Supplementary Materials

A. Some Basic Concepts

At first, we revisit some basics, which would help us understand properties of GPs and NPs. Fundamental concepts include *permutation invariant function* (*PIF*), *general stochastic function process* (*GSFP*) etc. We stress the importance of relationship disentanglement between GPs and NPs and motivations in approximating stochastic processes (*SPs*) in the main passage.

Definition 1. Permutation Invariant Function. The *f* mapping a set of N M-dimension elements to a D-dimension vector variable is said to be a *permutation invariant function* if:

$$\begin{aligned} f : \times_{i=1}^{N} \mathcal{R}_i^M &\to \mathcal{R}^D \\ x = [x_1, \dots, x_N] \mapsto f &= \big[f_1(x_{\pi(1:N)}), \dots, f_D(x_{\pi(1:N)})\big] \end{aligned} \tag{A.1}$$

where x_i is a M-dimensional vector, $x_{1:N} = [x_1, x_2, \dots, x_N]$ is a set, operation $\pi : [1, 2, \dots, N] \mapsto [\pi_1, \pi_2, \dots, \pi_N]$ imposes a permutation over the order of elements in the set. The *PIF* suggests the image of the map is regardless of the element order. Another related concept *permutation equivariant function (PEF)* keeps the order of elements in the output consistent with that in the input under any permutation operation π.

$$\begin{aligned} f : \times_{i=1}^{N} \mathcal{R}_i^M &\to \mathcal{R}^N \\ x_\pi = (x_{\pi_1}, \dots, x_{\pi_N}) \mapsto f_\pi &= \pi \circ f(x_{1:N}) \end{aligned} \tag{A.2}$$

The **Definition 1** is important, since the exchangeable stochastic function process of our interest in this domain is intrinsically a distribution over set function, and PIF plays as an inductive bias in preserving invariant statistics.

Permutation invariant functions are candidate functions for learning embeddings of a set or other order uncorrelated data structure $X_{1:N}$, and several examples can be listed in the following forms.

- Some structure with mean/summation over the output:

$$F(X_{\pi(1:N)}) = \big(\frac{1}{N}\Sigma_{i=1}^{N}\phi_1(x_i), \frac{1}{N}\Sigma_{i=1}^{N}\phi_2(x_i), \dots, \frac{1}{N}\Sigma_{i=1}^{N}\phi_M(x_i)\big) \tag{A.3}$$

- Some structure with Maximum/Minimum/Top k-th operator over the output (Take maximum operator for example), such as:

$$F(X_{\pi(1:N)}) = \big(max_{i\in\{1,2,\dots,N\}}\phi_1(x_i), max_{i\in\{1,2,\dots,N\}}\phi_2(x_i), \dots, max_{i\in\{1,2,\dots,N\}}\phi_M(x_i)\big) \tag{A.4}$$

- Some structure with symmetric higher order polynomials or other functions with a symmetry bi-variate function ϕ:

$$F(X_{\pi(1:N)}) = \big(\Sigma_{i,j=\{1,2,\dots,N\}}\phi_1(x_i, x_j), \Sigma_{i,j=\{1,2,\dots,N\}}\phi_2(x_i, x_j), \dots, \Sigma_{i,j=\{1,2,\dots,N\}}\phi_M(x_i, x_j)\big) \tag{A.5}$$

The invariant property is easy to be verified in these cases, and note that in all settings of NP family in this paper, **Eq. (A.3)** is used only. For the bi-variate symmetric function in **Eq. (A.5)** or other more complicated operators would result in more flexible functional translations, but additional computation is required as well. Some of the above mentioned transformations are instantiations in DeepSet (Zaheer et al., 2017). Further investigations in this domain can be the exploitation of these higher order permutation invariant neural network (Giles & Maxwell, 1987) into NPs since more correlations or higher order statistics in the set can be mined for prediction. Additionally, Set Transformer (Lee et al., 2019) is believed to be powerful in a permutation invariant representation.

