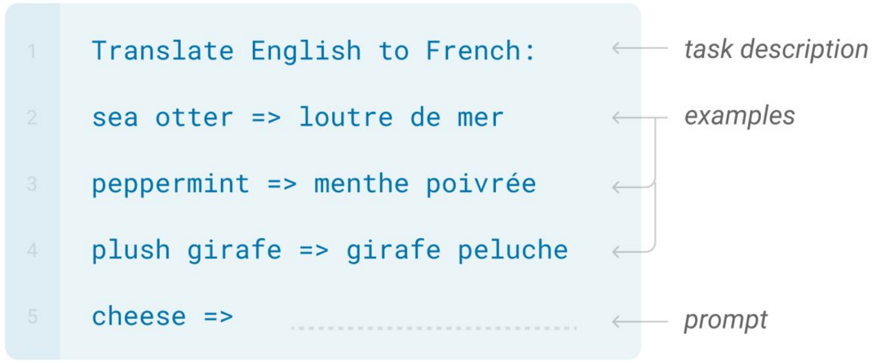
Transformer Neural Processes: Uncertainty-Aware Meta Learning Via Sequence Modeling

Tung Nguyen, Aditya Grover

Motivation

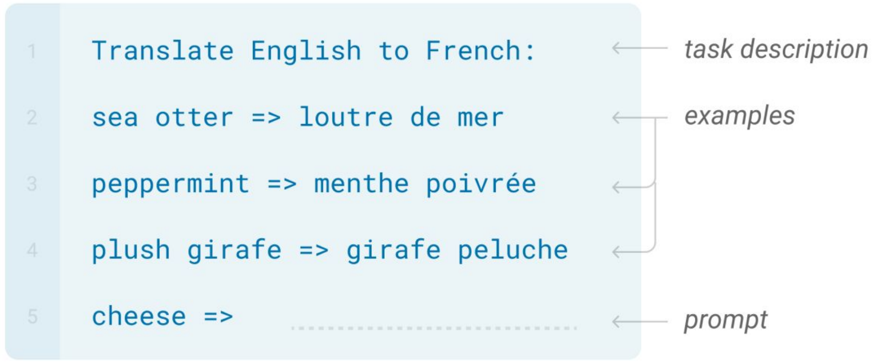
❑ Transformers have shown excellent capabilities in few-shot/meta learning



```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

Motivation

- ❑ Transformers have shown excellent capabilities in few-shot/meta learning

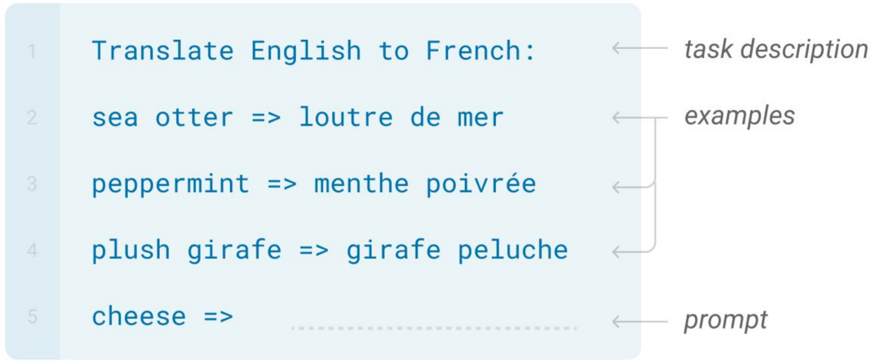


```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

- ❑ Most previous works only focus on accuracy

Motivation

- ❑ Transformers have shown excellent capabilities in few-shot/meta learning

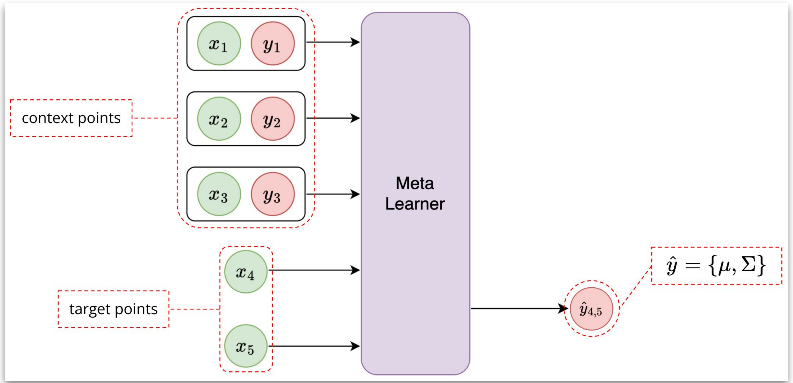


```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

- ❑ Most previous works only focus on accuracy
- ❑ Question: How can we represent uncertainty in meta learning with transformers?

