Exploiting Inferential Structure in Neural Processes

Dharmesh Tailor[1] **Mohammad Emtiyaz Khan**[2] **Eric Nalisnick**[1]

[1]University of Amsterdam, Amsterdam, Netherlands
[2]RIKEN Center for AI Project, Tokyo, Japan

Abstract

Neural processes (NPs) can be extremely fast at test time, but their training requires a wide range of context sets to generalize well. We propose to address this issue by incorporating the structure of graphical models into NPs. This leads to aggregation strategies in which context points are appropriately weighted, generalizing a recent proposal by Volpp et al. [2020]. The weighting further reveals an interpretation of each point, which we refer to as the neural sufficient statistics. It is expected that by exploiting information in structured priors, the data inefficiency of NPs can be alleviated.

1 INTRODUCTION

Many real-world tasks require models to make predictions in new scenarios on short notice. For example, climate models are often asked to make predictions at novel locations [Vaughan et al., 2022]. Data is collected in well-populated regions but predictions for remote regions (e.g. mountain ranges, forests) are desirable as well. *Neural processes* (NPs) [Garnelo et al., 2018] are models designed for situations such as this. At test time, the model is seeded with a context data set from a novel setting that (hopefully) allows the NP to make accurate predictions in the new setting. Moreover, the NP's predictive distribution is efficient to compute, scaling linearly with respect to the size of the context set. These properties make the NP a tractable model for online and adaptive predictive inference.

However, the ability for NPs to generalize to novel contexts is made possible only through intensive training. Usually many, many different context sets must be shown to the model.[1] We address this issue by endowing the NP with a *structured inference network* [Johnson et al., 2016]. This change is beneficial for several reasons: (i) the datapoint-wise representations of the encoder architecture now have a clear interpretation as neural sufficient statistics, (ii) the aggregation strategy naturally follows from the set of probabilistic assumptions, and (iii) structured priors are straightforward to incorporate. We demonstrate that a recent proposal by [Volpp et al., 2020] is recovered as a special case with Gaussian prior assumptions.

2 BACKGROUND

Data NPs assume a partition of the data into a *context* set and a *target* set. The former is used by the model to seed adaptation. The latter is a set of points from the same domain as the context set and for which we will make predictions. Specifically, we denote the context set for the lth task as $\mathcal{D}_{l,c} = \{\mathbf{x}_{l,c}^{(i)}, y_{l,c}^{(i)}\}_{i=1}^{N_{l,c}}$, where $\mathbf{x}$ denotes a feature vector and y the corresponding response. The target set for the lth task is denoted similarly as $\mathcal{D}_{l,t} = \{\mathbf{x}_{l,t}^{(i)}, y_{l,t}^{(i)}\}_{i=1}^{N_{l,t}}$. At test time, for a new task l^*, we observe $\mathcal{D}_{l^*,c}$ and $\{\mathbf{x}_{l^*,t}^{(i)}\}_{i=1}^{N_{l,t}}$. The target responses $\{y_{l^*,t}^{(i)}\}_{i=1}^{N_{l,t}}$ are unobserved, and our goal is to predict them.

Neural Processes *Neural processes* [Garnelo et al., 2018] frame few-shot learning as a multi-task learning problem [Heskes, 2000], employing a conditional latent variable model with context/target splits on task-specific datasets. Training amounts to maximising the following *conditional* marginal likelihood across L tasks:

$$\begin{aligned}\ell(\theta) &= \prod_{l=1}^{L} p_\theta(\mathcal{D}_{l,t} \,|\, \mathcal{D}_{l,c}) \\ &= \prod_{l=1}^{L} \int p_\theta(\mathcal{D}_{l,t} \,|\, z_l)\; p_\theta(z_l \,|\, \mathcal{D}_{l,c})\; dz_l,\end{aligned} \tag{1}$$

[1]Training the (conditional) NP to generalize over one dimensional samples from a Gaussian process requires 200,000 training iterations, each having 64 context sets containing 3 to 10 data points.