Definition 2. General Stochastic Function Process. Let $\mathcal{X}$ denote the Cartesian product of some intervals as the index set and let dimension of observations $d \in \mathbb{N}$. For each $k \in \mathbb{N}$ and any finite sequence of distinct indices $x_1, x_2, .., x_k \in \mathcal{X}$, let $\nu(x_1, x_2, .., x_k)$ be some probability measure over $(\mathcal{R}^d)^k$. Suppose the used measure satisfies Kolmogorov Extension Theorem, then there exists a probability space $(\Omega, \mathcal{F}, \mathcal{P})$, which induces a *general stochastic function process (GSFP)* $\mathcal{F} : \mathcal{X} \times \Omega \to \mathcal{R}^d$, keeping the property $\nu_{(x_1,x_2,..,x_k)}(\mathcal{C}_1 \times \mathcal{C}_2 \times ... \times \mathcal{C}_k) = \mathcal{P}(\mathcal{F}(x_1) \in C_1, \mathcal{F}(x_2) \in C_2, ..., \mathcal{F}(x_k) \in \mathcal{C}_k)$ for all $x_i \in \mathcal{X}$, $d \in \mathbb{N}$ and measurable sets $\mathcal{C}_i \in \mathcal{R}^d$.

The **Definition 2** presents an important concept for stochastic processes in high-dimensional cases, and this is a general description for the task to learn in mentioned related works. This includes but not limited to GPs and characterizes the distribution over the stochastic function family.

B. Proof of Proposition 1

As we know, a Gaussian distribution is closed under marginalization, conditional probability computation and some other trivial operations. Here the statistical parameter invariance towards the order of the context variables in per sample predictive distribution would be demonstrated.

Given a multivariate Gaussian as the context $X = [x_1, x_2, \dots, x_N]^T \sim \mathcal{N}(X; [\mu_1, \mu_2, \dots, \mu_N]^T, \Sigma(x_1, x_2, \dots, x_N))$, and for any permutation operation over the order $\pi : [1, 2, \dots, N] \to [\pi_1, \pi_2, \dots, \pi_N]$, there exist a permutation matrix $\mathcal{P}_\pi = [e_{\pi_1}^T, e_{\pi_2}^T, \dots, e_{\pi_N}^T]$, where only the π_i-th position is one with the rest zeros. Naturally, it results in a permutation over the random variables in coordinates.

$$P_\pi [x_1, x_2, \dots, x_N]^T = [x_{\pi_1}, x_{\pi_2}, \dots, x_{\pi_N}]^T = X_\pi \tag{B.1}$$

The random variable X_π follows another multivariate Gaussian as $X_\pi \sim \mathcal{N}(X_\pi; P_\pi \mu, P_\pi \Sigma P_\pi^T) = \mathcal{N}(X_\pi; \mu_\pi, \Sigma_\pi)$. In an elementwise way, we can rewrite the statistics in the form as follows.

$$\begin{aligned} &\mathbb{E}[x_{\pi_i}] = \mu_{\pi_i} \\ &\sigma_{ls}^\pi = e_l^T \Sigma_\pi e_s = e_l^T P_\pi \Sigma P_\pi^T e_s = e_{\pi_l} \Sigma e_{\pi_s} = cov(x_{\pi_l}, x_{\pi_s}) \end{aligned} \tag{B.2}$$

Notice in **Eq. (B.2)**, the statistics are *permutation equivariant* now.

As the most important component in GPs, the predictive distribution conditioned on the context $D = [X_{1:N}, Y_{1:N}]$ can be analytically computed once GPs are well trained and result in some mean function m_θ.

$$\begin{aligned} &p(y_*|Y_{1:N}) = \mathcal{N}(y_*|\tilde{\mu}, \tilde{\sigma}^2) \\ &\tilde{\mu} = m_\theta(x_*) + \Sigma_{x_*,D} \Sigma_{D,D}^{-1} (y_D - m_\theta(x_D)) \\ &\tilde{\sigma}^2 = \sigma_{x_*,x_*}^2 - \Sigma_{x_*,D} \Sigma_{D,D} \Sigma_{D,x_*} \end{aligned} \tag{B.3}$$

Similarly, after imposing a permutation π over the order of elements in the context, we can compute the first and second order of statistics between $D_\pi = [X_{\pi(1:N)}, Y_{\pi(1:N)}]$ and per target point $[x_*, y_*]$.

