Variational Set Operator Networks: Uncertainty-Aware Meta-Learning via Probabilistic Neural Operators

Anonymous authors
Paper under double-blind review

Abstract

We introduce a probabilistic neural operator framework for learning conditional distributions over functions from sample observations. The proposed model, the Variational Set Operator Network (VSON), extends set-based operator learners by incorporating an amortised latent representation of the branch outputs that induces predictive distributions over function values conditioned on arbitrary sets of input–output pairs. Uncertainty is represented through a learned variational latent structure implemented with expressive normalising flows, allowing the model to capture non-Gaussian behaviour. The resulting operator produces smooth function predictions and coherent joint samples over target sets. VSON improves predictive uncertainty calibration across benchmarks while remaining competitive on accuracy, outperforming the baselines on the regression and optimisation tasks.

1 Introduction

Operator learning methods such as Deep Operator Networks Lu et al. (2021) and the Fourier Neural Operator Li et al. (2021) have demonstrated strong empirical performance for learning mappings between function spaces, but they are typically defined on fixed input discretisations and therefore do not naturally accommodate variable-sized context sets. This limitation matters in applications where sensors are expensive or where measurement locations must be chosen sequentially. In this paper we position neural operators within a meta-learning framework, learning a map from sparse observations of a function to itself while accounting for uncertainty. Formally, given a variable-sized context set of observed input-output pairs

$$\mathcal{C} = \{(x_i, y_i)\}_{i=1}^{N}, \tag{1}$$

which come from an unknown underlying function $f : \mathcal{X} \mapsto \mathcal{Y}$ itself belonging to a family of functions $\mathcal{F}$, we address how to learn a fast, generalisable conditional posterior $p(y \mid x, \mathcal{C})$. Rather than fitting a new posterior from scratch for each task, we learn a mapping from context sets to predictive posteriors using representative training instances from $\mathcal{Y}$. At test time the learned operator returns cheap posterior samples for arbitrary, variable-size context sets.

To realise this we build on permutation-invariant branch encoders and introduce the *Variational Set Operator Network* (VSON): a variational operator network whose branch is explicitly set-based. The training objective balances latent regularisation and calibration. We benchmark VSON against various Neural Process baselines on 1D regression, image completion and Bayesian optimisation (BO) tasks. VSON consistently improves uncertainty calibration, and matches or exceeds the baselines on predictive accuracy in the regression and BO settings, while trading a small amount of accuracy for substantially better calibration on image completion.

1.1 Related work

Neural Processes and amortised learning Neural Processes (NPs) and their attention and transformer extensions (ANPs, TNPs) provide a direct meta-learning approach to conditional function modelling by mapping context sets to predictive distributions Garnelo et al. (2018b); Kim et al. (2019); Nguyen & Grover (2023); Lee et al. (2020). TNPs in particular recast the conditional prediction problem as a sequence

modelling task using transformer blocks and masking strategies, improving likelihood-based training and conditional fidelity. Our work adopts the same meta-learning objective but instantiates this objective in the operator-learning paradigm. There is also an overlap with the amortised sequential design literature Huang et al. (2025); Müller et al. (2023); Li et al. (2025), particularly with respect to the BO experiment, as well as more general amortised Bayesian inference models Müller et al. (2024); Chang et al. (2025). While these methods typically treat inference as a sequence-to-sequence task, VSON reformulates the problem as one of operator learning, enabling resolution-independent function estimation with non-Gaussian uncertainty.

Uncertainty-aware neural operators Uncertainty-aware neural operators have attracted attention as a natural progression of neural operators for modelling primarily physical systems. VB-DeepONet (Garg & Chakraborty, 2022) treats DeepONet weights in a variational Bayesian framework, enabling posterior uncertainty for parametric PDEs and showing improved generalisation compared to deterministic DeepONet. B-DeepONet (Lin et al., 2022) uses replica-exchange stochastic gradient Langevin dynamics to train a Bayesian DeepONet that handles noisy data more robustly and estimates uncertainty reliably. α-VI DeepONet (Lone et al., 2024) generalises variational inference using Rényi's α-divergence, aiming for greater robustness to prior misspecification while quantifying uncertainty in operator predictions.

Permutation-invariant function learning has also been extended to neural operator surrogates. Prasthofer et al. (2022) introduced a variable-input version of the original DeepONet architecture which extends the branch network to contain a learned, permutation-invariant encoder. Tretiakov et al. (2025) introduced a branch network architecture based concretely on DeepSets which passes each context pairing through an MLP before aggregating the obtained representations via either a mean pooling or an attention mechanism. While these SetONet variants successfully handle variable-sized inputs, they remain fundamentally deterministic mappings. More recently Ma et al. (2025), proposed a conditional variational autoencoder DeepSet neural operator, which split the branch network into deterministic and small latent components, not unlike the latent variable present in the NPs, but which was limited to modelling processes with Gaussian noise. VSON builds upon the architectural backbone of SetONets but elevates it to a flexible probabilistic framework, replacing the deterministic branch output with a variational latent structure.

2 Background

The problem tackled in the present work is that of meta-learning, that is, given a variable-sized (in N) context set $\mathcal{C} = \{(x_i, y_i)\}_{i=1}^N$ of observations from an unknown function f, we seek a predictive distribution for outputs at target, or query, locations x,

$$p(y \mid x, \mathcal{C}). \tag{2}$$

A strong model for this task should satisfy several desiderata: (i) permutation-invariance to the ordering of context points, (ii) the ability to handle variable context set sizes, and (iii) calibrated uncertainty quantification.

2.1 Neural Processes and their limitations

Neural Processes (NPs) and their attention-augmented variants provide a scalable, neural approximation to stochastic processes by learning a mapping from context sets to conditional predictive distributions Garnelo et al. (2018a); Kim et al. (2019); Gordon et al. (2020). In the canonical NP formulation a context encoder aggregates information from context pairs into a fixed-length summary vector which conditions a location-aware decoder that outputs a predictive distribution at any query position. Latent-variable extensions introduce a global stochastic latent to capture function-level variability.