Accepted for the 5th Workshop on Tractable Probabilistic Modeling at UAI (TPM 2022).

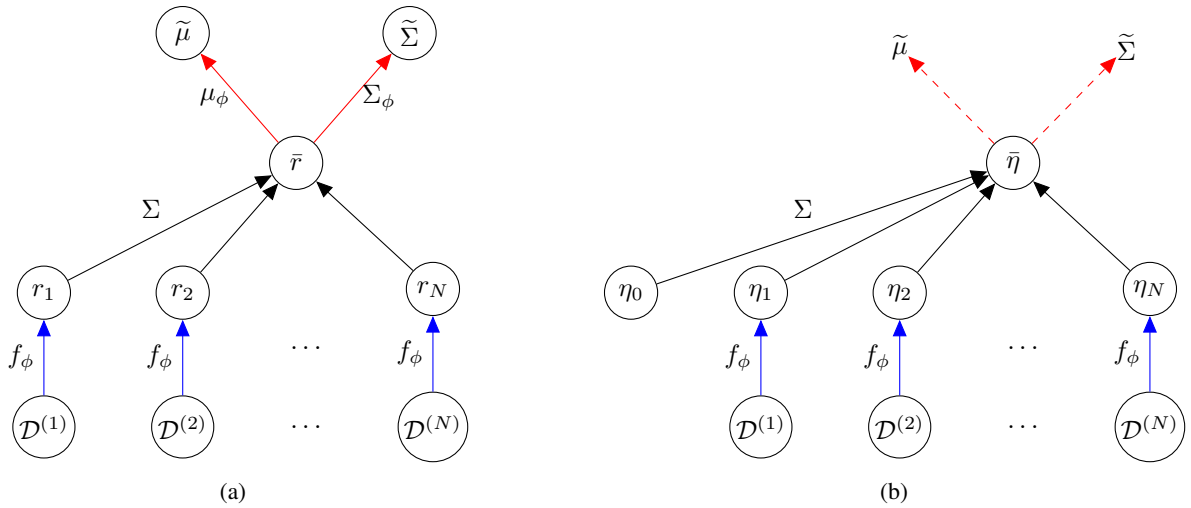


Figure 1: Computational graph of (a) sum-decomposition network and (b) structured inference network.

where z_l is the task-specific latent variable and θ is the global parameter that is shared across tasks. The marginalization over task-specific latent variables is typically intractable hence approximate inference is required:

$$p_\theta(z \mid \mathcal{D}_c) = \frac{1}{p_\theta(\mathcal{D}_c)} \prod_{i=1}^{N_c} p_\theta(\mathcal{D}_c^{(i)} \mid z)\, p_\theta(z) \tag{2}$$

$$\approx q_\phi(z \mid \mathcal{D}_c). \tag{3}$$

We have dropped task indices for the sake of notational simplicity. The variational approximation is amortised, meaning we will parameterize the local approximations with an inference model. For a Gaussian approximation, the mean and variance are parameterized by neural networks (NNs) that take as input sets of datapoints: $q_\phi(z \mid \mathcal{D}_c) = \mathrm{N}(z \mid \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\Sigma}})$ with $\widetilde{\boldsymbol{\mu}} = \mathrm{enc}_\phi^m(\mathcal{D}_c)$ and $\widetilde{\boldsymbol{\Sigma}} = \mathrm{enc}_\phi^v(\mathcal{D}_c)$. Throughout we assume covariance matrices have diagonal structure, resulting in factorized Gaussian distributions.

