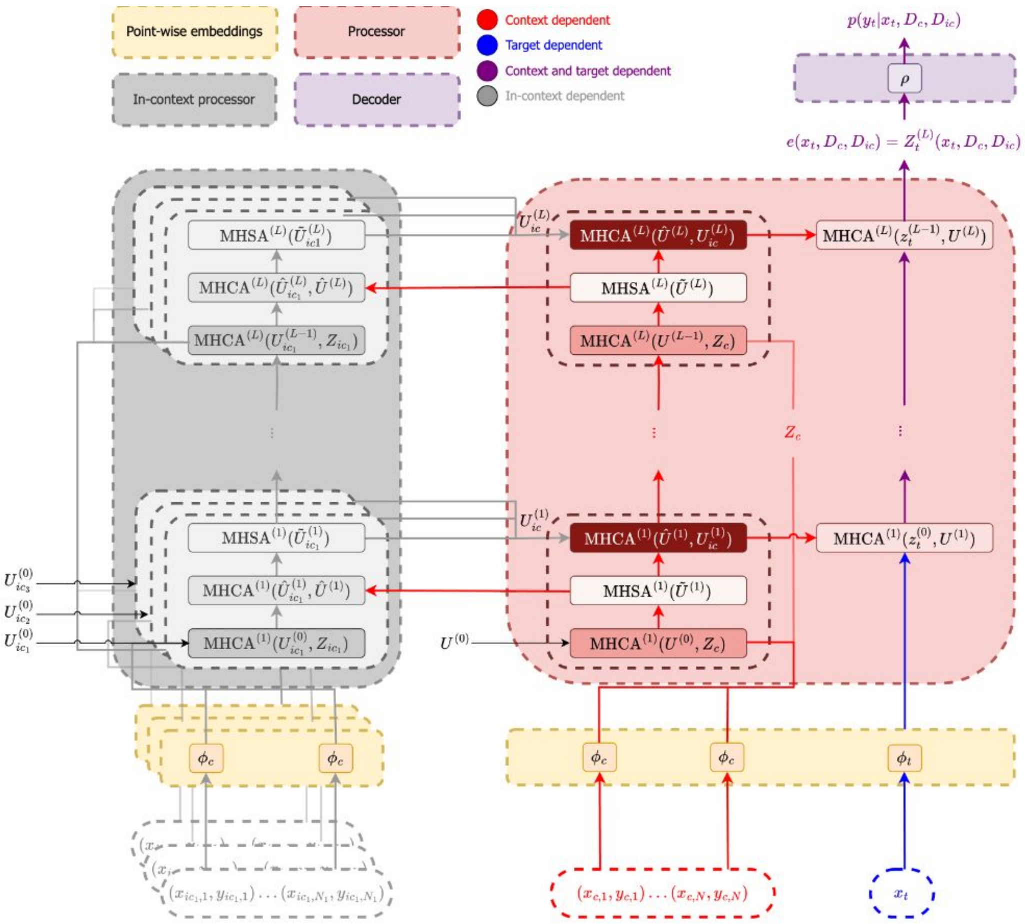


In-Context In-Context Learning with TNPs (ICICL-TNP)

Create extra pseudo-tokens for each of the in-context datasets, U_{ic_i}.

At each layer we perform cross attention:

1. Between latents Z_c (or Z_{ic_i}) and pseudo tokens U (or U_{ic_j})
2. Between context (i.e. regular) pseudo tokens U and concatenated in-context pseudo tokens U_{ic}

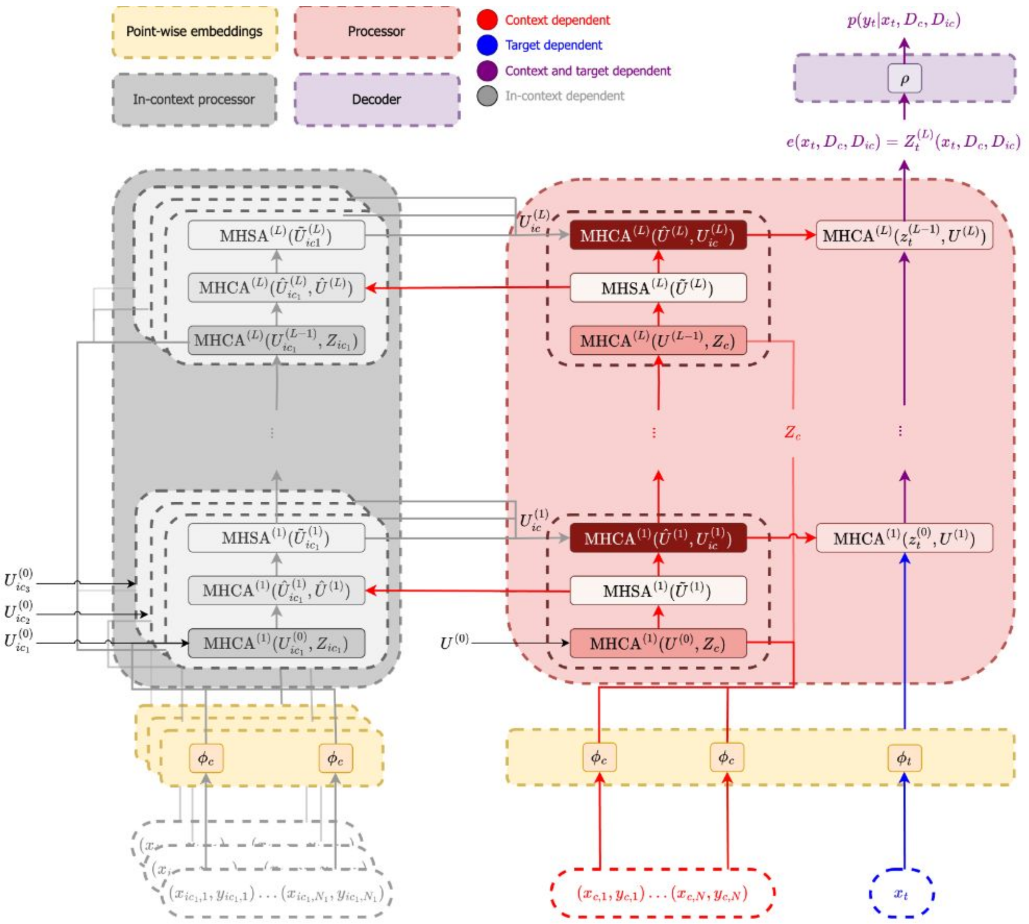


In-Context In-Context Learning with TNPs (ICICL-TNP)

Create extra pseudo-tokens for each of the in-context datasets, U_{ic_i}.