$$\begin{aligned} &\Sigma_{x_*,D_\pi} = \Sigma_{x_*,D} P_\pi^T = P_\pi \Sigma_{D,x_*} \\ &\Sigma_{D_\pi,D_\pi}^{-1} = P_\pi \Sigma_{D,D}^{-1} P_\pi^T \\ &y_{D_\pi} - m_\theta(x_{D_\pi}) = P_\pi (y_D - m_\theta(x_D)) \end{aligned} \tag{B.4}$$

Hence, with the property of orthogonality of permutation matrix P_π, it is easy to verify the *permutation invariance* in statistics for per target predictive distribution.

$$\begin{aligned} &\Sigma_{x_*,D} \Sigma_{D,D}^{-1} (y_D - m_\theta(x_D)) = \Sigma_{x_*,D_\pi} \Sigma_{D_\pi,D_\pi}^{-1} (y_{D_\pi} - m_\theta(x_{D_\pi})) \\ &\Sigma_{x_*,D} \Sigma_{D,D} \Sigma_{D,x_*} = \Sigma_{x_*,D_\pi} \Sigma_{D_\pi,D_\pi} \Sigma_{D_\pi,x_*} \end{aligned} \tag{B.5}$$

To inherit such a property, NP employs a permutation invariant function in embeddings, and the predictive distribution in NP models is invariant to the order of context points. Also, when there exist multiple target samples in the predictive distribution, it is trivial that the statistics between the context and the target in a GP predictive distribution are *permutation equivariant* in terms of the order of target variables.

C. Proof of DSVNP as Exchangeable Stochastic Process

In the main passage, we formulate the generation of DSVNP as:

$$\rho_{x_{1:N+M}}(y_{1:N+M}) = \iint \prod_{i=1}^{N+M} p(y_i|z_G, z_i, x_i) p(z_i|x_i, z_G) p(z_G) dz_{1:N+M} dz_G \tag{C.1}$$

which indicates the scenario of any finite collection of random variables in ***y***-space. Our intention is to show this induces an exchangeable stochastic process. Equivalently, two conditions for Kolmogorov Extension Theorem are required to be satisfied.

- **Marginalization Consistency.** Generally, when the integral is finite, the swap of orders in integration is allowed. Without exception, **Eq. (C.1)** is assumed to be bounded with some appropriate distributions. Then, for the subset of indexes $\{N+1, N+2, \dots, N+M\}$ in random variables ***y***, we have:

$$\begin{aligned}\int \rho_{x_{1:N+M}}(y_{1:N+M}) dy_{N+1:N+M} &= \iiint \prod_{i=1}^{N+M} p(y_i|z_G, z_i, x_i) \\ &\quad p(z_i|x_i, z_G) p(z_G) dz_{1:N+M} dz_G dy_{N+1:N+M} \\ = \iint \prod_{i=1}^{N} p(y_i|z_G, z_i, x_i) p(z_i|x_i, z_G) \big[&\iint \prod_{i=N+1}^{N+M} p(y_i|z_G, z_i, x_i) p(z_i|x_i, z_G) \\ &\quad dy_{N+1:N+M} dz_{N+1:N+M} \big] p(z_G) dz_G dz_{1:N} \\ = \iint \prod_{i=1}^{N} p(y_i|z_G, z_i, x_i) p(z_i|x_i, z_G) &p(z_G) dz_{1:N} dz_G = \rho_{x_{1:N}}(y_{1:N})\end{aligned} \tag{C.2}$$

  hence, the marginalization consistency is verified.

- **Exchangeability Consistency.** For any permutation π towards the index set $\{1, 2, \dots, N\}$, we have:

$$\begin{aligned}\rho_{x_{1:N}}(y_{1:N}) &= \iint \prod_{i=1}^{N} p(y_i|z_G, z_i, x_i) p(z_i|x_i, z_G) p(z_G) dz_{1:N} dz_G \\ &= \iint \prod_{i=1}^{N} \big[p(y_{\pi_i}|z_G, z_{\pi_i}, x_{\pi_i}) p(z_{\pi_i}|x_{\pi_i}, z_G) dz_{\pi_i} \big] p(z_G) dz_G \\ = \iint \prod_{i=1}^{N} p(y_{\pi_i}|z_G, z_{\pi_i}, x_{\pi_i}) &p(z_{\pi_i}|x_{\pi_i}, z_G) p(z_G) dz_{\pi(1:N)} dz_G = \rho_{x_{\pi(1:N)}}(y_{\pi(1:N)})\end{aligned} \tag{C.3}$$

  hence, the exchangeability consistency is demonstrated as well.