Sum-Decomposition Networks The inference networks for NPs must have at least two properties. The first is that they make no assumptions about the size of the context set. The second is that the encoder be invariant to the ordering of context points. A common way to satisfy these criteria is by having the encoder take the form of a sum-decomposition network [Edwards and Storkey, 2017, Zaheer et al., 2017]:

$$\mathbf{r}_i = f_\phi(\mathcal{D}_c^{(i)}), \quad \bar{\mathbf{r}} = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbf{r}_i \tag{4}$$

where $\mathbf{r}_i$ are datapoint-wise encodings given by a NN which are then aggregated. The aggregation operation is typically taken to be a simple average in NPs but other operators are valid as long as they are permutation-invariant. Thus the amortisation goes one level further with parameter sharing across context points. Finally, the aggregated representation $\bar{\mathbf{r}}$ is passed to further NNs to give the variational parameters. In the case of Gaussian posterior, there are two further NNs predicting the mean and variance as illustrated in Fig. 1a: $\widetilde{\boldsymbol{\mu}} = \mu_\phi(\bar{\mathbf{r}})$, $\widetilde{\boldsymbol{\Sigma}} = \Sigma_\phi(\bar{\mathbf{r}})$.

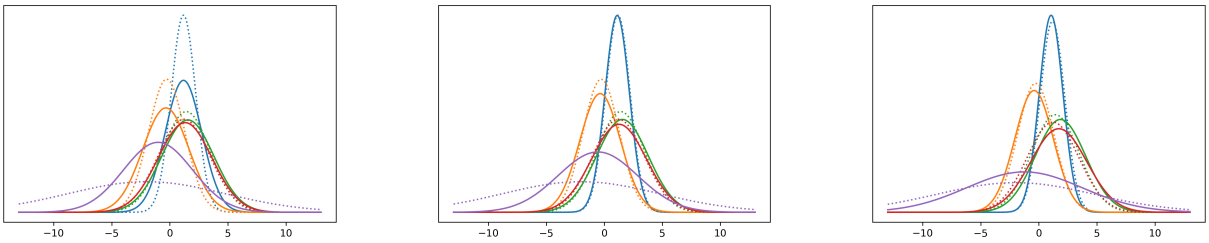
3 STRUCTURED INFERENCE NETWORKS

Despite the flexibility of the sum-decomposition architecture, its use is more for computational convenience than due to a strong connection to the variational approximation (Eq. (2)). The computation is clearly localized to each data point, similar to the factorization of the likelihood, but the additional NN-based transform make them uninterpretable. Furthermore, there is no explicit presence of a prior (whose underlying structure could be exploited).

We describe our approach that incorporates the structure of the graphical model into the encoder whilst performing fast amortized inference. In particular, we replace the sum-decomposition network with a *structured inference network* [Lin et al., 2018, Johnson et al., 2016] that reflects the likelihood's factorization (Eq. (2)) and the presence of a prior. We write the posterior approximation as:

$$q_\phi(z \mid \mathcal{D}_c) = \frac{1}{Z_c(\phi)} \underbrace{\left[\prod_{i=1}^{N_c} q(z \mid f_{\phi_{\text{NN}}}(\mathcal{D}_c^{(i)}))\right]}_{\text{NN factor}} \underbrace{\left[q(z; \phi_{\text{PGM}})\right]}_{\text{prior}}$$

where $Z_c(\phi)$ is the normalisation constant. The NN factors are constructed to be conjugate to the prior in order to preserve tractability. In turn the variational posterior can be evaluated by *conjugate computations*, i.e. adding natural parameters arising from the N_c predictions of the inference



(a) Linear (b) Quadratic (c) MLP: 2 hidden layers of 20 units

Figure 2: Structured inference network (solid curves) expressiveness to model Gaussian regression likelihood (dashed curves) accurately (see App. A.4 for details).