At each layer we perform cross attention:

1. Between latents Z_c (or Z_{ic_i}) and pseudo tokens U (or U_{ic_j})
2. Between context (i.e. regular) pseudo tokens U and concatenated in-context pseudo tokens U_{ic}
3. Between target latents z_t and pseudo tokens U (as in regular PT-TNPs)



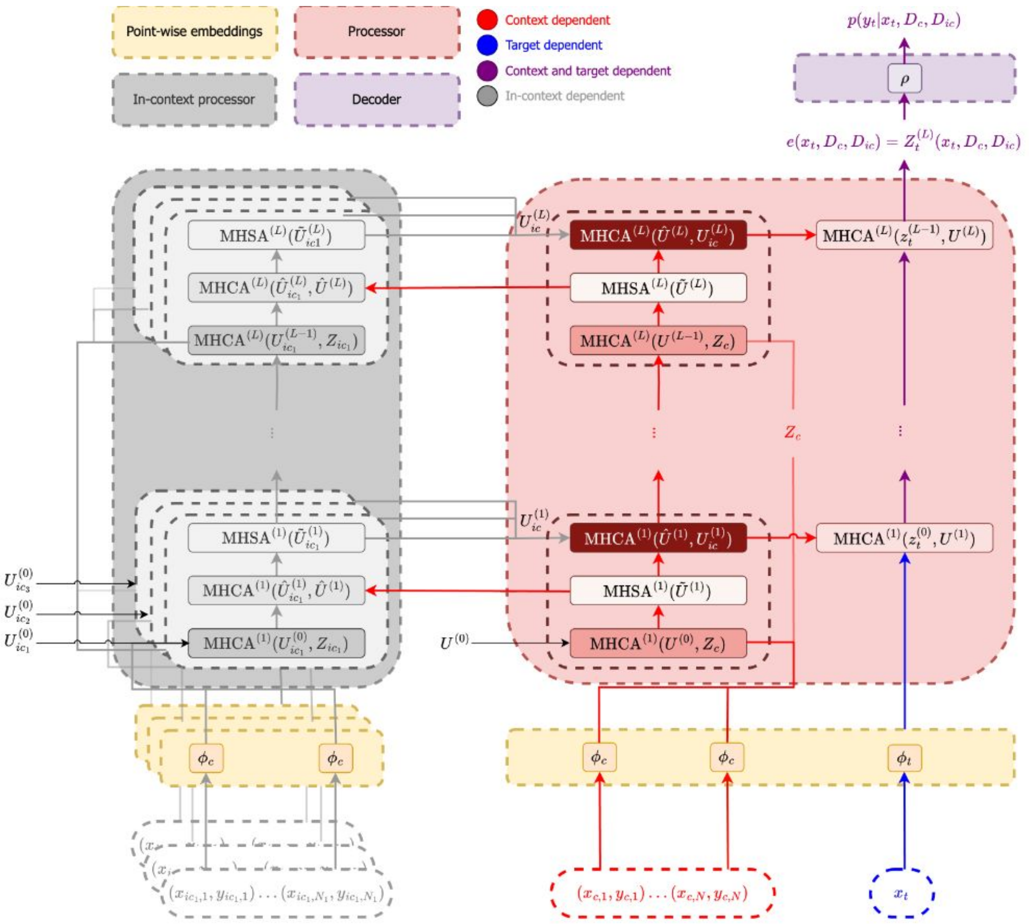
In-Context In-Context Learning with TNPs (ICICL-TNP)

Create extra pseudo-tokens for each of the in-context datasets, U_{ic_i}.

At each layer we perform cross attention:

1. Between latents Z_c(or Z_{ic_i}) and pseudo tokens U (or U_{ic_j})
2. Between context (i.e. regular) pseudo tokens U and concatenated in-context pseudo tokens U_{ic}
3. Between target latents z_t and pseudo tokens U (as in regular PT-TNPs)

Basically, the in-context processor maintains in-context latents Z_{ic_i} and in-context pseudo tokens U_{ic} which are used to modulate the main processor's pseudo tokens U



In-Context In-Context Learning with TNPs (ICICL-TNP)

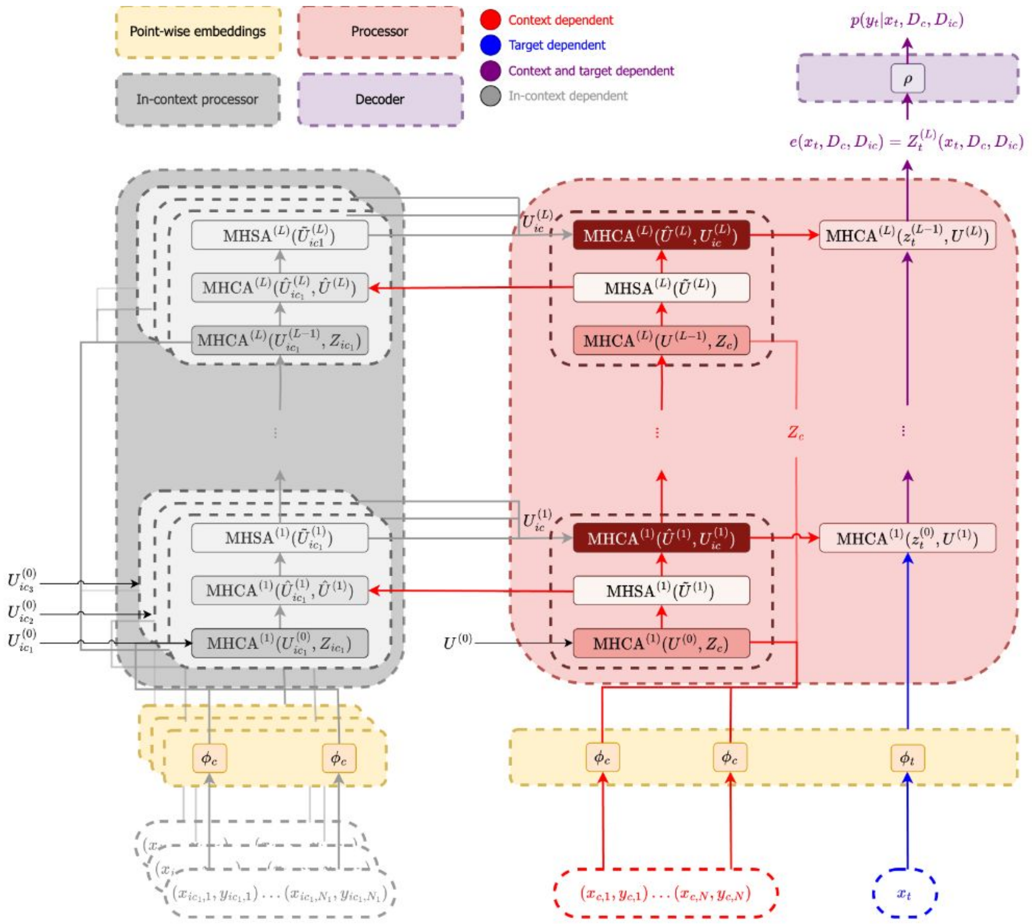
Create extra pseudo-tokens for each of the in-context datasets, U_{ic_i}.