With properties in **Eq. (C.2)** and **(C.3)**, our proposed DSVNP is an exchangeable stochastic process in this case.

D. Derivation of Evidence Lower Bound with Doubly Stochastic Variational Inference

Akin to vanilla NPs, we assume the existence of a global latent variable z_G, which captures summary statistics consistent between the context $[x_C, y_C]$ and the complete target $[x_T, y_T]$. With the involvement of an approximate distribution $q\big(z_G|[x_C, y_C, x_T, y_T]\big)$, we can naturally have an initial ELBO in the following form.

$$\begin{aligned}&\ln \big[p(y_*|x_C, y_C, x_*)\big] = \ln \Big[\mathbb{E}_{q(z_G|x_C, y_C, x_T, y_T)} p(y_*|z_G, x_*) \frac{p(z_G|x_C, y_C)}{q(z_G|x_C, y_C, x_T, y_T)}\Big] \\ &\geq \mathbb{E}_{q(z_G|x_C, y_C, x_T, y_T)} \ln \big[p(y_*|z_G, x_*)\big] - D_{KL}\big[q(z_G|x_C, y_C, x_T, y_T) \parallel p(z_G|x_C, y_C)\big]\end{aligned} \tag{D.1}$$

Note that in **Eq. (D.1)**, the conditional prior distribution $p(z_G|x_C, y_C)$ is intractable in practice and the approximation is used here and such a prior is employed to infer the global latent variable in testing processes. For the approximate posterior

$q(z_G|x_C, y_C, x_T, y_T)$, it makes use of the context and the full target information, and the sample $[x_*, y_*]$ is just an instance in the full target.

Further, by introducing a target specific local latent variable z_*, we can derive another ELBO for the prediction term in the right side of **Eq. (D.1)** with the same trick.

$$\begin{aligned}
&\mathbb{E}_{q(z_G|x_C,y_C,x_T,y_T)} \ln \left[p(y_*|z_G, x_*)\right] \\
&= \mathbb{E}_{q(z_G|x_C,y_C,z_T,y_T)} \ln \left[\mathbb{E}_{q(z_*|z_G,[x_*,y_*])} p(y_*|z_G, z_*, x_*) \frac{p(z_*|z_G, x_*)}{q(z_*|z_G, [x_*, y_*])}\right] \geq \\
&\mathbb{E}_{q(z_G|x_C,y_C,x_T,y_T)} \mathbb{E}_{q(z_*|z_G,[x_*,y_*])} \ln \left[p(y_*|z_G, z_*, x_*)\right] \\
&- \mathbb{E}_{q(z_G|x_C,y_C,x_T,y_T)} \left[D_{KL}[q(z_*|z_G, [x_*, y_*]) \parallel p(z_*|z_G, x_*)]\right]
\end{aligned} \tag{D.2}$$

With the combination of **Eq. (D.1)** and **(D.2)**, the final ELBO $\mathcal{L}$ as the right term in the following is formulated.

$$\begin{aligned}
&\ln \Big[\underbrace{p(y_*|x_C, y_C, x_*)}_{implicit\ data\ likelihood}\Big] \geq \mathbb{E}_{q_{\phi_1(z_G)}} \mathbb{E}_{q_{\phi_2(z_*)}} \ln \Big[\underbrace{p(y_*|z_G, z_*, x_*)}_{data\ likelihood}\Big] \\
&- \mathbb{E}_{q_{\phi_1(z_G)}} \Big[D_{KL}[q(z_*|z_G, x_*, y_*) \parallel \underbrace{p(z_*|z_G, x_*)}_{local\ prior}]\Big] - D_{KL}\Big[q(z_G|x_C, y_C, x_T, y_T) \parallel \underbrace{p(z_G|x_C, y_C)}_{global\ prior}\Big]
\end{aligned} \tag{D.3}$$