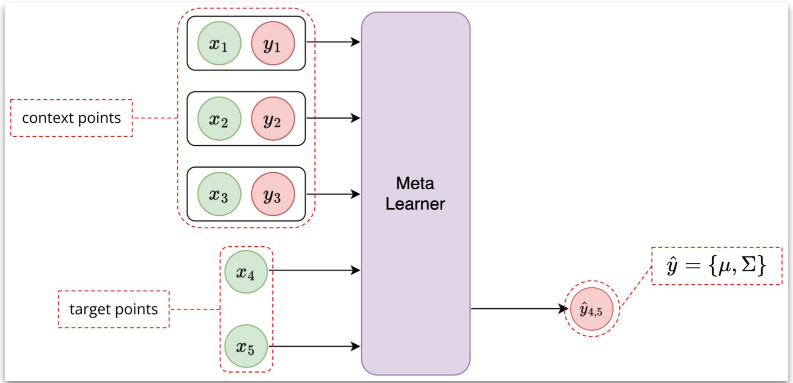
Uncertainty-aware Meta Learning

- For each unseen task, the model observes a small set of context points $\{x_i, y_i\}_{i=1}^{m}$ and makes predictions for a set of target points $\{x_j\}_{j=m+1}^{N}$.



Uncertainty-aware Meta Learning

- For each unseen task, the model observes a small set of context points $\{x_i, y_i\}_{i=1}^{m}$ and makes predictions for a set of target points $\{x_j\}_{j=m+1}^{N}$.



- Neural Processes formalize as learning a conditional distribution of the target points, given the context points and the target inputs:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_{1:N}, y_{1:N}, m}\left[\log p_\theta\left(y_{m+1:N} \mid x_{1:N}, y_{1:m}\right)\right]$$

Meta-learning as sequence modeling

- ❏ Most Neural Processes parameterize the objective by a latent variable model.
- ❏ We propose a novel autoregressive factorization

$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}_{x_{1:N}, y_{1:N}, m}\left[\log p_\theta\left(y_{m+1:N} \mid x_{1:N}, y_{1:m}\right)\right] \\ &= \mathbb{E}_{x_{1:N}, y_{1:N}, m}\left[\sum_{i=m+1}^{N} \log p_\theta\left(y_i \mid x_{1:i}, y_{1:i-1}\right)\right]\end{aligned}$$

Meta-learning as sequence modeling

- ❑ Most Neural Processes parameterize the objective by a latent variable model.