At each layer we perform cross attention:

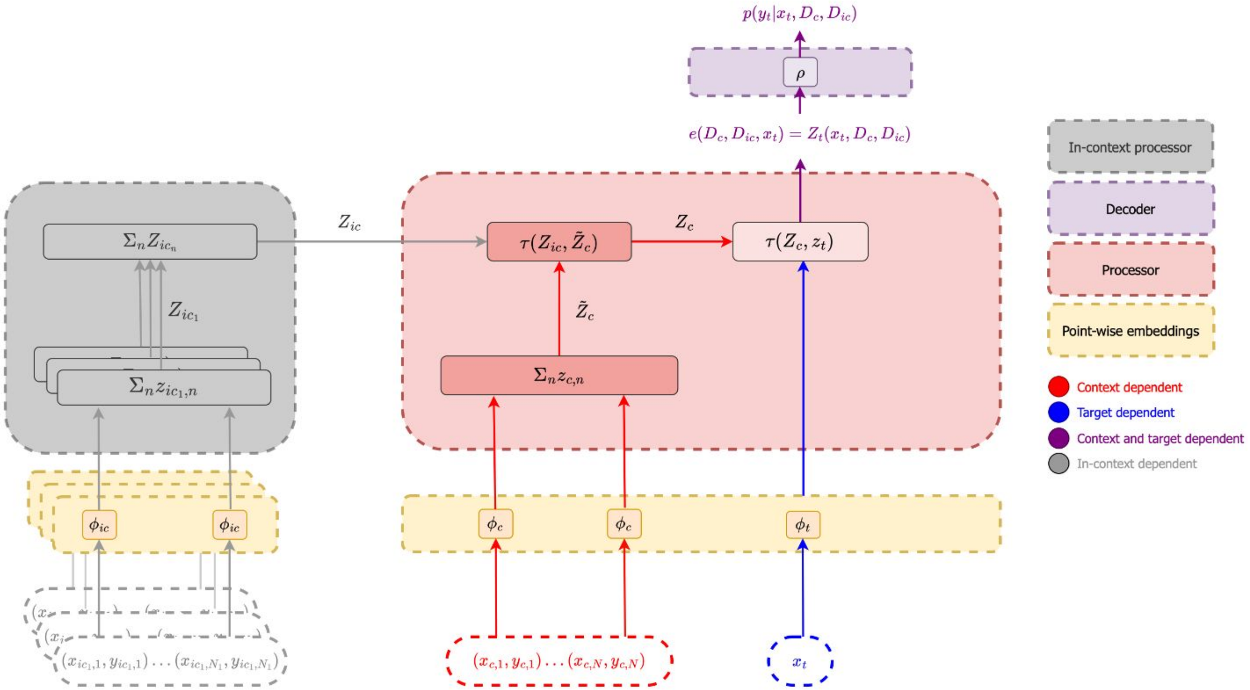
1. Between latents Z_c(or Z_{ic_i}) and pseudo tokens U (or U_{ic_j})
2. Between context (i.e. regular) pseudo tokens U and concatenated in-context pseudo tokens U_{ic}
3. Between target latents z_t and pseudo tokens U (as in regular PT-TNPs)

Basically, the in-context processor maintains in-context latents Z_{ic_i} and in-context pseudo tokens U_{ic} which are used to modulate the main processor's pseudo tokens U

$$\mathcal{O}\left(MN_c + MN_t + \sum_{j=1}^{N_{ic}} \left(M_{ic}N_{ic,j} + MM_{ic}\right)\right)$$



(Simpler model) ICICL Conditional Neural Processes (ICICL-CNPs)



4. Experiments

Experiments

Want to check:

1. Is ICICL still good without any in-context datasets?

2. Does ICICL get better with more in-context datasets (as we’d hope/expect)?

3. How does ICICL fare with OOD contexts?

4. How does ICICL fare with misspecified in-context datasets?

Synthetic 1D Regression

Draw samples from a Gaussian process with:

- Kernel sampled from {RBF, periodic}

Synthetic 1D Regression

Draw samples from a Gaussian process with:

- Kernel sampled from {RBF, periodic}
- Bandwidth/period sampled from $\log l \sim \mathcal{U}_{[\log 0.25, \log 4]}$

Synthetic 1D Regression

Draw samples from a Gaussian process with:

- Kernel sampled from {RBF, periodic}
- Bandwidth/period sampled from $\log l \sim \mathcal{U}_{[\log 0.25, \log 4]}$
- $N_c \sim \mathcal{U}_{\{1,\ldots,64\}}$ context points at $x_c \sim \mathcal{U}_{\{-2,2\}}$

Synthetic 1D Regression

Draw samples from a Gaussian process with:

- Kernel sampled from {RBF, periodic}
- Bandwidth/period sampled from $\log l \sim \mathcal{U}_{[\log 0.25, \log 4]}$
- $N_c \sim \mathcal{U}_{\{1,\ldots,64\}}$ context points at $x_c \sim \mathcal{U}_{\{-2,2\}}$
- $N_t = 128$ target points at $x_t \sim \mathcal{U}_{[-4,4]}$

Synthetic 1D Regression

Draw samples from a Gaussian process with:

- Kernel sampled from {RBF, periodic}
- Bandwidth/period sampled from $\log l \sim \mathcal{U}_{[\log 0.25, \log 4]}$
- $N_c \sim \mathcal{U}_{\{1,\ldots,64\}}$ context points at $x_c \sim \mathcal{U}_{\{-2,2\}}$
- $N_t = 128$ target points at $x_t \sim \mathcal{U}_{[-4,4]}$
- $N_{ic} \sim \mathcal{U}_{\{0,\ldots,5\}}$ in-context sets, each with $N_{ic,j} \sim \mathcal{U}_{\{64,\ldots,128\}}$ points at $x_{ic} \sim \mathcal{U}_{[-4,4]}$

Synthetic 1D Regression

Draw samples from a Gaussian process with:

- Kernel sampled from {RBF, periodic}
- Bandwidth/period sampled from $\log l \sim \mathcal{U}_{[\log 0.25, \log 4]}$
- $N_c \sim \mathcal{U}_{\{1,\ldots,64\}}$ context points at $x_c \sim \mathcal{U}_{\{-2,2\}}$
- $N_t = 128$ target points at $x_t \sim \mathcal{U}_{[-4,4]}$
- $N_{ic} \sim \mathcal{U}_{\{0,\ldots,5\}}$ in-context sets, each with $N_{ic,j} \sim \mathcal{U}_{\{64,\ldots,128\}}$ points at $x_{ic} \sim \mathcal{U}_{[-4,4]}$
- Observation noise 0.2

Synthetic 1D Regression

Draw samples from a Gaussian process with:

- Kernel sampled from {RBF, periodic}
- Bandwidth/period sampled from $\log l \sim \mathcal{U}_{[\log 0.25, \log 4]}$
- $N_c \sim \mathcal{U}_{\{1,\ldots,64\}}$ context points at $x_c \sim \mathcal{U}_{\{-2,2\}}$
- $N_t = 128$ target points at $x_t \sim \mathcal{U}_{[-4,4]}$
- $N_{ic} \sim \mathcal{U}_{\{0,\ldots,5\}}$ in-context sets, each with $N_{ic,j} \sim \mathcal{U}_{\{64,\ldots,128\}}$ points at $x_{ic} \sim \mathcal{U}_{[-4,4]}$
- Observation noise 0.2

(In-context datasets are drawn using the exact same kernel as the context-dataset.)