network and the prior:

$$q_\phi(z \mid \mathcal{D}_c) \propto \exp\left\{\left\langle \boldsymbol{\eta}_{\phi_{\text{PGM}}} + \sum_{i=1}^{N_c} \boldsymbol{\eta}\left(f_{\phi_{\text{NN}}}(\mathcal{D}_c^{(i)})\right), \mathbf{T}(\mathbf{z})\right\rangle\right\} \quad (5)$$

where $\boldsymbol{\eta}_{\phi_{\text{PGM}}}$ is the natural parameters of the prior distribution, $\boldsymbol{\eta}(\cdot)$ is the natural parameters of each NN factor and $\mathbf{T}(\mathbf{z})$ is the sufficient statistics. The computation is illustrated in Fig. 1b. The NN factors may not necessarily be in canonical form. In the case of Gaussian factors, $f_{\phi_{\text{NN}}}(\cdot)$ may output the mean and variance, and therefore we use $\boldsymbol{\eta}(\cdot)$ to indicate transformation to the natural parameterization.

Due to the close resemblance to computations in an actual conjugate system—where the natural parameters of the posterior are obtained by adding the sufficient statistics of the likelihood to the prior natural parameters—we can interpret the NN factors as *neural sufficient statistics* [Wu et al., 2020]. This is further supported by considering the VI objective used to train NPs. After substituting the structured inference network and simplifying, the resulting objective contains a term that resembles the entropy on the individual NN factors (for full derivation see App. A.1). Given the link between statistical sufficiency and information-maximizing representations of the data, this suggests the NN factors are approximating the true sufficient statistics.

These structured inference networks have several important differences from the sum-decomposition network. The first is the level at which aggregation is performed. Instead of having the intermediate representation $\bar{\mathbf{r}}$ (Eq. (4))—which has no interpretation—the encoder produces the NN factors. The variational parameters are then computed directly from these factors instead of having to be computed by a black-box NN-based transform. Not only does this aid in interpretability, but it also results in fewer NN parameters to estimate. The second change is the introduction of an explicit prior distribution whose parameters are aggregated along with the datapoint-wise representations. The third change is the aggregation strategy is determined directly from the parameterization chosen for the exponential-family. If operating in natural parameters (as shown above), the aggregation is a simple sum—thus recovering mean aggregation. However, as we will demonstrate below, other parameterizations lead to non-trivial pooling operations.

3.1 GAUSSIAN PRIOR

Before we show how a structured prior can be exploited, we first demonstrate the framework for the case of a Gaussian prior. The resulting conjugate exponential-family distribution for the NN factors is also Gaussian:

$$q_\phi(z \mid \mathcal{D}_c) = \frac{1}{Z_c(\phi)} \prod_{i=1}^{N_c} \mathrm{N}(z \mid \mathbf{m}_i, \mathbf{V}_i)\ \mathrm{N}(z \mid \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \quad (6)$$

where $\mathbf{m}_i = f^{(1)}_{\phi_{\text{NN}}}(\mathcal{D}_c^{(i)})$ and $\mathbf{V}_i = f^{(2)}_{\phi_{\text{NN}}}(\mathcal{D}_c^{(i)})$ are the mean and variance parameterized by a NN. To derive the posterior, we can swap z and $\mathbf{m}_i$ in the NN factor and apply the rules for Gaussian conditioning resulting in,

$$q_\phi(z \mid \mathcal{D}_c) = \mathrm{N}(z; \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\Sigma}}), \quad \text{with} \quad (7)$$

$$\widetilde{\boldsymbol{\Sigma}}^{-1} = \sum_{i=1}^{N_c} \mathbf{V}_i^{-1} + \boldsymbol{\Sigma}_0^{-1} \quad (8)$$

$$\widetilde{\boldsymbol{\mu}} = \widetilde{\boldsymbol{\Sigma}}\left(\sum_{i=1}^{N_c} \mathbf{V}_i^{-1}\mathbf{m}_i + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right). \quad (9)$$

The normalisation constant $Z_c(\phi)$ is also available in closed-form and can be evaluated by a completing-the-squares technique. Since we assume diagonal covariance matrices, the computations remain tractable, i.e. no expensive matrix operations, but this assumption could be relaxed.