While NPs are attractive for their scalability and ease of amortised inference, early implementations of NPs suffered from practical limitations. Kim et al. (2019) demonstrated that they tended to underfit to signal, and proposed an attention-based alternative, Attentive Neural processes (ANPs), which mitigate this by enabling each query to selectively attend to relevant context points. Transformer Neural Processes (TNPs) Nguyen & Grover (2023) further improve the idea by arranging context and target tokens into a single sequence and applying transformer-style blocks. Secondly, although the original NP includes a context-dependent

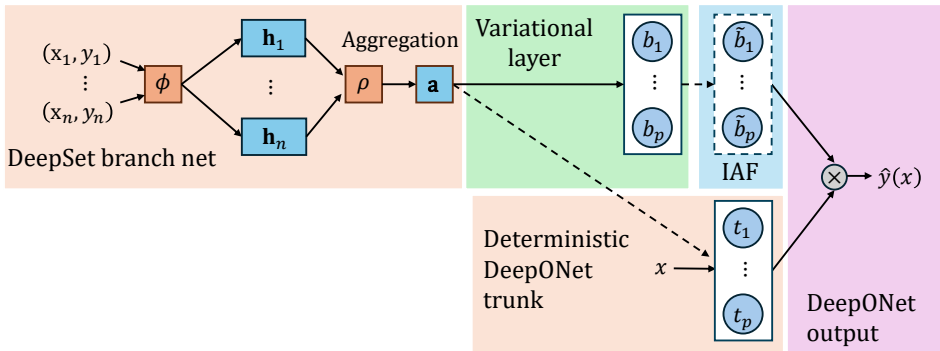


Figure 1: Architecture of the proposed method. The dashed arrow indicates an optional conditioning of the trunk on the contexts' aggregated representation, while the dashed IAF box indicates an optional inverse autoregressive flow layer. If the IAF transformation is not used, $\tilde{\mathbf{b}}$ is the same as $\mathbf{b}$, i.e., the IAF transformation is replaced by the identity function.

latent variable intended to capture function-level variability, predictive uncertainty is still largely expressed through an output Gaussian variance term, which tends to conflate epistemic and aleatoric uncertainty and can lead to miscalibrated predictions. Again, the TNP improved upon the prior formulations by introducing two methods to impose output space correlations: an autoregressive process and a low-rank non-diagonal covariance matrix. Nguyen & Grover (2023) showed that the autoregressive approach (named TNP-A) has a strong ability to capture functional variability, but lacked smoothness.

The present work intends to improve upon the state-of-the-art NPs in two ways. The first is by using a neural operator framework to allow for an expressive output space, where the contexts are fed into a permutation-invariant branch network allowing for variable-sized inputs, and targets fed into the trunk network. Secondly, instead of predicting distributional terms or modelling output space correlations, the uncertainty is learnt via variational variables comprising the branch network outputs. We show that using Inverse Autoregressive Flows (IAFs) Kingma et al. (2017) in conjunction with the diagonal Gaussian layer allows for expressive and non-Gaussian predictive distributions. The trunk output remains deterministic throughout, leading to smooth output space predictions with respect to input coordinates.

3 Methods

The problem is approached from the viewpoint of operator learning, that is, learning a mapping between function spaces. Figure 1 describes the architecture and highlights the various blocks that build upon previous architectures, namely SetONets and DeepONets. Importantly, the SetONet is extended so its branch outputs consist of variational latent variables, which are trained to capture functional uncertainty via their joint distribution.

3.1 Set Operator Networks

The original SetONet implementation proceeds as follows. Given a set of contexts $\mathcal{C}$, the model predicts the value of the target function f at any query location $x \in \Omega$ by first embedding each context pair (x_i, y_i) into a latent representation via a neural network encoder multi-layer perceptron (MLP) ϕ, and the resulting embeddings are aggregated in a permutation-invariant manner:

$$\begin{aligned} \mathbf{a} &= \rho\left(\{\phi(x_i, y_i)_{i=1}^N\}\right) \\ &= \rho\left(\{\mathbf{h}_i\}_{i=1}^N\right) \end{aligned} \tag{3}$$

where ρ denotes the aggregation method. The resulting aggregated vector $\mathbf{a}$ is passed through a final MLP and outputs a p dimensional vector, which represents the basis coefficients in the original DeepONet framework. The query coordinate is passed through a trunk network, typically an MLP, and outputs a set

of p basis functions evaluated at that coordinate. The final prediction at x is given by the inner product of the basis coefficients and basis functions plus a global bias term.

3.2 Variational SetONet

In this work the set-based neural operator described previously is made uncertainty-aware by assuming the branch outputs to be variational variables. Instead of predicting a vector of basis coefficients like in the original SetONet (which itself borrows from the DeepONet architecture), the final branch MLP in the present work predicts distributional terms for each output. Adopting the Variational Autoencoder Kingma & Welling (2022) approach, the branch output vector $\mathbf{b}$ is modelled as a diagonal Gaussian, $q(\mathbf{b}|\mathcal{C}) \sim \mathcal{N}(\mu, \Sigma)$, where the means and diagonal variance terms are outputted by the final branch MLP. We place a standard normal regularising term on the branch latents, $p(\mathbf{b}) \sim \mathcal{N}(0, I)$, akin to a prior in typical variational inference, and train by minimising the task loss and a Kullback-Leibler divergence term $\mathrm{KL}(q(\mathbf{b}|\mathcal{C})||p(\mathbf{b}))$ to keep the learned distribution $q(\mathbf{b}|\mathcal{C})$ from drifting away from the prior or collapsing to a single value Titsias (2009). More details on the training procedure may be found in Section 3.4. For uncertainty estimation, multiple samples $(\mathbf{b}^{(i)})_{i=1}^{K}$ may be drawn from the variational posterior, combined with the deterministic trunk outputs $\mathbf{t}$ and sample statistics gathered. Note that when the branch terms are independent Gaussian, the output of the inner product becomes a sum of scaled Gaussians, which is itself Gaussian. When modelling non-Gaussian behaviour, the latent variables can be augmented via normalising flows to increase flexibility ($\tilde{\mathbf{b}}$ in Figure 1). More detail can be found in Section 3.3.

Transformer aggregation layer To obtain a fixed-size summary of the encoded context pairs $\{\mathbf{h}_i\}_{i=1}^{N}$ we apply a permutation-invariant aggregator ρ, following the DeepSets implementation Zaheer et al. (2018):

$$\mathbf{a} = \rho(\{\mathbf{h}_i\}_{i=1}^{N}), \tag{4}$$

where ρ denotes the aggregation method. There are a few options for aggregation method. The SetONet Tretiakov et al. (2025) utilises mean-pooling and self attention, and here we extend the aggregation to use the Set Transformer module Lee et al. (2019).

3.3 Further extensions

Conditioned basis functions While the original DeepONet implementation uses independent branch and trunk networks, we allow a conditioning of the trunk network on the aggregated context representations, represented in Figure 1 by the dashed line, to account for the increase in complexity of the problem. A similar structure was adopted by Wang et al. (2021), where they conditioned both networks on the other via cross-connections between MLP layers. In the present work, we chose to condition only the trunk on the contexts to preserve continuous predictive samples. If the branch outputs were conditioned on the coordinate input, each coordinate would have a different latent variational representation, breaking the continuity in output space when making posterior draws.