Synthetic 1D Regression

Model	Log lik. (↑)
CNP	-0.812 ± 0.005
PT-TNP	-0.598 ± 0.005
ICICL-TNP (0)	-0.607 ± 0.005
ICICL-TNP (1)	-0.499 ± 0.005
ICICL-TNP (2)	-0.474 ± 0.005
ICICL-TNP (3)	-0.469 ± 0.005
ICICL-TNP (4)	-0.467 ± 0.005
ICICL-TNP (5)	$\mathbf{-0.466 \pm 0.005}$

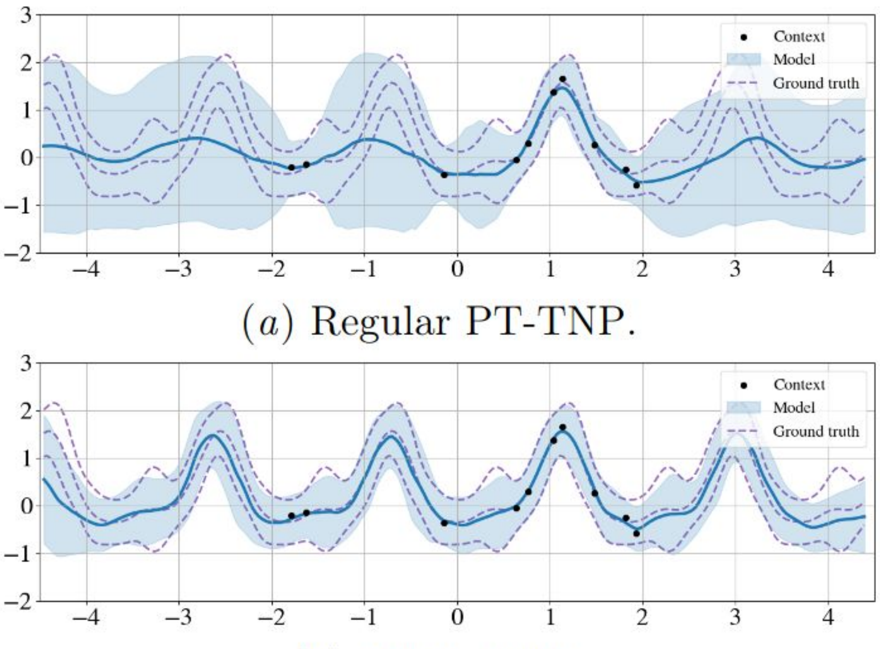
Table 1: Comparison of the predictive performance (in terms of test log likelihood) between the CNP, PT-TNP, and the ICICL-TNP with varying number of in-context datasets (indicated within brackets).

(CNP just uses MLPs instead of transformers.)

Synthetic 1D Regression

Model	Log lik. (↑)
CNP	-0.812 ± 0.005
PT-TNP	-0.598 ± 0.005
ICICL-TNP (0)	-0.607 ± 0.005
ICICL-TNP (1)	-0.499 ± 0.005
ICICL-TNP (2)	-0.474 ± 0.005
ICICL-TNP (3)	-0.469 ± 0.005
ICICL-TNP (4)	-0.467 ± 0.005
ICICL-TNP (5)	$\mathbf{-0.466 \pm 0.005}$

Table 1: Comparison of the predictive performance (in terms of test log likelihood) between the CNP, PT-TNP, and the ICICL-TNP with varying number of in-context datasets (indicated within brackets).



(a) Regular PT-TNP.

(b) ICICL-TNP.

Figure 2: The difference between the predictive distributions of the regular PT-TNP and the ICICL-TNP when conditioning on three in-context datasets with 128 datapoints (not shown here).

Synthetic 1D Regression - OOD

Test on datasets with bandwidth/period sampled as

$$\log l \sim \mathcal{U}_{[\log 0.1, \log 0.25] \cup [\log 4, \log 10]}$$

Model	Log lik. (↑)
CNP	-0.880 ± 0.006
PT-TNP	-0.798 ± 0.007
ICICL-TNP (0)	-0.783 ± 0.007
ICICL-TNP (1)	-0.721 ± 0.006
ICICL-TNP (2)	-0.702 ± 0.006
ICICL-TNP (3)	$\mathbf{-0.700 \pm 0.006}$

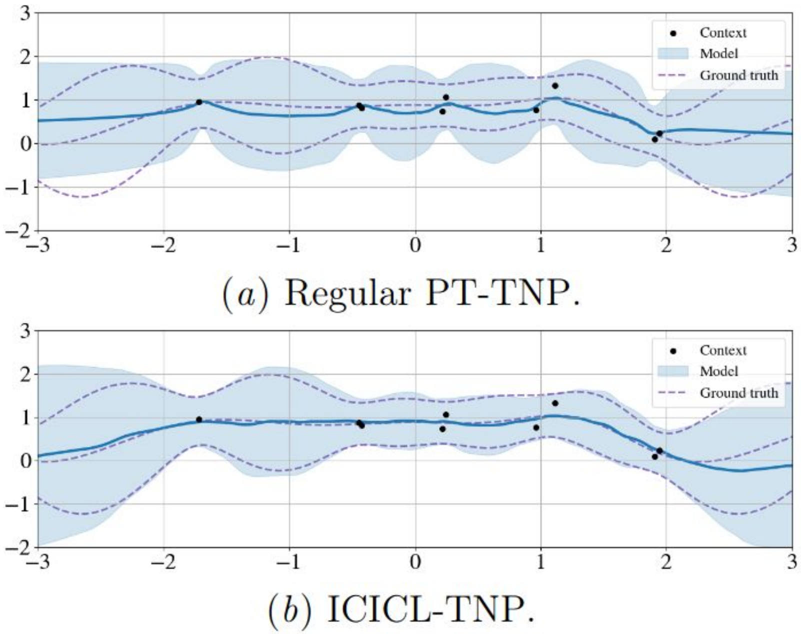
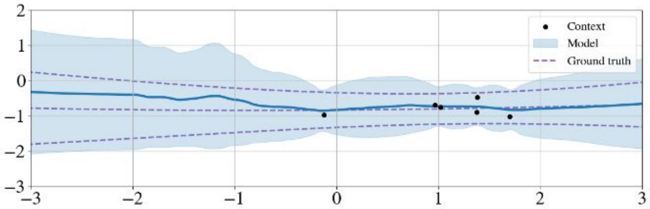
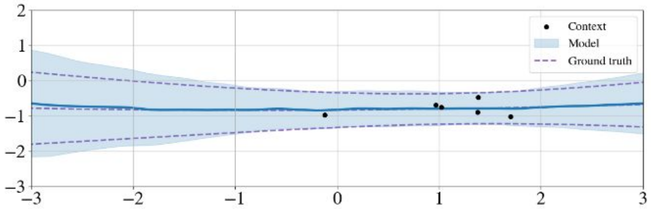


Figure 3: The difference between the predictive distributions when tested OOD of the regular PT-TNP and the ICICL-TNP when conditioning on three in-context datasets. The context datapoints come from a GP with a periodic kernel with $\ell = 6.08$.

Synthetic 1D Regression - OOD



(a) PT-TNP, RBF $\ell = 5.19$



(b) ICICL-TNP, RBF $\ell = 5.19$

Figure 13: A comparison between the predictive distribution when tested OOD of the PT-TNP (left) and the ICICL-TNP (right). In the top row the samples come from a GP with RBF kernel and $\ell = 5.19$. In the bottom row the samples come from a GP with periodic kernel and $\ell = 0.24$. The ICICL-TNP is conditioned on three in-context datasets.