Equivalence to Bayesian Aggregation The Gaussian-based procedure above is equivalent to *Bayesian context aggregation* [Volpp et al., 2020] (when the prior is fixed). Their approach was motivated by introducing a surrogate Gaussian CLV model with the local datapoint-wise encodings being

interpreted as noisy observations of the underlying latent variable. They give the expression for the mean of Gaussian posterior in the form of an incremental update to the prior. We show equivalence to Eq. (14) in App. A.2. They discuss the relationship of Eq. (14) to mean aggregation when a non-informative prior is imposed along with uniform observation variances. In addition to this, an interpretation of Eq. (14) as a weighted average of datapoint-wise encodings drawing connections to self-attention mechanisms in neural processes [Kim et al., 2019].

Simulation Study To verify the importance of the functional form of the NN factors, we perform a simulation study with a simple Gaussian model that admits exact posterior computations. We train NPs on one-dimensional data generated from a hierarchical linear model with Gaussian likelihood $y_l^{(i)} \sim \mathrm{N}(x_l^{(i)} z_l, 1)$ and Gaussian prior $z_l \sim \mathrm{N}(0, 5)$. Inputs are generated according to $x_{(l)}^{(i)} \sim U(0, 1)$. The true sufficient statistics of this problem take the form: $[(xy)/\sigma_y^2,\ -x^2/(2\sigma_y^2)]$. Therefore the structured inference network must be sufficiently expressive to represent a quadratic function of the data in order to learn the true sufficient statistics. We consider three NPs, each with a structured inference network but with three variations of f_ϕ: (1) linear; (2) linear but with polynomial basis expansion of degree 2; (3) MLP with 2 hidden layers each with 20 units. The linear model should only be able to capture the location whereas the degree-two polynomial and MLP should be sufficiently expressive to represent location and scale. The results of the simulation are shown in Fig. 2. Indeed, the curvature is poorly approximated in the linear case (subfigure a). Moreover, there is no clear advantage to using the NN (subfigure c), which is overly expressive.

3.2 MIXTURE OF GAUSSIAN PRIOR

We now consider a structured prior such as mixture of Gaussian which is a conditionally-conjugate exponential-family distribution. This may a beneficial modelling assumption if we expect the data to arise from multiple sources. The prior is given by,

$$q(z; \phi_{\text{PGM}}) = \sum_{k=1}^{K} \pi_k \mathrm{N}(z \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{10}$$

with K component Gaussian distributions and weighting terms such that $\sum_{k=1}^{K} \pi_k = 1$. Taking the DNN factors to be Gaussian and in moment parameterization similar to Sec. 3.1, the posterior can also be expressed as a mixture of Gaussians,

$$q_\phi(z \,|\, \mathcal{D}_c) = \frac{1}{Z_c(\phi)} \prod_{i=1}^{N_c} \mathrm{N}(z \,|\, \mathbf{m}_i, \mathbf{V}_i)\ q(z; \phi_{\text{PGM}}) \tag{11}$$

$$= \frac{1}{Z_c(\phi)} \sum_{k=1}^{K} \widetilde{\pi}_k \mathrm{N}(z \,|\, \widetilde{\boldsymbol{\mu}}_k, \widetilde{\boldsymbol{\Sigma}}_k) \tag{12}$$

with,

$$\widetilde{\boldsymbol{\Sigma}}_k^{-1} = \sum_{i=1}^{N_c} \mathbf{V}_i^{-1} + \boldsymbol{\Sigma}_k^{-1} \tag{13}$$

$$\widetilde{\boldsymbol{\mu}}_k = \widetilde{\boldsymbol{\Sigma}}_k \left(\sum_{i=1}^{N_c} \mathbf{V}_i^{-1} \mathbf{m}_i + \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\mu}_k \right) \tag{14}$$

$$\widetilde{\pi}_k = \pi_k\ C_k \tag{15}$$

where $Z_c = \sum_{k=1}^{K} \widetilde{\pi}_k$ and C_k is given in App. A.3. The aggregation for the mean and variance parameters for each component in the posterior takes an identical form to Eqs. (8) and (14). However, we arrive at a non-trivial expression for the mixing proportions.