Conditioning is implemented via layer-wise, context-conditioned modulation of trunk activations using Feature-wise Linear Modulation (FiLM) Perez et al. (2017). For trunk layer ℓ with a post-activation value of $u^{(\ell)}$ the conditioned output is

$$g^{(\ell)}(x) = \gamma^{(\ell)}(\mathbf{a}) \odot \mathbf{u}^{(\ell)}(x) + \boldsymbol{\beta}^{(\ell)}(\mathbf{a}), \tag{5}$$

where $\gamma^{(\ell)}(\mathbf{a}), \beta^{(\ell)}(\mathbf{a}) \in \mathbb{R}^{d_\ell}$ are learned transformation layers dependent on the aggregated context summary $\mathbf{a}$ and $\odot$ denotes the Hadamard product. The final trunk bases $\mathbf{t}(x; \mathbf{a}) \in \mathbb{R}^p$ are then combined with a sampled branch latent $\mathbf{b} \sim q_\omega(\mathbf{b} \mid \mathcal{C})$ via an inner product to produce a scalar value:

$$\hat{y}(x) = \frac{1}{p}\, \mathbf{b}^\top \mathbf{t}(x; \mathbf{a}) + b_0, \tag{6}$$

where the $1/p$ factor stabilises inner-product variance Lu et al. (2022) and b_0 is a learned global bias.

Table 1: Results of the 1D regression problem evaluated on three different kernels. We report mean and standard deviation of RMSE on the evaluation sets for 5 different initial seeds. VSON performs best with respect to predictive accuracy on all of the kernels.

Method	**RBF**	**Matérn 5/2**	**Periodic**
ANP	0.193 ± 0.001	0.23 ± 0.002	0.703 ± 0.002
BANP	0.192 ± 0.001	0.228 ± 0.001	0.701 ± 0.08
TNP-D	0.177 ± 0.001	0.222 ± 0.001	0.664 ± 0.014
TNP-A	0.178 ± 0.001	0.222 ± 0.001	0.660 ± 0.002
TNP-ND	0.180 ± 0.001	0.223 ± 0.001	0.67 ± 0.009
VSON	0.102 ± 0.001	0.14 ± 0.001	0.57 ± 0.006
VSON+IAF	0.100 ± 0.001	0.136 ± 0.001	$\mathbf{0.561 \pm 0.005}$
VSON+Cond	$\mathbf{0.095 \pm 0.001}$	$\mathbf{0.128 \pm 0.001}$	0.567 ± 0.004
VSON+Cond+IAF	0.096 ± 0.001	0.131 ± 0.001	0.574 ± 0.004

Inverse autoregressive flows To increase the expressivity of the amortised branch over function-level latents we may incorporate inverse autoregressive flows (IAFs) Kingma & Welling (2014). Let the encoder produce a reparameterisable base density

$$q_\omega(\mathbf{b} \mid \mathcal{C}) = \mathcal{N}\big(\mathbf{b};\, \mu_\omega(\mathcal{C}),\ \mathrm{diag}(\sigma^2_\omega(\mathcal{C}))\big), \tag{7}$$

and let $f_\psi : \mathbb{R}^p \to \mathbb{R}^p$ be a composition of autoregressive bijections. The final branch latent is $\tilde{\mathbf{b}} = f_\psi(\mathbf{b})$, whose density follows by change of variables:

$$q_\psi(\tilde{\mathbf{b}} \mid \mathcal{C}) = q_\omega\big(\mathbf{b} \mid \mathcal{C}\big) \left| \det \frac{\partial f_\psi^{-1}(\tilde{\mathbf{b}})}{\partial \tilde{\mathbf{b}}} \right|. \tag{8}$$

IAFs provide a flexible family of variational posteriors that can represent correlations, skewness and multimodality while remaining easy to sample from. Since the flow is invertible, the KL term used for latent regularisation remains computable, so variational regularisation can still be applied without approximations. There is an important practical trade-off to note. The neural operator maps the latent variables to scalar outputs by a linear combination, that is, the inner product with the trunk terms. If $\mathbf{b}$ is Gaussian then $p(y \mid x, \mathcal{C})$ is Gaussian and many predictive functionals (for example, likelihood) admit closed forms. After a nonlinear flow $\tilde{\mathbf{b}} = f_\psi(\mathbf{b})$ this closed form is generally lost. Predictive evaluation and proper scoring must therefore be estimated by Monte-Carlo sampling from the normalising flow posterior.

3.4 Training the model

We train the VSON by minimising an expected scoring-rule loss on task-wise targets plus a latent regulariser. Each training example consists of a partially observed context set of input-output pairs and a set of target, or query, locations where predictions are made. The context is a finite set of observed input-output pairs that summarise partial information about the underlying function f. The model predicts a distribution at the set of target input locations, also called queries, with corresponding target output values used only during training to compute the loss. During training we sample many random partitions of observed points into context and targets (queries) to encourage robustness to different amounts and locations of partial observation. Let a task provide context $\mathcal{C}$ (Equation 1) and targets $\mathcal{T} = \{(x_j, y_j)\}_{j=1}^m$. The branch induces a variational posterior $q_\omega(\mathbf{b} \mid \mathcal{C})$ over branch latents $\mathbf{b}$, and the inner product with the trunk defines a predictive CDF over the output variable $\hat{y}$ at query x

$$F_\theta(\hat{y} \mid x, \mathbf{b}), \tag{9}$$

where θ denotes all model parameters, which consists of parameters in the deterministic trunk, context encoder and variational parameters.