The real data likelihood is generally implicit, and the ELBO is an approximate objective. Note that the conditional prior distribution in **Eq. (D.3)**, $p(z_*|z_G, x_*)$ functions as a local latent variable and is approximated with a Gaussian distribution for the sake of easy implementation. With reparameterization trick, used as: $z_G = \mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}$ and $z_* = \mu_{\phi_2} + \epsilon_2 \sigma_{\phi_2}$, we can estimate the gradient towards the sample (x_*, y_*) analytically in **Eq. (D.4)**, **(D.5)** and **(D.6)**.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \phi_1} &= \mathbb{E}_{\epsilon_1 \sim N(0,I)} \mathbb{E}_{\epsilon_2 \sim N(0,I)} \frac{\partial}{\partial \phi_1} \ln \left[p(y_*|\mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}, \mu_{\phi_2} + \epsilon_2 \sigma_{\phi_2}, x_*)\right] \\
&- \mathbb{E}_{\epsilon_1 \sim N(0,I)} \frac{\partial}{\partial \phi_1} D_{KL}\left[q(\mu_{\phi_2} + \epsilon_2 \sigma_{\phi_2}|\mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}, x_*, y_*) \parallel p(\mu_{\phi_2} + \epsilon_2 \sigma_{\phi_2}|\mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}, x_*)\right] \\
&- \mathbb{E}_{\epsilon_1 \sim N(0,I)} \frac{\partial}{\partial \phi_1} D_{KL}\left[q(\mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}|x_C, y_C, x_T, y_T) \parallel p(\mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}|x_C, y_C)\right]
\end{aligned} \tag{D.4}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \phi_2} &= \mathbb{E}_{\epsilon_1 \sim N(0,I)} \mathbb{E}_{\epsilon_2 \sim N(0,I)} \frac{\partial}{\partial \phi_2} \ln \left[p(y_*|\mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}, \mu_{\phi_2} + \epsilon_2 \sigma_{\phi_2}, x_*)\right] \\
&- \mathbb{E}_{\epsilon_1 \sim N(0,I)} \frac{\partial}{\partial \phi_2} D_{KL}\left[q(\mu_{\phi_2} + \epsilon_2 \sigma_{\phi_2}|\mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}, x_*, y_*) \parallel p(\mu_{\phi_2} + \epsilon_2 \sigma_{\phi_2}|\mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}, x_*)\right]
\end{aligned} \tag{D.5}$$

$$\frac{\partial \mathcal{L}}{\partial \theta} = \mathbb{E}_{\epsilon_1 \sim N(0,I)} \mathbb{E}_{\epsilon_2 \sim N(0,I)} \frac{\partial}{\partial \theta} \ln \left[p_\theta(y_*|\mu_{\phi_1} + \epsilon_1 \sigma_{\phi_1}, \mu_{\phi_2} + \epsilon_2 \sigma_{\phi_2}, x_*)\right] \tag{D.6}$$

E. Implementation Details in Experiments

Unless explicitly mentioned, otherwise we make of an one-step amortized transformation as $dim\_lat \mapsto [\mu\_lat, \ln \sigma\_lat]$ to approximate parameters of the posterior in NP models. Especially for DSVNP, the approximate posterior of a local latent variable is learned with the neural network transformation in the approximate posterior $[dim\_lat, dim\_latx, dim\_laty] \mapsto dim\_lat$ and in the piror network $[dim\_lat, dim\_latx] \mapsto dim\_lat$. (For the sake of simplicity, these are not further mentioned in tables of neural structures.) All models are trained with Adam (Kingma & Ba, 2014), implemented on Pytorch.

E.1. Synthetic Experiments

For synthetic experiments, all implementations resemble that in (Kim et al., 2019) [1]. And the neural structures for NPs are reported in Table (3), where dim_lat is 128. Note that for the amortized transformations in encoders of NP, AttnNP

[1]https://github.com/deepmind/neural-processes

Table 1. Pointwise Average Negative Log-likelihoods for 2000 realisations. Rows with J consider all data points including the context, while those with P exclude the context points in statistics. (Figures in brackets are variances.)