- ❑ We propose a novel autoregressive factorization

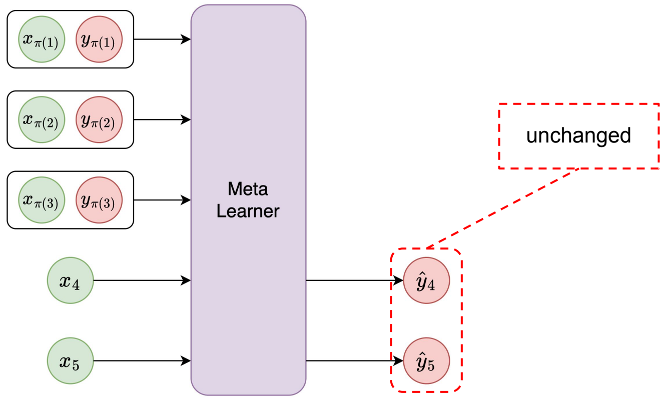
$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}_{x_{1:N}, y_{1:N}, m}\left[\log p_\theta\left(y_{m+1:N} \mid x_{1:N}, y_{1:m}\right)\right] \\ &= \mathbb{E}_{x_{1:N}, y_{1:N}, m}\left[\sum_{i=m+1}^{N} \log p_\theta\left(y_i \mid x_{1:i}, y_{1:i-1}\right)\right]\end{aligned}$$

- ❑ In principle, any sequence model can optimize this objective, but what architecture is suitable for meta learning?

Meta-learning desiderata

Two important properties of meta learning

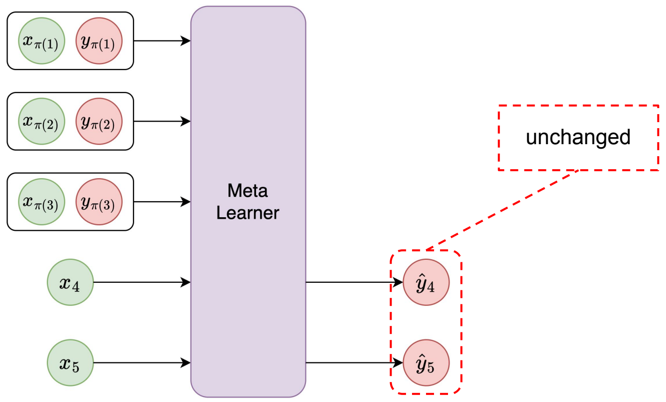
Property 1. *Context invariance.*
In other words, the model's predictions do not depend on the permutation of the context points.



Meta-learning desiderata

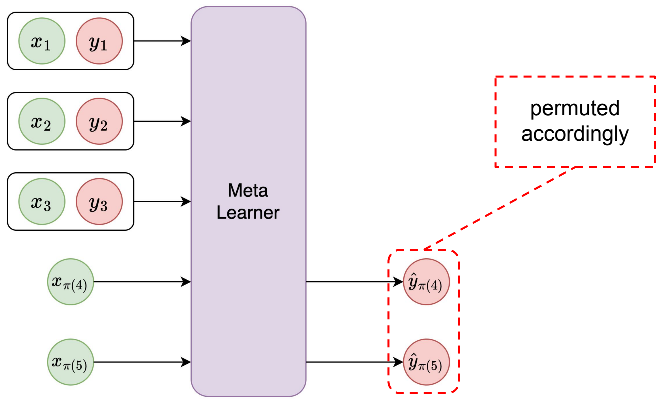
Two important properties of meta learning

Property 1. ***Context invariance.***
In other words, the model's predictions do not depend on the permutation of the context points.



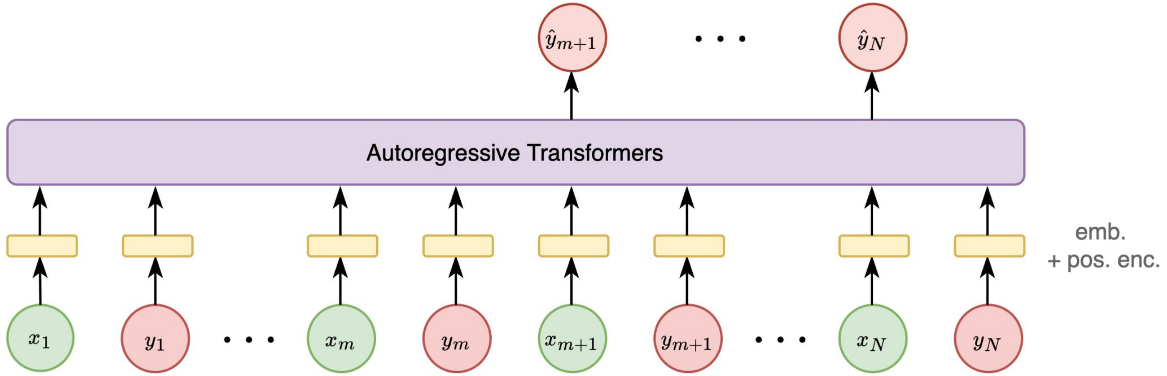
$x_{\pi(1)}$ $y_{\pi(1)}$
$x_{\pi(2)}$ $y_{\pi(2)}$
$x_{\pi(3)}$ $y_{\pi(3)}$
x_4
x_5
Meta Learner
unchanged
$\hat{y}_4$
$\hat{y}_5$

Property 2. ***Target equivariance.***
Whenever we permute the target inputs, the predictions are permuted accordingly.