Instead of assuming a Gaussian noise model on outputs, we use the continuous ranked probability score (CRPS) to match the predicted distributions with data.

$$\begin{aligned}&\mathcal{L}(\theta;\mathcal{C},\mathcal{T}) = \\ &\mathbb{E}_{\tilde{\mathbf{b}}\sim q_\psi(\tilde{\mathbf{b}}|\mathcal{C})}\left[\frac{1}{m}\sum_{j=1}^{m}\mathrm{CRPS}\big(F_\theta(\cdot \mid x_j,\tilde{\mathbf{b}}),\, y_j\big)\right] \\ &+\beta\,\mathrm{KL}\big(q_\psi(\tilde{\mathbf{b}}\mid\mathcal{C})\,\|\,p(\tilde{\mathbf{b}})\big),\end{aligned} \tag{10}$$

where the CRPS is defined by

$$\mathrm{CRPS}(F,y) = \int_{-\infty}^{\infty}\big(F(\hat{y}) - \mathbb{1}\{\hat{y}\geq y\}\big)^2\,\mathrm{d}\hat{y}. \tag{11}$$

We adopt the CRPS as a strictly proper scoring rule for evaluating and training distributional predictions, as it measures the discrepancy between the full predictive cumulative distribution and the empirical distribution of targets Matheson & Winkler (1976). Unlike likelihood-based training, CRPS remains well-defined and numerically stable when the predictive distribution is only accessible through samples, which is the case for flow-augmented branch latents. The regularisation term in Eq 10 arises from placing a prior $p(b)$ on the branch latents and performing amortised variational inference. Minimising the CRPS loss together with $\mathrm{KL}\big(q_\psi(\tilde{\mathbf{b}}\mid\mathcal{C})\,\|\,p(\tilde{\mathbf{b}})\big)$ corresponds to optimising a variational objective that trades off predictive accuracy and latent regularity, analogous to the evidence lower bound (ELBO) in variational autoencoders. When the predictive distribution $F_\theta(\hat{y}\mid x,\mathbf{b})$ is approximated through K Monte-Carlo samples $\{\hat{y}^{(k)}\}_{k=1}^{K}$, $\hat{y}^{(k)}\sim F_\theta(\cdot\mid x,b)$, we use an unbiased Monte-Carlo estimator Gneiting & Raftery (2007)

$$\begin{aligned}&\widehat{\mathrm{CRPS}}\big(F_\theta(\hat{y}\mid x,\tilde{\mathbf{b}}),y\big) = \\ &\frac{1}{K}\sum_{k=1}^{K}\big|\hat{y}^{(k)}-y\big| - \frac{1}{2K^2}\sum_{k=1}^{K}\sum_{\ell=1}^{K}\big|\hat{y}^{(k)}-\hat{y}^{(\ell)}\big|,\end{aligned} \tag{12}$$

where the first component encourages fitting to the observed data, and the second encourages spread. Optimisation proceeds over all model parameters θ.

4 Results

The proposed method was tested against the state-of-the-art in three different settings: 1D regression, image regression and Bayesian optimisation in 1- and 2-D settings, which constitute typical benchmarks for Neural process evaluation. The method was tested against Transformer Neural Processes, Attentive Neural Processes and Bootstrapping Neural Processes, and used the implementation and hyperparameters of the models provided by Nguyen & Grover (2023). Four different VSON models were used to test for impact and interactions of the extensions described in Section 3.3: with or without IAF on the branch latents and with or without conditioning the trunk on the aggregated context representation.

4.1 1D regression

In the 1D regression example, the methods are trained on realisations from a Gaussian Process (GP) prior with an RBF kernel using varying lengthscales and amplitudes. The training and evaluation methodology mirrored the train and evaluation methodology laid out by Nguyen & Grover (2023). Specifically, for each GP realisation lengthscales were sampled from $\ell\sim\mathcal{U}(0.1,0.6)$[1], and signal amplitudes from $\sigma_f\sim\mathcal{U}(0.1,1)$. A constant jitter term σ was added to the diagonal of the covariance matrix equal to 0.02. The number of contexts and targets for each sampled GP were sampled discretely from $N\sim\mathcal{U}(6,50)$ and $N\sim\mathcal{U}(3,47)$,

[1]Although authors of previous works using the benchmark have stated ℓ is sampled from $\mathcal{U}(0.6,1)$, inspection of the code used revealed the interval was as stated in the present text.

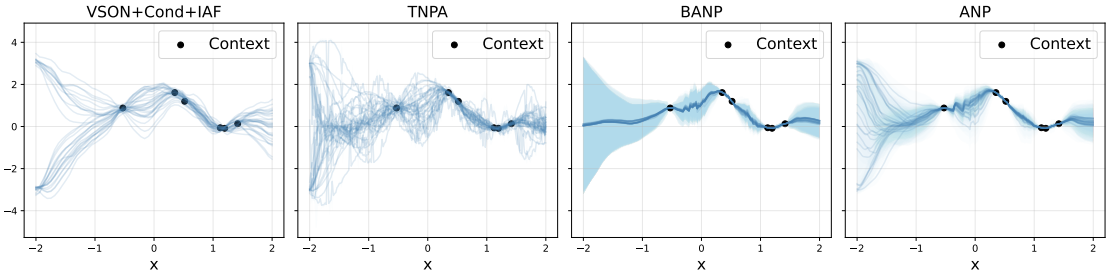


Figure 2: Posterior samples in output space obtained using VSON-Cond-IAF and the comparison models for the conditional GP problem, where an observation is made at x=-2 with value y=-3 or y=3 with even probabilities. VSON is well-equipped to handle multimodal output behaviour. 6 context points were used.

respectively. Training was performed in batches of 16 functions for a total of 100,000 batches. A cosine-decayed learning rate going from 5×10^{-4} to 5×10^{-5} was used in conjunction with Adam Kingma & Ba (2014).

The trained models were evaluated on three different test sets obtained by sampling from GP priors with RBF, Matérn 5/2 and periodic kernels, where the lengthscale and amplitude ranges remained the same as during training, and the periodic kernel additionally sampled its period from $\mathcal{U}(0.1, 0.5)$. Each test set consisted of 48000 functions which were evaluated in batches of size 16, and the metrics averaged over the batches. Results with regard to predictive accuracy are reported in Table 1, while additional metrics detailing uncertainty calibration using CRPS and coverage error (CE) can be found in the Appendix B. Conditioning the trunk on the context's aggregation leads to a slight improvement in predictive accuracy and calibration on the first two kernels, while IAF doesn't have as large an impact. Given that the data is roughly Gaussian for the problem, it follows that augmenting the latents via a normalising flow wouldn't substantially improve results.

Table 2: Results of the 1D multimodal regression problem trained and evaluated on realisations from a conditioned GP. We report RMSE on the evaluation set. VSON performs best with respect to predictive accuracy.

Method	**RMSE**
ANP	0.134
BANP	0.141
TNP-D	0.107
TNP-A	0.115
TNP-ND	0.111
VSON+Cond+IAF	**0.096**

4.2 1D multi-modal regression

The model was also trained and tested on a realisations from a GP with conditioned outputs and an RBF kernel. Specifically, a data point was observed at $x = -2$ and with value $y = 3$ or $y = -3$ with even probabilities, and the GP draws were made conditional on the observed value. Outside of the conditioning, training followed the same setup as the previous regression task. To capture the multi-modality in output space and based on the performance in the first regression task, only a VSON model with IAF on the branch latents and a conditioned trunk was used.