4 CONCLUSION

We proposed to improve NPs by incorporating structured inference networks. This change is attractive for several reasons: (i) the local encodings ($\mathbf{r}_i$) now have a clear interpretation as neural sufficient statistics, (ii) the aggregation step is predetermined by and follows from the probabilistic assumptions, and (iii) structured priors are straightforward to incorporate. We demonstrated in a Gaussian simulation that these structural assumptions do indeed matter: the linear encoder was only able to capture the location, as predicted by the form of the true sufficient statistics. In future work, we plan to explore structured inference networks for different priors as well as scale up to large experiments.

References

Harrison Edwards and Amos Storkey. Towards a neural statistician. In *International Conference on Learning Representations*, 2017.

Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

TM Heskes. Empirical bayes for learning to learn. 2000.

Matthew J Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. *Advances in neural information processing systems*, 29, 2016.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *International Conference on Learning Representations*, 2019.

Wu Lin, Nicolas Hubacher, and Mohammad Emtiyaz Khan. Variational message passing with structured inference networks. In *International Conference on Learning Representations*, 2018.

Anna Vaughan, Will Tebbutt, J Scott Hosking, and Richard E Turner. Convolutional conditional neural processes for local climate downscaling. *Geoscientific Model Development*, 15(1):251–268, 2022.

Michael Volpp, Fabian Flürenbrock, Lukas Grossberger, Christian Daniel, and Gerhard Neumann. Bayesian context aggregation for neural processes. In *International Conference on Learning Representations*, 2020.

Hao Wu, Heiko Zimmermann, Eli Sennesh, Tuan Anh Le, and Jan-Willem Van De Meent. Amortized population Gibbs samplers with neural sufficient statistics. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10421–10431. PMLR, 2020.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

A APPENDIX

A.1 ELBO DERIVATION WITH STRUCTURED INFERENCE NETWORK

$$\log p(\mathcal{D}_t \mid \mathcal{D}_c) \tag{16}$$

$$= \log \int_z p(\mathcal{D}_t \mid z)\; p(z \mid \mathcal{D}_c) \tag{17}$$

$$= \log \int_z q_\phi(z \mid \mathcal{D}_c \cup \mathcal{D}_t) \frac{p(\mathcal{D}_t \mid z)\; p(z \mid \mathcal{D}_c)}{q_\phi(z \mid \mathcal{D}_c \cup \mathcal{D}_t)} \tag{18}$$

$$\geq \mathbb{E}_{q_\phi(z \mid \mathcal{D}_c \cup \mathcal{D}_t)} \left[\log \frac{p(\mathcal{D}_t \mid z)\; p(z \mid \mathcal{D}_c)}{q_\phi(z \mid \mathcal{D}_c \cup \mathcal{D}_t)} \right] \tag{19}$$

$$\approx \mathbb{E}_q \left[\log \frac{p(\mathcal{D}_t \mid z)\; q_\phi(z \mid \mathcal{D}_c)}{q_\phi(z \mid \mathcal{D}_c \cup \mathcal{D}_t)} \right] \tag{20}$$

$$= \mathbb{E}_q[\log p(\mathcal{D}_t \mid z)] + \mathbb{E}_q \left[\log \frac{\cancel{\prod_{i=1}^{N_c} q(z \mid f_{\phi_{\text{NN}}}(\mathcal{D}_c^{(i)}))}}{\cancel{\prod_{i=1}^{N_c} q(z \mid f_{\phi_{\text{NN}}}(\mathcal{D}_c^{(i)}))} \prod_{i=1}^{N_t} q(z \mid f_{\phi_{\text{NN}}}(\mathcal{D}_t^{(i)}))} \frac{\cancel{q(z;\phi_{\text{PGM}})}\, Z_{c,t}(\phi)}{\cancel{q(z;\phi_{\text{PGM}})}\, Z_c(\phi)} \right] \tag{21}$$