Synthetic 1D Regression - OOD

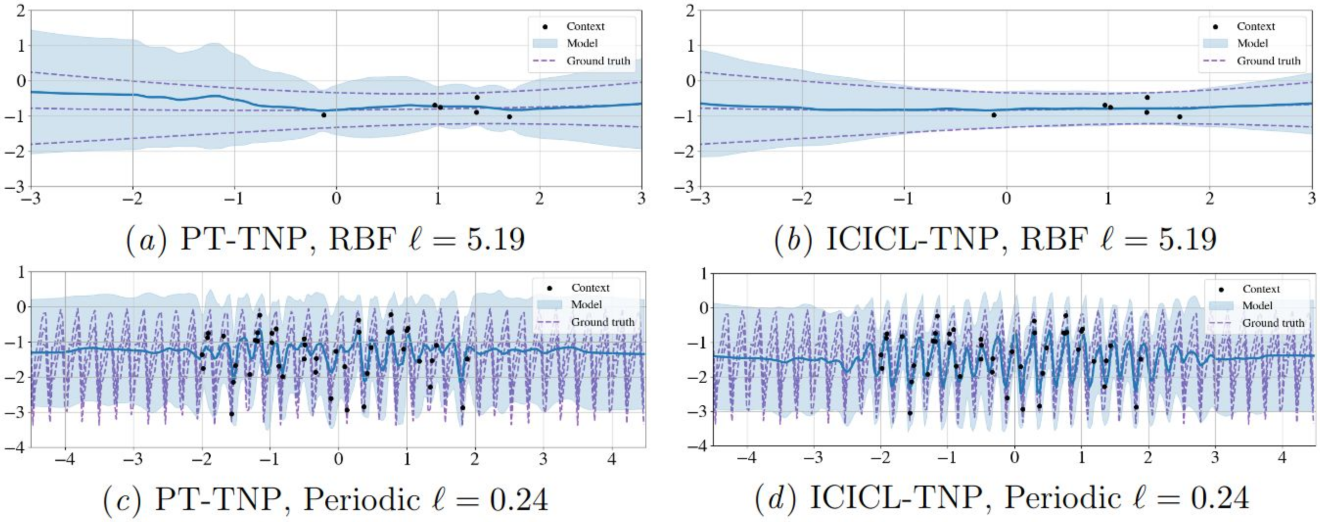
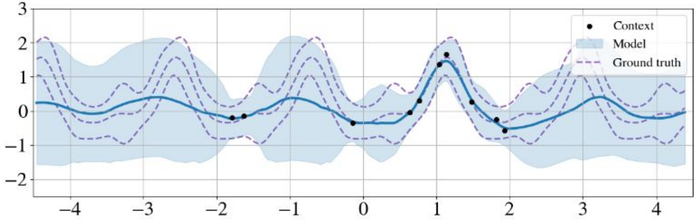
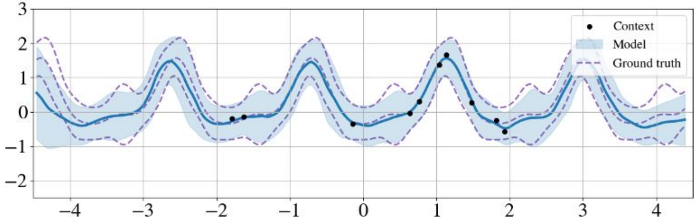


Figure 13: A comparison between the predictive distribution when tested OOD of the PT-TNP (left) and the ICICL-TNP (right). In the top row the samples come from a GP with RBF kernel and $\ell = 5.19$. In the bottom row the samples come from a GP with periodic kernel and $\ell = 0.24$. The ICICL-TNP is conditioned on three in-context datasets.

Synthetic 1D Regression - Incorrect In-context Conditioning

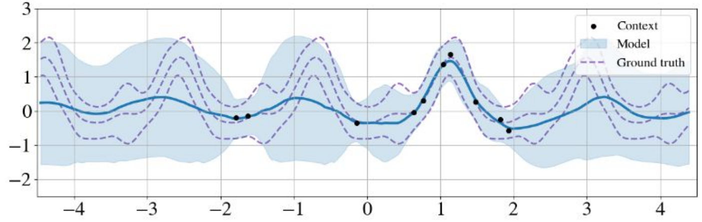


(a) No in-context datasets

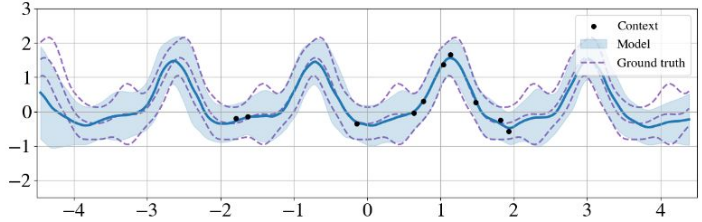


(b) Periodic kernel, $\ell = 1.85$

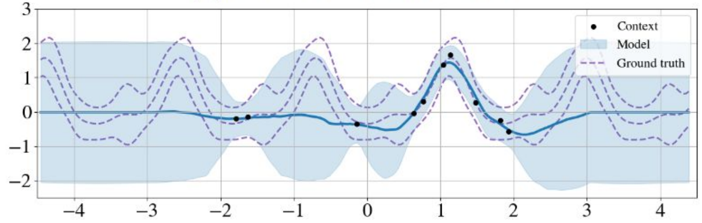
Synthetic 1D Regression - Incorrect In-context Conditioning



(a) No in-context datasets

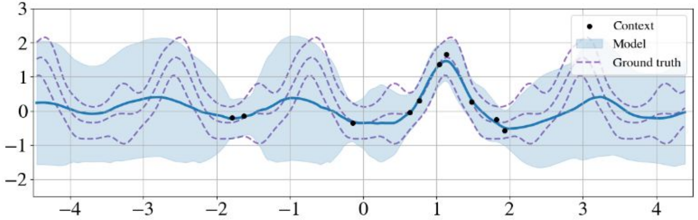


(b) Periodic kernel, $\ell = 1.85$

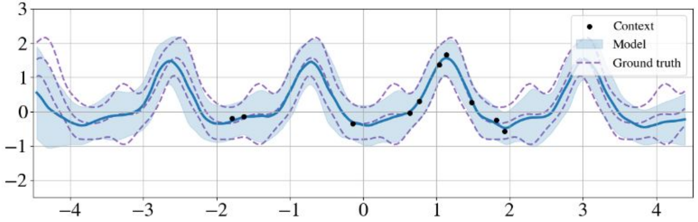


(c) RBF kernel, $\ell = 0.52$

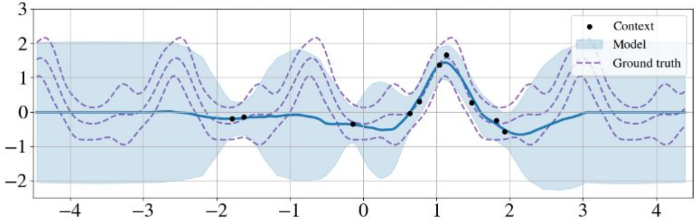
Synthetic 1D Regression - Incorrect In-context Conditioning