Figure 2 represents qualitative performance via samples from the posterior of the trained VSON+Cond+IAF model and 3 comparison models given 6 context points. It highlights the fact that the VSON is able to recover

the data-generating process accurately, even with multi-modal outputs. Additional metrics may be found in Table 2.

4.3 Image completion

Table 3: Results of the image regression on EMNIST for seen and unseen classes (digits and letters, respectively). We report mean and one standard deviation for RMSE and CRPS on the evaluation set for 5 initial seeds. The VSON model with both a conditioned trunk and IAF performs best with respect to uncertainty calibration and slightly worse than the state of the art with respect to predictive accuracy on the seen dataset, and best with respect to uncertainty quantification on the unseen dataset. The experiment was run for 5 different initial seeds.

Setting	**Method**	**RMSE**	**CRPS**
Seen classes (0–9)	ANP	0.142 ± 0.000	0.061 ± 0.003
	BANP	0.142 ± 0.001	0.063 ± 0.003
	TNP-D	0.119 ± 0.002	0.046 ± 0.010
	TNP-A	0.122 ± 0.001	0.047 ± 0.005
	TNP-ND	$\mathbf{0.116 \pm 0.000}$	0.046 ± 0.002
	VSON	0.153 ± 0.002	0.060 ± 0.002
	VSON+IAF	0.145 ± 0.001	0.054 ± 0.002
	VSON+Cond	0.119 ± 0.001	0.039 ± 0.002
	VSON+Cond+IAF	0.118 ± 0.001	$\mathbf{0.037 \pm 0.002}$
Unseen classes (10–46)	ANP	0.166 ± 0.001	0.0712 ± 0.004
	BANP	0.161 ± 0.001	0.0713 ± 0.003
	TNP-D	$\mathbf{0.139 \pm 0.001}$	0.054 ± 0.003
	TNP-A	0.142 ± 0.001	0.0555 ± 0.005
	TNP-ND	0.140 ± 0.001	0.0545 ± 0.004
	VSON	0.185 ± 0.001	0.076 ± 0.002
	VSON+IAF	0.176 ± 0.002	0.067 ± 0.002
	VSON+Cond	0.150 ± 0.001	0.05 ± 0.002
	VSON+Cond+IAF	0.148 ± 0.001	$\mathbf{0.048 \pm 0.002}$

Image completion was investigated using the EMNIST dataset Cohen et al. (2017). Each pixel was treated as the value of an underlying image function $y : [-1, 1]^2 \to [-0.5, 0.5]$ at a coordinate $x \in [-1, 1]^2$. VSON was used as a surrogate model, accepting a context set and a query location as inputs and outputting pixel brightness. The model was trained on the first 10 classes of the dataset, corresponding to the digits 0-9, and evaluated on held-out test sets of the seen classes (digits) and remaining unseen classes (letters). For each task, the number of contexts were sampled from $\mathcal{U}(6, 200)$ and number of targets from $\mathcal{U}(3, 197)$, mimicking the values used in previous studies. Optimisation was carried out using mini-batches containing 100 tasks for a total of 200 epochs. Table 3 displays results on the evaluation sets with respect to predictive accuracy and uncertainty quantification. The results indicate the combination of IAF and conditioned trunk have a large effect on the more complex problem of digit completion, and significantly more so than in the 1D regression problem. Figure 3 presents mean predictions of 3 digits belonging to the observed classes and 3 letters belonging to the unobserved classes. The VSON model with the conditioned trunk produces sharper outputs due to the additional information fed to the basis functions. Comparisons to the state-of-the-art may be found in Section C in the appendix.

4.4 Bayesian optimisation

Bayesian optimisation (BO) was carried out in 1-, and 2-dimensional settings. The 1D problem utilised the same setup as the 1D regression problem, namely, VSON was implemented as a surrogate model used to make predictions at new points while accounting for uncertainty. However, the evaluation differed, in that the model was used to optimise black box functions. In the 1D case, the model was trained on GP samples using an RBF kernel, and evaluated on three evaluation sets of synthetic black box functions obtained by

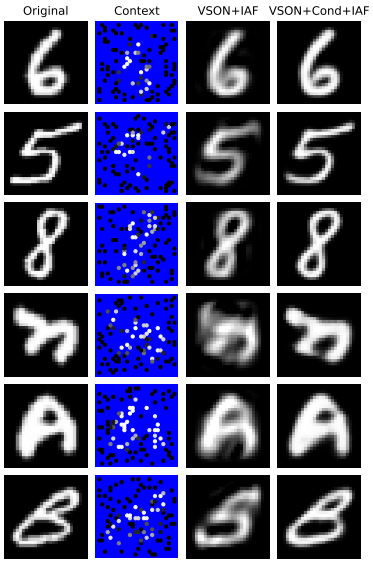


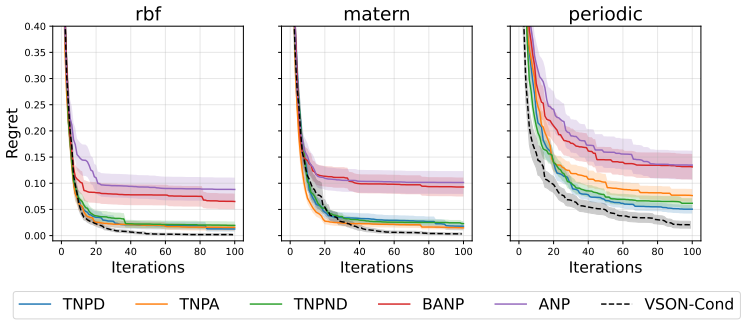
Figure 3: Image completion using 100 context points on the EMNIST dataset. The 2 rightmost columns display mean predictions using VSON+IAF and VSON+Cond+IAF. The first 3 rows display tasks from observed classes, while the latter 3 rows are tasks from unobserved classes. The VSON model with the conditioned trunk produces sharper outputs due to the more informed basis functions.

sampling from a GP prior using RBF, Matérn 5/2, and periodic kernels. Each evaluation set contained 100 functions, and BO was carried out for 100 iterations. The acquisition function used for the 1D problem was Expected Improvement (EI) Mockus (1978) while upper confidence bound (UCB) was used for the 2D setting, following the procedure of previous authors, and performance was measured using simple regret. The conditioned trunk VSON model was used for the task, while IAF wasn't, given that the training data came from a GP, which also allowed use of the closed form of EI.