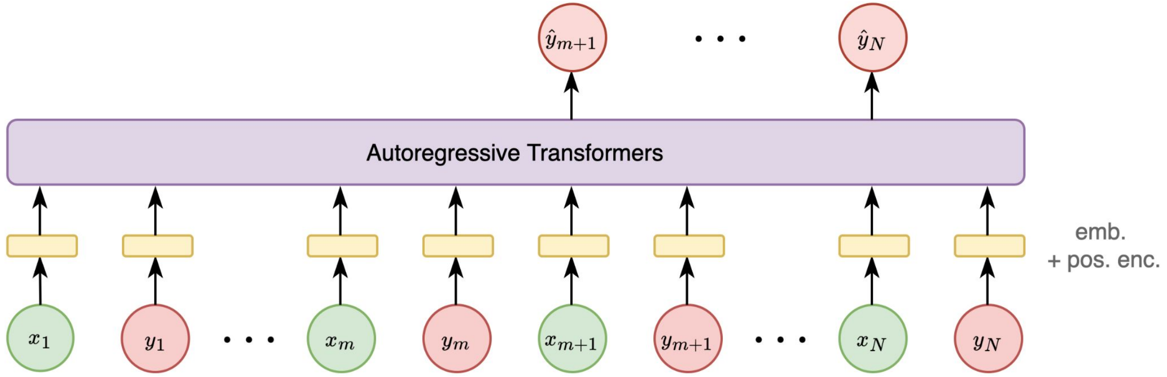
First attempt: Vanilla Transformers

❑ Feed all tokens to the model and employ a causal mask



First attempt: Vanilla Transformers

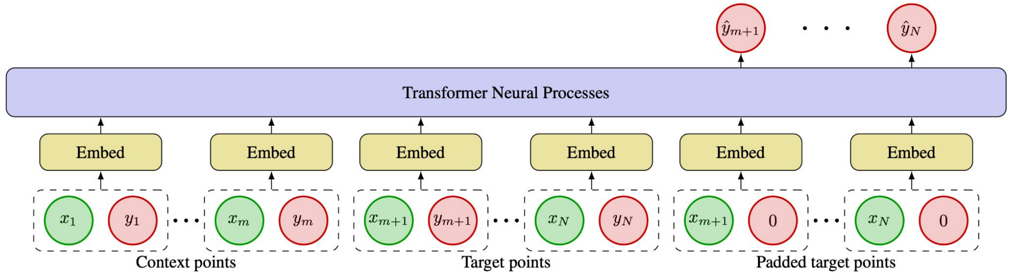
- Feed all tokens to the model and employ a causal mask



- Positional embedding is important to associate x and y, but breaks both conditions

Transformer Neural Processes (TNPs)

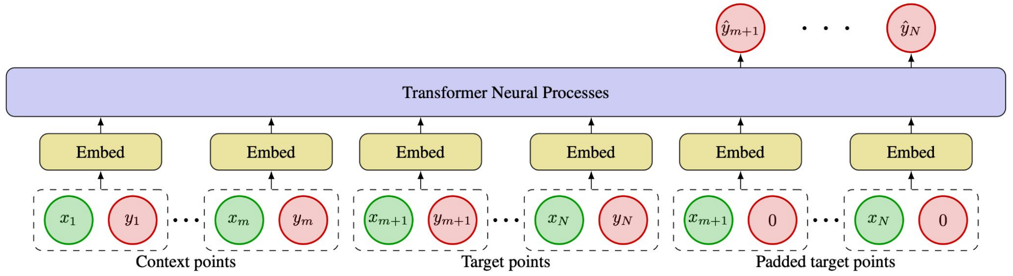
❑ Solution: concatenate (x_i, y_i) to form a token, removing positional embedding.