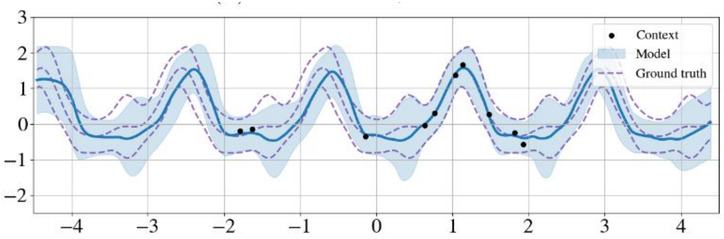
(a) No in-context datasets



(b) Periodic kernel, $\ell = 1.85$

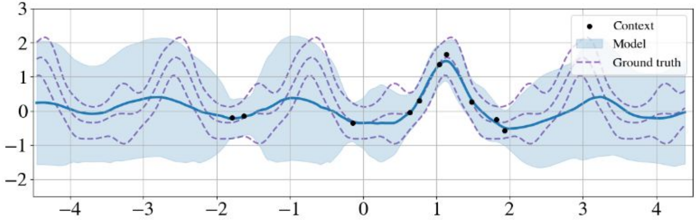


(c) RBF kernel, $\ell = 0.52$

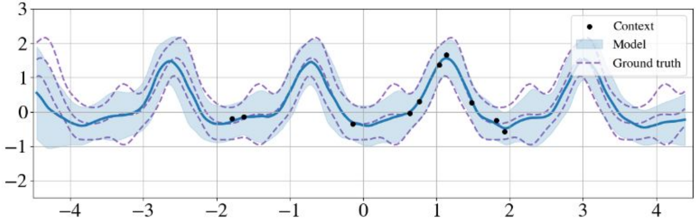


(d) Periodic kernel, $\ell = 1.76$

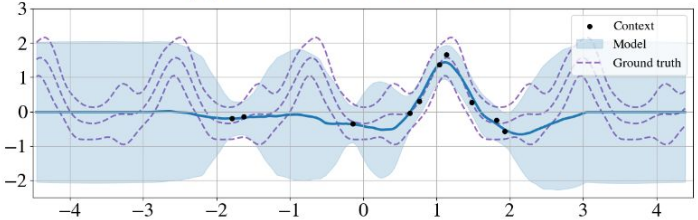
Synthetic 1D Regression - Incorrect In-context Conditioning



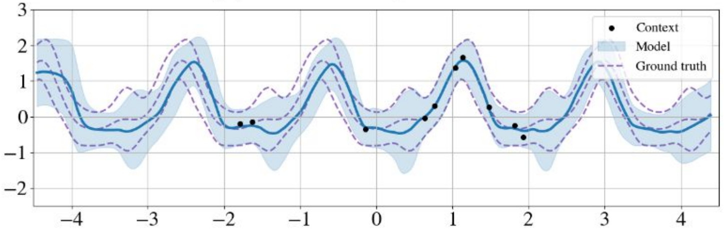
(a) No in-context datasets



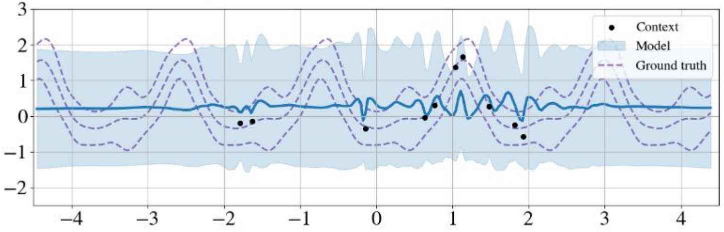
(b) Periodic kernel, $\ell = 1.85$



(c) RBF kernel, $\ell = 0.52$

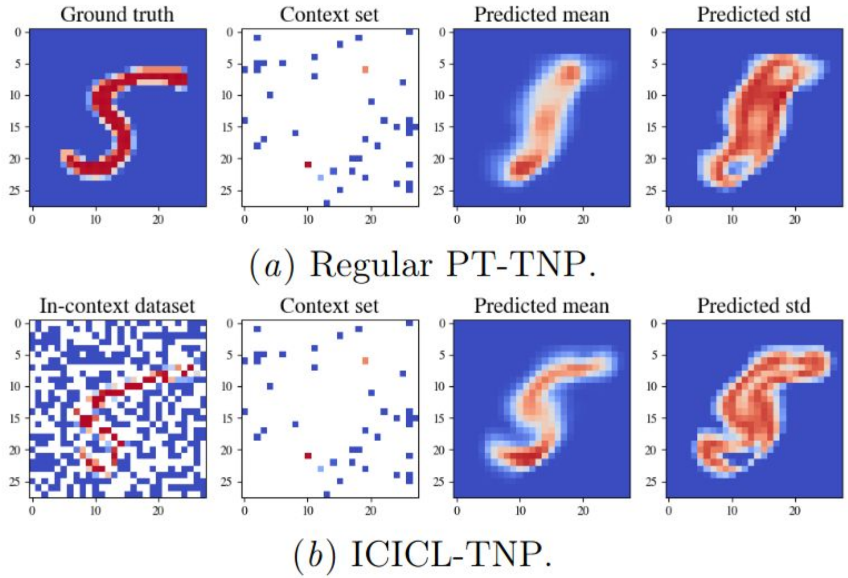


(d) Periodic kernel, $\ell = 1.76$



(e) Periodic kernel, $\ell = 0.28$

MNIST Image Completion



- $N = 784$ 2D pixel locations $x \in \mathbb{R}^2$
- Pixel values $y \in \mathbb{R}$

Figure 5: A comparison between the predictive distribution of the ICICL-TNP and the regular PT-TNP when conditioning on the context and in-context dataset shown.

MNIST Image Completion

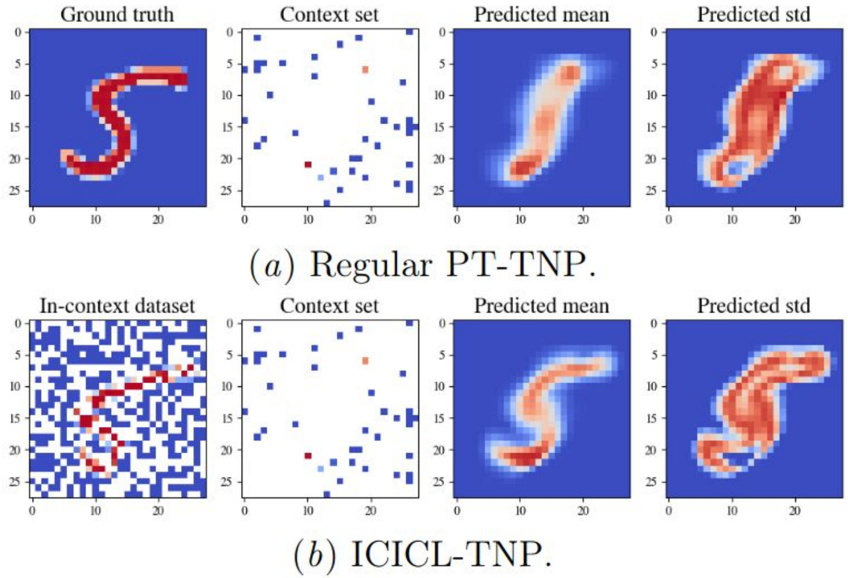


Figure 5: A comparison between the predictive distribution of the ICICL-TNP and the regular PT-TNP when conditioning on the context and in-context dataset shown.