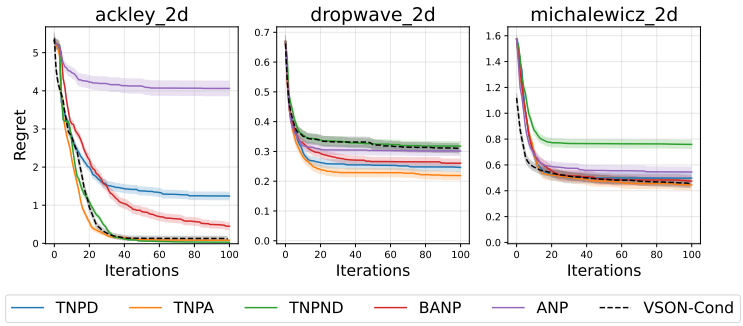
Figure 4a displays results of the 1D BO problem, where VSON performed best across all kernels, mirroring the results of the 1D regression problem. For the 2D problem, the VSON was trained on realisations from a multivariate GP with an RBF kernel, and evaluated on examples borrowed from Bayesian optimisation benchmarks Kim (2023). Figure 4b displays the results over 100 different seeds for 100 BO iterations, indicating that VSON was competitive on 2/3 of the problems.

4.5 Computational complexity

An attractive property of the proposed method is the separation of the query from the context attention mechanisms present in the branch. TNP-D encounters $\mathcal{O}((N+m)^2)$ complexity at prediction time, where N is the number of context points and m the number of query points, due to the cross-attention mechanisms in the architecture. Although research has been carried out in finding a balance between the flexibility of TNPs and overall complexity Feng et al. (2023), state-of-the-art neural operators still rely on attention mechanisms between contexts and targets to avoid the under-fitting issues present in the original conditional NP implementation Garnelo et al. (2018a), which has $\mathcal{O}(m)$ complexity at prediction time. Due to the independence between branch and trunk, VSON shares a $\mathcal{O}(n^2)$ encoding complexity, while retaining an $\mathcal{O}(m)$ querying complexity that matches the cheapest NP baselines, without sacrificing the calibration improvements reported in Section 3. Note that both the base VSON model and the conditioned trunk model share this complexity, as the conditioning vectors described in Section 3.3 depend on the number of contexts only, and then remain fixed for any query.



(a)



(b)

Figure 4: (a) Results of the 1D Bayesian optimisation problem using EI on 3 different evaluation sets. VSON-Cond retrieves the ground truth quicker than the alternatives, and performs well on kernels it wasn't trained on. (b) Results of the 2D Bayesian optimisation problem using UCB on 3 different benchmark problems. VSON-Cond is competitive with state-of-the-art on 2/3 of the benchmarks.

5 Discussion

This work introduces a Variational Set Operator Network, an uncertainty-aware neural operator, to tackle meta-learning problems. The framework combines the meta-learning capabilities inherent in Neural Processes (NPs) with the expressive, continuous function-mapping strengths of Deep Operator Networks. Building upon previous research which tackles the limitations of standard operators that typically rely on fixed input discretisations, we make the natural extension to account for the uncertainty that follows sparse context. Empirically, VSON improves uncertainty calibration across all benchmarks while remaining competitive on predictive accuracy. It outperforms the Neural Process baselines on 1D regression, multimodal regression, and Bayesian optimisation, and on image completion accepts a small reduction in accuracy in exchange for markedly better calibration.

A core architectural advantage of VSON is the preservation of smoothness and continuity in predictive samples. Unlike autoregressive variants like TNP-A, which may produce jagged predictions due to their sequential nature, VSON utilises a deterministic trunk network that ensures function samples remain smooth with respect to input coordinates. Furthermore, the model offers computational benefits. While attention-heavy models like TNP-D scale quadratically ($O((N+m)^2)$) with the number of context and query points, VSON maintains a linear $O(m)$ querying complexity, albeit a $O(N^2)$ context encoding complexity. Nevertheless, the querying efficiency makes the framework particularly suitable for high-resolution tasks where dense querying is required, for example BO or adaptive design problems in higher dimensions.

Despite these advantages, the transition to non-Gaussian latent spaces introduces a necessary trade-off. The loss of a closed-form predictive distribution requires reliance on Monte-Carlo sampling for inference. Looking forward, the versatility of the VSON framework suggests several promising avenues for extension. Applying the model to physical systems and partial differential equations (PDEs), the original application

of DeepONets, where uncertainty quantification is critical, would be a natural progression. Exploring incorporating prior knowledge in the model via latent representations such as in the Amortised Conditioning Engine introduced by Chang et al. (2025) may also be an interesting extension, enabling stronger inference.

Software and Data

Code is to be made available in the near future.

References

Paul E. Chang, Nasrulloh Loka, Daolang Huang, Ulpu Remes, Samuel Kaski, and Luigi Acerbi. Amortized Probabilistic Conditioning for Optimization, Simulation and Inference, March 2025. URL http://arxiv.org/abs/2410.15320. arXiv:2410.15320 [stat].

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters, March 2017. URL http://arxiv.org/abs/1702.05373. arXiv:1702.05373 [cs].

Leo Feng, Hossein Hajimirsadeghi, Yoshua Bengio, and Mohamed Osama Ahmed. Latent Bottlenecked Attentive Neural Processes, March 2023. URL http://arxiv.org/abs/2211.08458. arXiv:2211.08458 [cs].

Shailesh Garg and Souvik Chakraborty. Variational bayes deep operator network: A data-driven bayesian solver for parametric differential equations. *arXiv preprint arXiv:2206.05655*, 2022.

Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional Neural Processes, July 2018a. URL http://arxiv.org/abs/1807.01613. arXiv:1807.01613 [cs].

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural Processes, July 2018b. URL http://arxiv.org/abs/1807.01622. arXiv:1807.01622 [cs].

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Jonathan Gordon, Wessel P. Bruinsma, Andrew Y. K. Foong, James Requeima, Yann Dubois, and Richard E. Turner. Convolutional Conditional Neural Processes. April 2020. URL https://iclr.cc/virtual_2020/poster_Skey4eBYPS.html.

Daolang Huang, Yujia Guo, Luigi Acerbi, and Samuel Kaski. Amortized Bayesian Experimental Design for Decision-Making, January 2025. URL http://arxiv.org/abs/2411.02064. arXiv:2411.02064 [stat].

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive Neural Processes, July 2019. URL http://arxiv.org/abs/1901.05761. arXiv:1901.05761 [cs].