$$\mathcal{L}(\theta) = \mathbb{E}\left[\log p_\theta\left(y_{m+1:N} \mid x_{1:N}, y_{1:m}\right)\right]$$
$$= \mathbb{E}\left[\sum_{i=m+1}^{N} \log p_\theta\left(y_i \mid x_{1:i}, y_{1:i-1}\right)\right]$$

Transformer Neural Processes (TNPs)

- Solution: concatenate (x_i, y_i) to form a token, removing positional embedding.

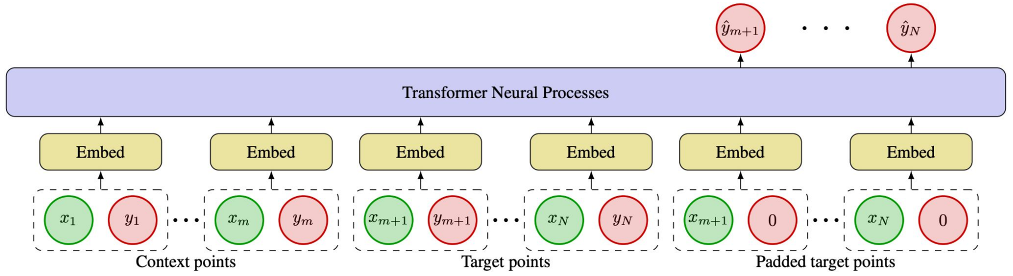


$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}\left[\log p_\theta\left(y_{m+1:N} \mid x_{1:N}, y_{1:m}\right)\right] \\ &= \mathbb{E}\left[\sum_{i=m+1}^{N} \log p_\theta\left(y_i \mid x_{1:i}, y_{1:i-1}\right)\right]\end{aligned}$$

- TNPs employ a masking mechanism to preserve autoregressive ordering (TNP-A).

Transformer Neural Processes (TNPs)

- Solution: concatenate (x_i, y_i) to form a token, removing positional embedding.



$$\mathcal{L}(\theta) = \mathbb{E}\left[\log p_\theta\left(y_{m+1:N} \mid x_{1:N}, y_{1:m}\right)\right]$$
$$= \mathbb{E}\left[\sum_{i=m+1}^{N} \log p_\theta\left(y_i \mid x_{1:i}, y_{1:i-1}\right)\right]$$

- TNPs employ a masking mechanism to preserve autoregressive ordering (TNP-A).
- TNP-A is expressive, but is slow during inference due to the sequential prediction.