PREDICTION	CNP	NP	ATTNNP	DSVNP
INTER(J)	NAN	-0.958(2E-5)	**-1.149(8E-6)**	-0.975(2E-5)
INTER(P)	-0.802(1E-6)	-0.949(2E-5)	**-1.141(6E-6)**	-0.970(2E-5)
EXTRA(J)	NAN	8.192(7E1)	8.091(7E2)	**4.203(9E0)**
EXTRA(P)	**1.764(1E-1)**	8.573(8E1)	8.172(7E2)	4.303(1E1)

Table 2. Tested Entropies of Logit Probability on Classification Dataset. For rows of MNIST and CIFAR10, the second figures in columns are classification accuracies. Both MC-Dropout and DSVNP are averaged with 100 Monte Carlo samples.

	NN	MC-DROPOUT	CNP	NP	ATTNNP	DSVNP
MNIST	0.011/0.990	**0.009/0.993**	0.019/0.993	0.010/0.991	0.012/0.989	0.027/0.990
FMNIST	0.385	0.735	0.711	0.434	0.337	**0.956**
KMNIST	0.282	0.438	0.497	0.322	0.294	**0.545**
GAUSSIAN	0.601	**1.623**	1.313	0.588	0.611	0.966
UNIFORM	0.330	**1.739**	0.862	0.094	0.220	0.375
CIFAR10	0.151/0.768	0.125/0.838	0.177/0.834	0.124/0.792	0.124/0.795	**0.081/0.863**
SVHN	0.402	0.407	**0.459**	0.315	0.269	0.326
RADEMACHER	0.021	0.062	0.079	0.078	0.010	**0.146**
GAUSSIAN	0.351	0.266	**0.523**	0.451	0.349	0.444
UNIFORM	0.334	0.217	**0.499**	0.463	0.261	0.374

and DSVNP, we use the network to learn the distribution parameters as: $dim\_lat \mapsto [\mu_{lat}, \ln \sigma_{lat}]$. In training process, the maximum number of iterations for all (C)NPs is 800k,, and the learning rate is 5e-4. For testing process, in interpolation tasks, the maximum number of context points is 50, while that in extrapolation tasks is 200. Note that the coefficient for KL divergence terms in (C)NPs is set 1 as default, but for DSVNP, we assign more penalty to KL divergence term of local latent variable to avoid overfitting, where the weight is set $\beta_2 = 1000$ for simplicity. Admittedly, more penalty to such term reduces prediction accuracy and some dynamically tuning such parameter would bring some promotion in accuracy.

E.2. System Identification Experiments

In the Cart-Pole simulator, the input of the system is the vector of the coordinate/angle and their first order derivative and a random action $[x_c, \theta, x_c', \theta', a]$, while the output is the transited state in the next time step $[x_c, \theta, x_c', \theta']$. The force as the action space ranges between [-10,10] N, where the action is a randomly selected force value to impose in the system. For training dataset from 6 configurations of environments, we sample 100 state transition pairs for each configuration as the maximum context points, and these context points work as identification of a specific configuration. The neural architectures for CNP, NP, AttnNP and DSVNP refer to Table (4), and default parameters are listed in $\{dim\_latxy = 32, dim\_lat = 32, dim\_h = 400\}$. All neural network models are trained with the same learning rate 1e-3. The batch size and the maximum number of epochs are 100. For AttnNP, we notice the generalization capability degrades with training process, so early stopping is used. For DSVNP, the weight of regularization is set as $\{\beta_1 = 1, \beta_2 = 5\}$, while the KL divergence term weight is fixed as 1 for NP and AttnNP.

E.3. Multi-output Regression Experiments

SARCOS records inverse dynamics for an anthropomorphic robot arm with seven degree freedoms, and the mission is to predict 7 joint torques with 21-dimensional state space (7 joint positions, 7 joint velocities and 7 accelerations). WQ targets at predicting the relative representation of plant and animal species in Slovenian rivers with some physical and chemical water quality parameters. SCM20D is some supply chain time series dataset for many products, while SCFP records some online click logs for several social issues.