- $N = 784$ 2D pixel locations $x \in \mathbb{R}^2$
- Pixel values $y \in \mathbb{R}$
- $N_c \sim \mathcal{U}_{\{N/100,\ldots,N/5\}}$ context points

MNIST Image Completion

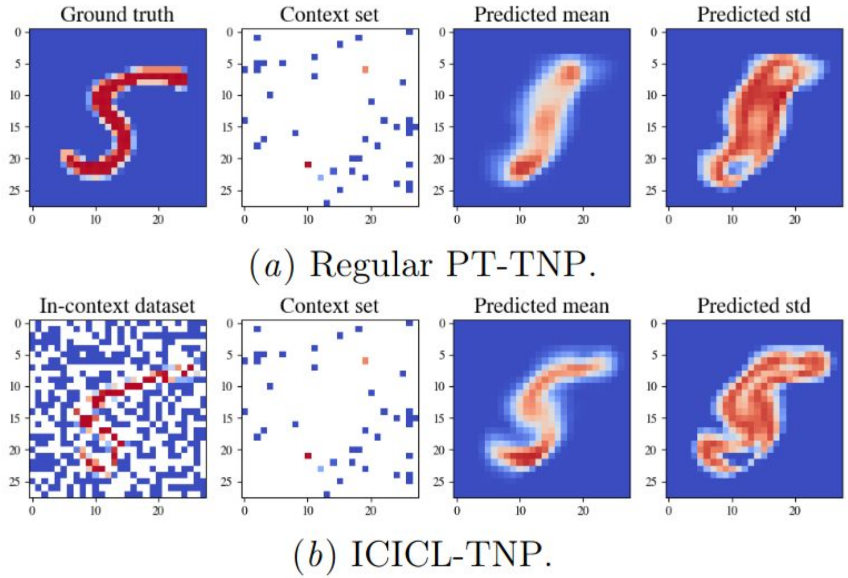


Figure 5: A comparison between the predictive distribution of the ICICL-TNP and the regular PT-TNP when conditioning on the context and in-context dataset shown.

- $N = 784$ 2D pixel locations $x \in \mathbb{R}^2$
- Pixel values $y \in \mathbb{R}$
- $N_c \sim \mathcal{U}_{\{N/100,\ldots,N/5\}}$ context points
- $N_{ic} \sim \mathcal{U}_{\{0,\ldots,3\}}$ in-context sets with $N_{ic,j} \sim \mathcal{U}_{\{N/100,\ldots,N/2\}}$

MNIST Image Completion

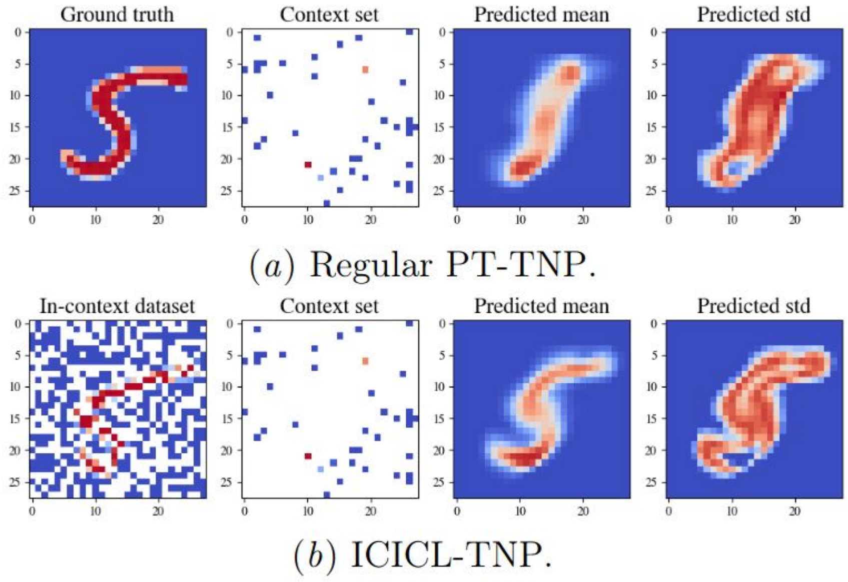


Figure 5: A comparison between the predictive distribution of the ICICL-TNP and the regular PT-TNP when conditioning on the context and in-context dataset shown.

- $N = 784$ 2D pixel locations $x \in \mathbb{R}^2$
- Pixel values $y \in \mathbb{R}$
- $N_c \sim \mathcal{U}_{\{N/100,\ldots,N/5\}}$ context points
- $N_{ic} \sim \mathcal{U}_{\{0,\ldots,3\}}$ in-context sets with $N_{ic,j} \sim \mathcal{U}_{\{N/100,\ldots,N/2\}}$
- $N_t = N - N_c$

MNIST Image Completion

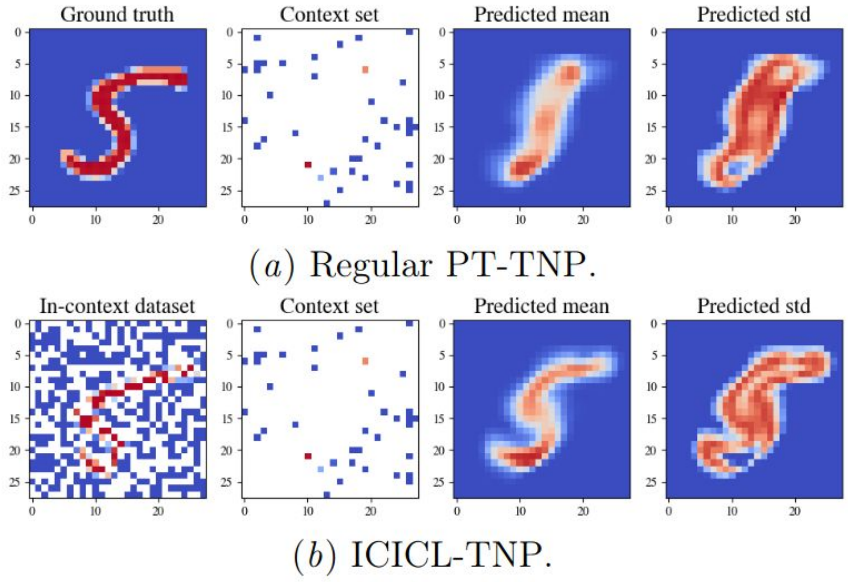


Figure 5: A comparison between the predictive distribution of the ICICL-TNP and the regular PT-TNP when conditioning on the context and in-context dataset shown.

- $N = 784$ 2D pixel locations $x \in \mathbb{R}^2$
- Pixel values $y \in \mathbb{R}$
- $N_c \sim \mathcal{U}_{\{N/100,\ldots,N/5\}}$ context points
- $N_{ic} \sim \mathcal{U}_{\{0,\ldots,3\}}$ in-context sets with $N_{ic,j} \sim \mathcal{U}_{\{N/100,\ldots,N/2\}}$
- $N_t = N - N_c$

(In-context datasets are drawn using the same class label as the context-dataset.)

MNIST Image Completion

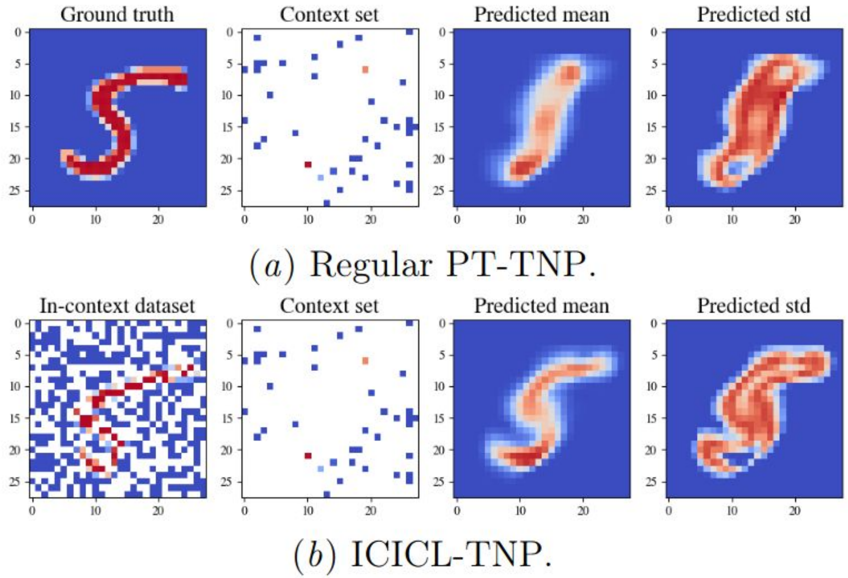


Figure 5: A comparison between the predictive distribution of the ICICL-TNP and the regular PT-TNP when conditioning on the context and in-context dataset shown.