Variants of TNPs

We introduce TNP-D and TNP-ND to trade off expressivity and simplicity

Diagonal Transformer Neural Process (TNP-D)
Assume that the target points are conditionally independent given the context points:

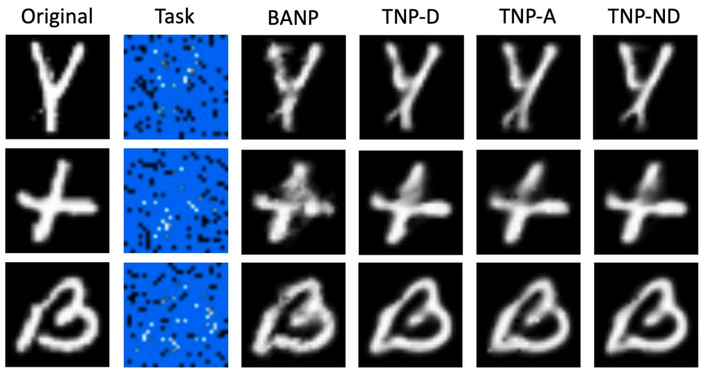
$$p_\theta\left(y_{m+1:N} \mid x_{1:N}, y_{1:m}\right) \\ = \prod_{i=m+1}^{N} p_\theta\left(y_i \mid x_i, x_{1:m}, y_{1:m}\right)$$

Non-Diagonal Transformer Neural Process (TNP-ND)
Predict the target points jointly using a multivariate Gaussian distribution with a non-diagonal covariance matrix:

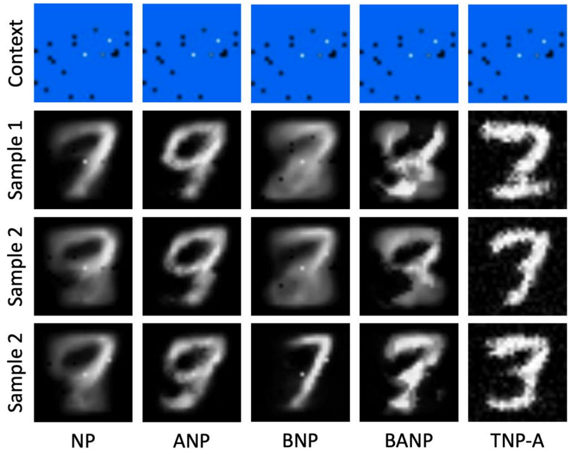
$$p_\theta\left(y_{m+1:N} \mid x_{1:N}, y_{1:m}\right) \\ = \mathcal{N}\left(y_{m+1:N} \mid \mu_\theta\left(x_{1:N}, y_{1:m}\right), \Sigma_\theta\left(x_{1:N}, y_{1:m}\right)\right)$$

Experiments: image completion

- ❏ Image completion: Complete an image given a few pixels
- ❏ Similar to 2-D regression, where input is coordinates, output is pixel intensities



Completed images produced by the best baseline and TNPs from 100 context points.



Samples produced by TNPs and the baselines given 20 context points.

Experiments: Results summary

- Benchmark tasks: 1-D regression, image completion, contextual bandits, Bayesian optimization
- TNPs achieve gains of 16%, 64%, 266%, and 567% over the best baseline on 4 benchmarks (in the above order).

Method	RBF	Matérn 5/2	Periodic
CNP	0.26 ± 0.02	0.04 ± 0.02	-1.40 ± 0.02
CANP	0.79 ± 0.00	0.62 ± 0.00	-7.61 ± 0.16
NP	0.27 ± 0.01	0.07 ± 0.01	-1.15 ± 0.04
ANP	0.81 ± 0.00	0.63 ± 0.00	-5.02 ± 0.21
BNP	0.38 ± 0.02	0.18 ± 0.02	$\mathbf{-0.96 \pm 0.02}$
BANP	0.82 ± 0.01	0.66 ± 0.00	-3.09 ± 0.14
TNP-D	1.39 ± 0.00	0.95 ± 0.01	-3.53 ± 0.37
TNP-A	$\mathbf{1.63 \pm 0.00}$	$\mathbf{1.21 \pm 0.00}$	-2.26 ± 0.17
TNP-ND	1.46 ± 0.00	1.02 ± 0.00	-4.13 ± 0.33

1-D regression