Jungtaek Kim. BayesO Benchmarks: Benchmark functions for Bayesian optimization. https://doi.org/10.5281/zenodo.7577330, 2023. URL https://github.com/jungtaekkim/bayeso-benchmarks.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014. URL https://arxiv.org/abs/1312.6114.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, December 2022. URL http://arxiv.org/abs/1312.6114. arXiv:1312.6114 [stat].

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving Variational Inference with Inverse Autoregressive Flow, January 2017. URL http://arxiv.org/abs/1606.04934. arXiv:1606.04934 [cs].

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks, May 2019. URL http://arxiv.org/abs/1810.00825. arXiv:1810.00825 [cs].

Juho Lee, Yoonho Lee, Jungtaek Kim, Eunho Yang, Sung Ju Hwang, and Yee Whye Teh. Bootstrapping Neural Processes, October 2020. URL http://arxiv.org/abs/2008.02956. arXiv:2008.02956 [cs].

Cen-You Li, Marc Toussaint, Barbara Rakitsch, and Christoph Zimmer. Amortized Safe Active Learning for Real-Time Data Acquisition: Pretrained Neural Policies from Simulated Nonparametric Functions, May 2025. URL http://arxiv.org/abs/2501.15458. arXiv:2501.15458 [cs].

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations, May 2021. URL http://arxiv.org/abs/2010.08895. arXiv:2010.08895 [cs].

Guang Lin, Christian Moya, Zecheng Zhang, et al. B-deeponet: An enhanced bayesian deeponet for solving noisy parametric pdes using accelerated replica exchange sgld. *Journal of Computational Physics*, 473: 111624, 2022. doi: 10.1016/j.jcp.2022.111713.

Soban Nasir Lone, Subhayan De, and Rajdip Nayek. Alpha-vi deeponet: A prior-robust variational bayesian approach for enhancing deeponets with uncertainty quantification. *arXiv preprint arXiv:2408.00681*, 2024.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5. URL https://www.nature.com/articles/s42256-021-00302-5. Publisher: Nature Publishing Group.

Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, April 2022. ISSN 0045-7825. doi: 10.1016/j.cma.2022.114778. URL https://www.sciencedirect.com/science/article/pii/S0045782522001207.

Lei Ma, Ling Guo, Hao Wu, and Tao Zhou. Deep set based operator learning with uncertainty quantification, September 2025. URL http://arxiv.org/abs/2509.25646. arXiv:2509.25646 [cs].

James E. Matheson and Robert L. Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976.

Jonathan Mockus. *Application of Bayesian Methods to Numerical Methods of Global and Stochastic Optimization*. PhD thesis, Vilnius University, Vilnius, Lithuania, 1978.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-Context Learning for Bayesian Optimization, July 2023. URL http://arxiv.org/abs/2305.17535. arXiv:2305.17535 [cs].

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers Can Do Bayesian Inference, August 2024. URL http://arxiv.org/abs/2112.10510. arXiv:2112.10510 [cs].

Tung Nguyen and Aditya Grover. Transformer Neural Processes: Uncertainty-Aware Meta Learning Via Sequence Modeling, February 2023. URL http://arxiv.org/abs/2207.04179. arXiv:2207.04179 [cs].

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer, December 2017. URL http://arxiv.org/abs/1709.07871. arXiv:1709.07871 [cs].

Michael Prasthofer, Tim De Ryck, and Siddhartha Mishra. Variable-Input Deep Operator Networks, May 2022. URL http://arxiv.org/abs/2205.11404. arXiv:2205.11404 [cs].

Michalis Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pp. 567–574. PMLR, April 2009. URL https://proceedings.mlr.press/v5/titsias09a.html.

Stepan Tretiakov, Xingjian Li, and Krishna Kumar. SetONet: A Deep Set-based Operator Network for Solving PDEs with permutation invariant variable input sampling, May 2025. URL http://arxiv.org/abs/2505.04738. arXiv:2505.04738 [cs].

Sifan Wang, Hanwen Wang, and Paris Perdikaris. Improved architectures and training algorithms for deep operator networks, October 2021. URL http://arxiv.org/abs/2110.01654. arXiv:2110.01654 [cs].

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets, April 2018. URL http://arxiv.org/abs/1703.06114. arXiv:1703.06114 [cs].

A Implementation

Here we report the hyperparameters used for each of the problems. Model dimension d refers to the dimension of the aggregation vector, and latent dimension p serves as the dimension of the inner product output as well. For all problems, the number of samples K to construct the predictive distribution during training was 50, and during evaluation 200, and the scaling parameter β of the regulariser was kept at 1×10^{-4}. When utilising IAF, one IAF layer containing an MLP of depth 2 and using the model dimension width was used. All GPU tasks were run on a single NVIDIA RTX A6000.

1D regression

- Model dimension d: 64
- Latent dimension p: 64
- Number of transformer layers: 2
- Number of transformer heads: 4
- Branch MLP width: 64
- Branch MLP depth: 1
- Trunk MLP width: 64
- Trunk MLP depth: 2
- Activation: ReLU

1D multimodal regression See 1D regression

Image completion

- Model dimension d: 256
- Latent dimension p: 256
- Number of transformer layers: 2
- Number of transformer heads: 8
- Branch MLP width: 256
- Branch MLP depth: 2
- Trunk MLP width: 256
- Trunk MLP depth: 4
- Activation: ReLU

1D BO See 1D regression

Table 4: Parameter counts in the 1D regression problem for the various models.

Method	**Number of parameters**
ANP	348418
BANP	364674
TNP-D	222082
TNP-A	220082
TNP-ND	332821
VSON	110913
VSON+IAF	193601
VSON+Cond	127553
VSON+Cond+IAF	210241

2D BO

- Model dimension d: 256
- Latent dimension p: 256
- Number of transformer layers: 2
- Number of transformer heads: 8
- Branch MLP width: 246
- Branch MLP depth: 2
- Trunk MLP width: 256
- Trunk MLP depth: 4
- Activation: ReLU

Table 4 displays parameter counts for the compared models for the 1D regression problem.

B 1D regression additional metrics

B.1 GP prior metrics

Table 5 displays metrics of the 1D regression experiment where 3 different kernels were used to sample from a GP prior. Using IAF in the VSON model didn't appear to have a large impact on the results, while conditioning the trunk lead to slightly improved performance on the first two kernels. The fact that the conditioned trunk model performed worse on the (out-of-distribution) periodic kernel may indicate overfitting, and that the basis functions learnt by the independent trunk tend to be more generalisable.