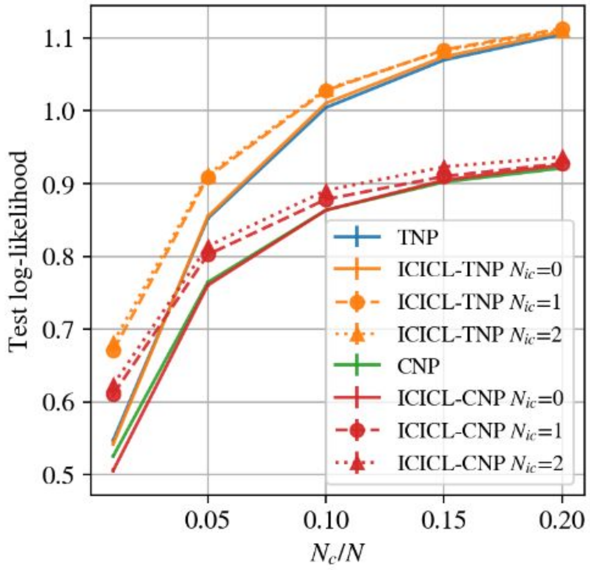
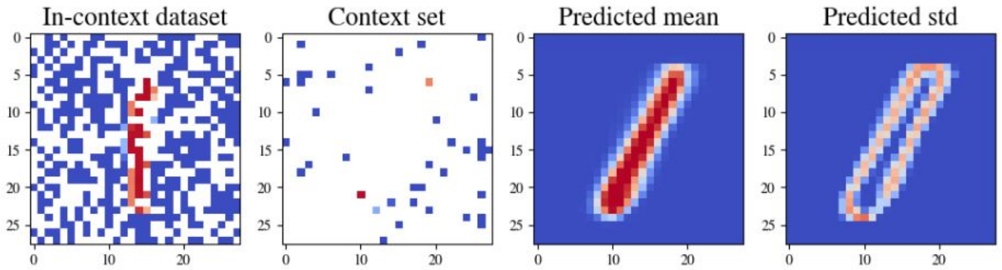


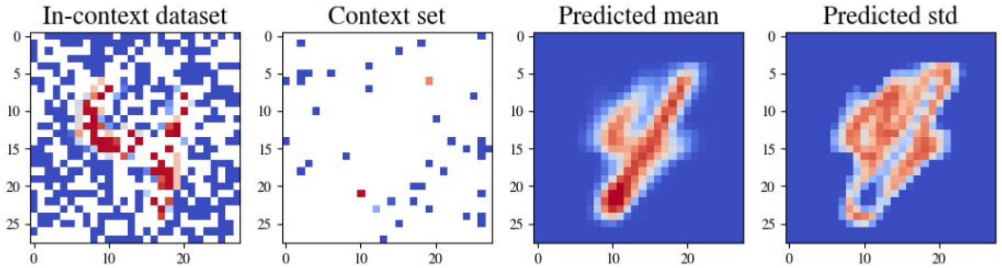
Figure 4: A comparison between the predictive performance of the ICICL-TNP, regular PT-TNP, ICICL-CNP and CNP as the proportion of context datapoints N_c/N varies in the MNIST inpainting experiment.

MNIST Image Completion

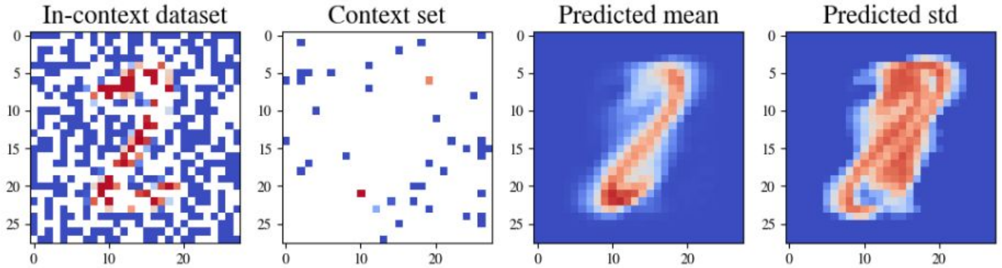
(Notice high uncertainty for 4 and 2, where the context set is less compatible with the label of the in-context dataset.)



(b) In-context dataset label: 1



(c) In-context dataset label: 4



(d) In-context dataset label: 2

Environmental Data

Regression on spatio-temporal data for surface air temperature.

In-context datasets are obtained from the same location but at different times.

Model	**Test Log-Likelihood**
PT-TNP	1.15 ± 0.01
ICICL-TNP (0)	1.15 ± 0.01
ICICL-TNP (1)	1.18 ± 0.01
ICICL-TNP (2)	1.19 ± 0.01

Table 3: Comparison of the test log-likelihood on the environmental data for the PT-TNP and ICICL-TNP with varying number of in-context datasets (indicated within brackets).

Environmental Data

Another baseline: also allow PT-TNP to train on the in-context sets (but just stick that data in with the main context).

Model	Test Log-Likelihood
PT-TNP	1.15 ± 0.01
PT-TNP-merged (0)	1.14 ± 0.01
PT-TNP-merged (1)	1.15 ± 0.01
PT-TNP-merged (2)	1.14 ± 0.01
ICICL-TNP (0)	1.15 ± 0.01
ICICL-TNP (1)	1.18 ± 0.01
ICICL-TNP (2)	1.19 ± 0.01

Table 5: Comparison of the test log-likelihood on the environmental data for the PT-TNP and ICICL-TNP with varying number of in-context datasets (indicated within brackets). As an additional baseline, we also consider the PT-TNP-merged where the in-context datasets are merged into the context datasets.

(This doesn't help.)

Weaknesses

- More experiments?
  - Larger-scaled
  - (Though the experiments in this paper seem to match standard experiments in other NP papers…)

- Include the PT-TNP-merged baseline in all experiments.
  - (Or some other method to let PT-TNP use the in-context data, even if badly.)

- ICICL relies on identifying in-context datasets drawn from the same stochastic process as the context dataset. This is often not possible.

References

Ashman, Matthew, et al. "In-Context In-Context Learning with Transformer Neural Processes." arXiv preprint arXiv:2406.13493 (2024).

Nguyen, Tung, and Aditya Grover. "Transformer neural processes: Uncertainty-aware meta learning via sequence modeling." arXiv preprint arXiv:2207.04179 (2022).

Foong, Andrew, et al. "Meta-learning stationary stochastic process prediction with convolutional neural processes." Advances in Neural Information Processing Systems 33 (2020): 8284-8295.

Lee, Juho, et al. "Set transformer: A framework for attention-based permutation-invariant neural networks." International conference on machine learning. PMLR, 2019.

Kim, Hyunjik, et al. "Attentive neural processes." arXiv preprint arXiv:1901.05761 (2019).

Garnelo, Marta, et al. "Conditional neural processes." International conference on machine learning. PMLR, 2018.

Appendix: Attentive Neural Process (ANP)