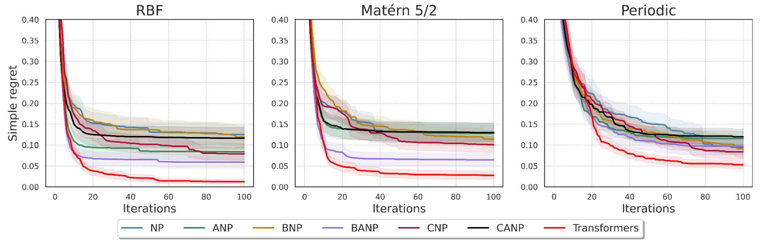
Method	$\delta = 0.7$	$\delta = 0.9$	$\delta = 0.95$	$\delta = 0.99$	$\delta = 0.995$	$\delta = 0.999$	Average
Uniform	100.00 ± 1.18	100.00 ± 3.03	100.00 ± 4.16	100.00 ± 7.52	100.00 ± 8.11	100.00 ± 7.96	100.00 ± 5.97
CNP	4.08 ± 0.29	8.14 ± 0.33	8.01 ± 0.40	26.78 ± 0.85	38.25 ± 1.01	93.17 ± 2.81	29.74 ± 30.85
CANP	8.08 ± 9.93	11.69 ± 11.96	24.49 ± 13.25	47.33 ± 20.49	49.59 ± 17.87	33.29 ± 5.05	29.08 ± 21.28
NP	1.56 ± 0.13	2.96 ± 0.28	4.24 ± 0.22	18.00 ± 0.42	25.53 ± 0.18	62.73 ± 1.49	19.17 ± 21.36
ANP	1.62 ± 0.16	4.05 ± 0.31	5.39 ± 0.50	19.57 ± 0.67	27.65 ± 0.95	73.36 ± 5.95	21.94 ± 24.92
BNP	62.51 ± 1.07	57.49 ± 2.13	58.22 ± 2.27	58.91 ± 3.77	62.50 ± 4.85	77.46 ± 6.18	62.85 ± 7.78
BANP	4.23 ± 16.58	12.42 ± 29.58	31.10 ± 36.10	52.59 ± 18.11	49.55 ± 14.52	45.45 ± 11.71	32.56 ± 29.43
TNP-D	$\mathbf{1.18 \pm 0.94}$	1.70 ± 0.41	2.55 ± 0.43	$\mathbf{3.57 \pm 1.22}$	$\mathbf{4.68 \pm 1.09}$	9.56 ± 0.44	$\mathbf{3.87 \pm 2.91}$
TNP-A	3.67 ± 4.88	4.04 ± 2.38	4.29 ± 2.36	5.79 ± 5.27	9.29 ± 7.62	$\mathbf{6.13 \pm 2.50}$	5.54 ± 4.98
TNP-ND	1.76 ± 0.61	$\mathbf{1.41 \pm 0.98}$	$\mathbf{1.61 \pm 1.65}$	4.98 ± 2.84	7.22 ± 3.28	13.66 ± 2.92	5.11 ± 4.94

Contextual bandits

Method	CelebA
CNP	2.15 ± 0.01
CANP	2.66 ± 0.01
NP	2.48 ± 0.02
ANP	2.90 ± 0.00
BNP	2.76 ± 0.01
BANP	3.09 ± 0.00
TNP-D	3.89 ± 0.01
TNP-A	$\mathbf{5.82 \pm 0.01}$
TNP-ND	5.48 ± 0.02

Method	EMNIST	
	Seen classes (0-9)	Unseen classes (10-46)
CNP	0.73 ± 0.00	0.49 ± 0.01
CANP	0.94 ± 0.01	0.82 ± 0.01
NP	0.79 ± 0.01	0.59 ± 0.01
ANP	0.98 ± 0.00	0.89 ± 0.00
BNP	0.88 ± 0.01	0.73 ± 0.01
BANP	1.01 ± 0.00	0.94 ± 0.00
TNP-D	1.46 ± 0.01	1.31 ± 0.00
TNP-A	$\mathbf{1.54 \pm 0.01}$	$\mathbf{1.41 \pm 0.01}$
TNP-ND	1.50 ± 0.00	1.31 ± 0.00

Image completion



Bayesian optimization

Paper Link

Tung: tungnd@cs.ucla.edu @tungnd_13