C Image completion comparison figures

Figure 5 displays the mean predictions on 8 tasks from the seen classes and 8 from the unseen classes in the EMNIST dataset using VSON+IAF, VSON+Cond+IAF, TNP-D and TNP-ND.

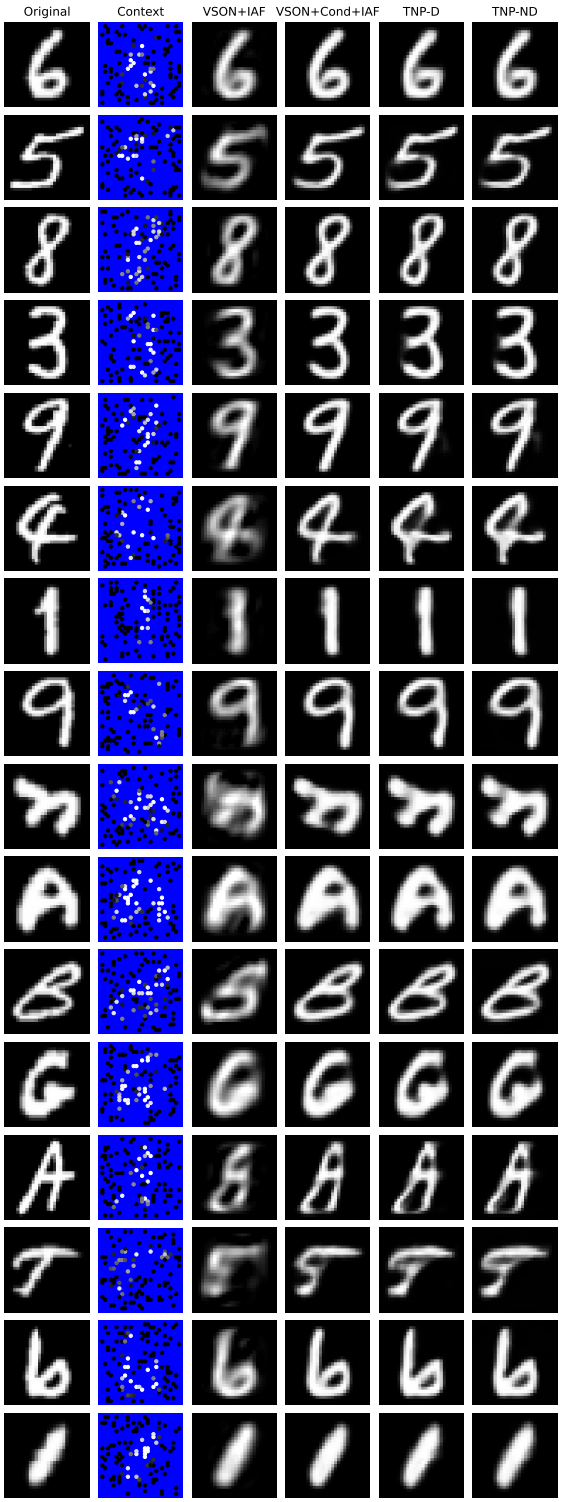


Figure 5: Mean predictions on 8 tasks from the seen classes and 8 from the unseen classes in the EMNIST dataset using VSON+IAF, VSON+Cond+IAF, TNP-D and TNP-ND.

Table 5: Results of the 1D regression problem evaluated on three different kernels. We report mean and one standard deviation for CRPS, RMSE and CE for each of the evaluation sets for 5 initial seeds. VSON performs best with respect to uncertainty calibration and predictive accuracy on the on all kernels.

Metric	**Method**	**RBF**	**Matérn 5/2**	**Periodic**
RMSE	ANP	0.19 ± 0.00	0.23 ± 0.00	0.70 ± 0.00
	BANP	0.19 ± 0.00	0.23 ± 0.00	0.70 ± 0.01
	TNP-D	0.18 ± 0.00	0.22 ± 0.00	0.66 ± 0.01
	TNP-A	0.18 ± 0.00	0.22 ± 0.00	0.66 ± 0.00
	TNP-ND	0.18 ± 0.00	0.22 ± 0.00	0.67 ± 0.01
	VSON	0.102 ± 0.00	0.14 ± 0.00	0.57 ± 0.001
	VSON+IAF	0.100 ± 0.00	0.136 ± 0.00	$\mathbf{0.561 \pm 0.001}$
	VSON+Cond	$\mathbf{0.095 \pm 0.00}$	$\mathbf{0.128 \pm 0.00}$	0.567 ± 0.001
	VSON+Cond+IAF	0.096 ± 0.00	0.131 ± 0.00	0.574 ± 0.001
CE	ANP	0.24 ± 0.00	0.17 ± 0.00	0.27 ± 0.00
	BANP	0.24 ± 0.00	0.17 ± 0.00	0.22 ± 0.01
	TNP-D	0.04 ± 0.00	0.05 ± 0.00	0.13 ± 0.00
	TNP-A	0.05 ± 0.00	0.04 ± 0.00	0.12 ± 0.00
	TNP-ND	0.05 ± 0.00	0.05 ± 0.00	0.16 ± 0.01
	VSON	0.03 ± 0.00	0.074 ± 0.00	$\mathbf{0.06 \pm 0.001}$
	VSON+IAF	0.026 ± 0.00	0.060 ± 0.00	0.129 ± 0.001
	VSON+Cond	0.03 ± 0.00	0.055 ± 0.00	0.127 ± 0.001
	VSON+Cond+IAF	$\mathbf{0.024 \pm 0.00}$	$\mathbf{0.051 \pm 0.00}$	0.13 ± 0.001
CRPS	ANP	0.084 ± 0.00	0.109 ± 0.00	0.463 ± 0.01
	BANP	0.097 ± 0.00	0.121 ± 0.00	0.463 ± 0.01
	TNP-D	0.0698 ± 0.00	0.092 ± 0.00	0.3507 ± 0.02
	TNP-A	0.0595 ± 0.00	0.0862 ± 0.00	0.387 ± 0.02
	TNP-ND	0.0713 ± 0.00	0.0937 ± 0.00	0.3613 ± 0.01
	VSON	0.053 ± 0.00	0.075 ± 0.00	0.33 ± 0.001
	VSON+IAF	0.052 ± 0.00	0.074 ± 0.00	$\mathbf{0.328 \pm 0.001}$
	VSON+Cond	0.047 ± 0.00	$\mathbf{0.067 \pm 0.00}$	0.33 ± 0.001
	VSON+Cond+IAF	$\mathbf{0.046 \pm 0.00}$	0.068 ± 0.00	0.34 ± 0.001