Before data split, standardization over input and output space is operated on dataset, scaling each dimension of dataset in zero

Table 3. Neural Network Structure of (C)NP Models for 1-D Stochastic Process. The transformations in the table are in linear form, followed with ReLU activation mostly. And Dropout rate for DNN is defined as 0.5 for all transformation layers in Encoder. As for AttnNP and DSVNP, the encoder network is doubled in the table since there exist some local variable for prediction.

NP MODELS	ENCODER	DECODER
CNP&NP	$[dim\_x, dim\_y] \mapsto 32 \mapsto 32 \mapsto dim\_lat$	$[dim\_x, dim\_lat] \mapsto 2 * dim\_y$
ATTNNP&DSVNP	$[dim\_x, dim\_y] \mapsto 32 \mapsto 32 \mapsto dim\_lat$	$[dim\_x, 2 * dim\_lat] \mapsto 2 * dim\_y$

Table 4. Neural Network Structure of (C)NP Models in System Identification Tasks. The transformations in the table are linear, followed with ReLU activation mostly. As for AttnNP and DSVNP, the encoder network is doubled in the table since there exist some local variable for prediction. Here only dot product attention is used in AttnNP.

NP MODELS	ENCODER	DECODER
CNP&NP	$[dim\_x, dim\_y] \mapsto \underbrace{dim\_latxy \mapsto dim\_latxy}_{2\ times}$ $dim\_latxy \mapsto dim\_lat.$	$[dim\_x, dim\_lat] \mapsto dim\_h \mapsto dim\_h$ $dim\_h \mapsto 2 * dim\_y$
ATTNNP&DSVNP	$[dim\_x, dim\_y] \mapsto \underbrace{dim\_latxy \mapsto dim\_latxy}_{2\ times}$ $dim\_x \mapsto dim\_latx;$ $[dim\_latx, dim\_laty] \mapsto dim\_lat.$	$[dim\_x, 2 * dim\_lat] \mapsto dim\_h \mapsto dim\_h$ $dim\_h \mapsto 2 * dim\_y$

mean and unit variance [2]. Such pre-processing is required to ensure the stability of training. Also, we find directly treating the data likelihood term as some Gaussian and optimizing negative log likelihood of Gaussian to learn both mean and variance do harm to the prediction, hence average MSE is selected as the objective. As for the variance estimation for uncertainty, Monte Carlo estimation can be used. For all dataset, we employ the neural structure in Table (5), and default parameters in Encoder and Decoder are in the list $\{dim\_h = 100, dim\_latx = 32, dim\_laty = 8, dim\_lat = 64\}$. The learning rate for Adam is selected as 1e-3, the batch size for all dataset is 100, the maximum number of context points is randomly selected during training, and the maximum epochs in training are up to the scale of dataset and convergence condition. Here the maximum epochs are respectively 300 for SARCOS, 3000 for SCM20D and 5000 for WQ. For the testing process, 30 data points are randomly selected as the context for prediction in each batch. Also notice that, the hyper-parameters as the weights of KL divergence term are the same in implementation as one without additional modification in this experiments.

E.4. Image Classification and O.O.D. Detection

The implementations of NP related models and Monte-Carlo Neural Network are quite similar. On MNIST task, the feature extractor for images is taken as LeNet-like structure as [20, 'M', 50, 'F', '500'][3], and the decoder is one-layer transformation. On CIFAR10 task, the extractor is parameterized in VGG-style network as [64, 64, 'M', 128, 128, 'M', 256, 256, 256, 256, 'M', 512, 512, 512, 512, 'M', 512, 512, 512, 512, 'M'][4], and the decoder is also one-layer transformation from latent variable to label output in softmax form. Other parameters are in the list $\{dim\_latx = 32, dim\_laty = 64, dim\_lat = 64\}$ for MNIST and $\{dim\_latx = 32, dim\_laty = 64, dim\_lat = 128\}$. The labels for both are represented in one-hot encoding way and then further transform to some continuous embedding. Batch size in training is 100 as default, the number of context samples for NP related models is randomly selected no larger than 100 in each batch, while the optimizer Adam is with learning rate $1e^{-3}$ for MNIST task and $5e^{-5}$ for CIFAR10 task. The maximum epochs for both is 100 in both cases, and the size of all source and o.o.d. dataset is 10000. Dropout rates for MC-Dropout in encoder networks are respectively as 0.1 and 0.2 for LeNet-like one and VGG-like one. In the testing process, 100 samples from source dataset are randomly selected as the context points.