$$= \mathbb{E}_q \left[\log p(\mathcal{D}_t \mid z)\right] - \sum_{i=1}^{N_t} \mathbb{E}_q \left[\log q(z \mid f_{\phi_{\text{NN}}}(\mathcal{D}_t^{(i)}))\right] + \log Z_{c,t}(\phi) - \log Z_c(\phi) \tag{22}$$

A.2 EQUIVALENCE TO BAYESIAN AGGREGATION MEAN UPDATE EQUATION IN [Volpp et al., 2020]

$$\widetilde{\boldsymbol{\mu}} = \widetilde{\boldsymbol{\Sigma}} \left(\sum_{i=1}^{N_c} \mathbf{V}_i^{-1} \mathbf{m}_i + \boldsymbol{\Sigma}_0^{-1} \boldsymbol{\mu}_0 \right) \tag{23}$$

$$= \widetilde{\boldsymbol{\Sigma}} \left(\sum_{i=1}^{N_c} \mathbf{V}_i^{-1} \mathbf{m}_i + \widetilde{\boldsymbol{\Sigma}}^{-1} \boldsymbol{\mu}_0 - \sum_{i=1}^{N_c} \mathbf{V}_i^{-1} \boldsymbol{\mu}_0 \right) \tag{24}$$

$$= \boldsymbol{\mu}_0 + \widetilde{\boldsymbol{\Sigma}} \sum_{i=1}^{N_c} \mathbf{V}_i^{-1} \left(\mathbf{m}_i - \boldsymbol{\mu}_0 \right) \tag{25}$$

A.3 MIXING QUANTITY FOR MIXTURE OF GAUSSIAN POSTERIOR

$$C_k = (2\pi)^{-\frac{DN}{2}} \prod_{i=1}^{N_c} \det(\mathbf{V}_i)^{-\frac{1}{2}} \left(\frac{\det(\boldsymbol{\Sigma}_k)}{\det(\widetilde{\boldsymbol{\Sigma}}_k)} \right)^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} \left(\sum_{i=1}^{N_c} \mathbf{m}_i^\top \mathbf{V}_i^{-1} \mathbf{m}_i + \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\mu}_k - \widetilde{\boldsymbol{\mu}}_k^\top \widetilde{\boldsymbol{\Sigma}}_k^{-1} \widetilde{\boldsymbol{\mu}}_k \right) \right\} \tag{26}$$

A.4 EXPERIMENTAL DETAILS

We train a NP on 1D data generated from a hierarchical linear model with Gaussian likelihood $y_l^{(i)} \sim \mathrm{N}(x_l^{(i)} z_l, \sigma_y^2)$ and Gaussian prior $z_l \sim \mathrm{N}(\mu_0, \sigma_0^2)$. Inputs are generated according to $x_{(l)}^{(i)} \sim U(x_{\text{min}}, x_{\text{max}})$. The following configuration of parameters are used: $\sigma_y^2 = 1, \mu_0 = 0, \sigma_0^2 = 5, x_{\text{min}} = 0, x_{\text{max}} = 1$. We consider $L = 10000$ tasks with $N_c = 5$ context points and $N_t = 15$ target points, both generated in the same fashion.

We consider the NP with structured inference network in natural parameterization and with Gaussian prior but with 3 varieties for $f_\phi(\cdot)$: (1) linear; (2) linear but with polynomial basis expansion of degree 2; (3) MLP with 2 hidden layers each with 20 units. The NP is fitted using the VI objective via full-batch gradient descent with the Adam optimizer for 10000 epochs and learning rate 10^{-3}. The number of latent samples during training is set to 128 and the decoder is fixed to the true likelihood.