[2]https://scikit-learn.org/stable/modules/preprocessing.html

[3]Numbers are dimensions of Out-Channel with kernel size 5, 'F' is flattening operation, and each layer is followed with ReLU activation.

[4]Numbers are dimensions of Out-Channel with kernel size 3 and padding 1 in each layer, followed with BatchNorm and ReLU function, here M means max-polling operation.

Table 5. Neural Network Structure of (C)NP Models in Multi-Output Regression Tasks. The transformations in the table are linear, followed with ReLU activation mostly. And Dropout rate for DNN is defined as 0.01 for all transformation layers in Encoder. As for AttnNP and DSVNP, the encoder network is doubled in the table since there exist some local variable for prediction. Here only dot product attention is used in AttnNP.

NP MODELS	ENCODER	DECODER
DNN(MC-DROPOUT)	$dim\_x \mapsto \underbrace{dim\_h \mapsto dim\_h}_{2\ times}$	$dim\_lat \mapsto dim\_h$
	$dim\_h \mapsto dim\_lat$	$dim\_h \mapsto dim\_y$
	$dim\_x \mapsto \underbrace{dim\_h \mapsto dim\_h}_{2\ times} \mapsto dim\_latx$	$[dim\_latx, dim\_lat] \mapsto dim\_h$
CNP&NP	$dim\_y \mapsto dim\_laty;$	
	$[dim\_latx, dim\_laty] \mapsto dim\_lat.$	$dim\_lat \mapsto dim\_y$
	$dim\_x \mapsto \underbrace{dim\_h \mapsto dim\_h}_{2\ times} \mapsto dim\_latx$	$[dim\_latx, 2 * dim\_lat] \mapsto dim\_h$
ATTNNP&DSVNP	$dim\_y \mapsto dim\_laty;$	
	$[dim\_latx, dim\_laty] \mapsto dim\_lat.$	$dim\_lat \mapsto dim\_y$

Table 6. Neural Network Structure of (C)NP Models in Image Classification Tasks. The transformations in the table are linear, followed with ReLU activation mostly. And Dropout rate for DNN is defined as 0.5 for all transformation layers in Encoder. As for AttnNP and DSVNP, the encoder network is doubled in the table since there exist some local variable for prediction.

NP MODELS	ENCODER	DECODER
DNN(MC-DROPOUT)	$\underbrace{dim\_x \mapsto dim\_h}_{embedding\ net}$	$dim\_lat \mapsto dim\_y$
	$dim\_h \mapsto dim\_lat$	
	$\underbrace{dim\_x \mapsto dim\_h}_{embedding\ net} \mapsto dim\_latx$	
CNP&NP	$dim\_y \mapsto dim\_laty;$	$[dim\_latx, dim\_lat] \mapsto dim\_y$
	$[dim\_latx, dim\_laty] \mapsto dim\_lat.$	
	$\underbrace{dim\_x \mapsto dim\_h}_{embedding\ net} \mapsto dim\_latx$	
ATTNNP&DSVNP	$dim\_y \mapsto dim\_laty;$	$[dim\_latx, 2 * dim\_lat] \mapsto dim\_y$
	$[dim\_latx, dim\_laty] \mapsto dim\_lat.$	

Before prediction process (estimating predictive entropies on both domain dataset and o.o.d dataset), images on MNIST are normalized in interval $[0, 1]$, those on CIFAR10 are standarized as well, and all o.o.d. dataset follow similar way as that on MNIST or CIFAR10. More specifically, Rademacher dataset is generated in the way: place bi-nominal distribution with probability 0.5 over in image shaped tensor and then minus 0.5 to ensure the zero-mean in statistics. Similar operation is taken in uniform cases, while Gaussian o.o.d. dataset is from standard Normal distribution. All results are reported in Table (2).

References

Giles, C. L. and Maxwell, T. Learning, invariance, and generalization in high-order neural networks. *Applied optics*, 26(23): 4972–4978, 1987.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019.

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

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753, 2019.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.