Information Theoretic Meta Learning with Gaussian Processes: Supplementary Material

Michalis K. Titsias[1] **Francisco J. R. Ruiz**[1] **Sotirios Nikoloutsopoulos**[2] **Alexandre Galashov**[1]

[1]DeepMind
[2]Athens Univ. of Econ. and Business, Greece

A FURTHER DETAILS ABOUT VIB IN META LEARNING

A.1 BOUNDS ON THE MUTUAL INFORMATION

Here, we review the standard variational bounds on the mutual information from Barber and Agakov [2003]. Recall the definition of the mutual information,

$$I(x,y) = \int q(x,y)\log\frac{q(x,y)}{q(x)q(y)}dxdy = \int q(x,y)\log\frac{q(x|y)}{q(x)}dxdy.$$

By introducing $p(x|y)$ that approximates $q(x|y)$ we get

$$I(x,y) = \int q(x,y)\log\frac{p(x|y)q(x|y)}{p(x|y)q(x)}dxdy = \int q(x,y)\log\frac{p(x|y)}{q(x)}dxdy + \int q(y)\mathrm{KL}[q(x|y)||p(x|y)]dy,$$

which shows that

$$I(x,y) \geq \int q(x,y)\log\frac{p(x|y)}{q(x)}dxdy, \tag{17}$$

since $\int q(y)\mathrm{KL}[q(x|y)||p(x|y)]dy$ is non negative.

An upper bound is obtained similarly. Suppose $p(x)$ approximates $q(x)$; then

$$I(x,y) = \int q(x,y)\log\frac{p(x)q(x|y)}{p(x)q(x)}dxdy = \int q(x,y)\log\frac{q(x|y)}{p(x)}dxdy - \mathrm{KL}[q(x)||p(x)]dy,$$

which shows that

$$I(x,y) \leq \int q(x,y)\log\frac{q(x|y)}{p(x)}dxdy. \tag{18}$$

A.2 THE GENERAL VIB META LEARNING CASE

Consider the general case, where we work with the unconditional mutual information and we wish to approximate the information bottleneck (IB): $I(Z,D^v) - \beta I(Z,D^t)$. Recall that the joint distribution is written as

$$q_w(D^v,D^t,Z) = q_w(Z|D^t)p(D^v,D^t), \tag{19}$$

from which we can express any marginal or conditional. In particular observe that

$$q_w(Z,D^v) = \int q_w(Z|D^t)p(D^v,D^t)dD^t.$$

If we have a function $f(Z,D^v)$ and we wish to approximate the expectation,

$$\int q_w(Z,D^v)f(Z,D^v)dZdD^v = \int q_w(Z|D^t)p(D^v,D^t)f(Z,D^v)dZdD^vdD^t, \tag{20}$$

then, given that we sample a task pair $(D_i^v,D_i^t) \sim p(D^v,D^t)$, we can obtain the following unbiased estimate of this expectation,

$$\int q_w(Z|D_i^t)f(Z,D_i^v)dZ. \tag{21}$$

We are going to make use of Eqs. 20 and 21 in the derivation below.

To compute the variational approximation to IB, we need to lower bound $I(Z,D^v)$ as

$$\begin{aligned} I(Z,D^v) &= \int q_w(Z,D^v)\log\frac{q_w(Z,D^v)}{q_w(Z)p(D^v)} \\ &= \int q_w(Z,D^v)\log\frac{q_w(D^v|Z)}{p(D^v)}dZdD^v \\ &\geq \int q_w(Z,D^v)\log\frac{p_\theta(D^v|Z)}{p(D^v)}dZdD^v \quad \text{(by Eq. 17)} \\ &= \int q_w(Z,D^v)\log p_\theta(D^v|Z)dZdD^v + \mathcal{H}(D^v), \end{aligned}$$

Supplement for the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021).

where the entropy $\mathcal{H}(D^v)$ is just a constant.

Subsequently, we upper bound $I(Z, D^t)$ as follows,

$$\begin{aligned} I(Z, D^t) &= \int q_w(Z, D^t) \log \frac{q_w(Z, D^t)}{q_w(Z)p(D^t)} dZ dD^t \\ &= \int q_w(Z|D^t)p(D^t) \log \frac{q_w(Z|D^t)}{q_w(Z)} dZ dD^t, \\ &\leq \int q_w(Z|D^t)p(D^t) \log \frac{q_w(Z|D^t)}{p_\theta(Z)} dZ dD^t \quad \text{(by Eq. 18)} \end{aligned}$$

Then we obtain the overall loss, $\mathcal{F}(\theta, w) \leq \mathcal{L}_{\text{IB}}(w)$:

$$\begin{aligned} \mathcal{F}(\theta, w) = &\int q_w(Z, D^v) \log p_\theta(D^v|Z) dZ dD^v \\ &- \beta \int q_w(Z|D^t)p(D^t) \log \frac{q_w(Z|D^t)}{p_\theta(Z)} dZ dD^t, \end{aligned}$$

where we dropped the constant entropic term $\mathcal{H}(D^v)$. Therefore, given a set of task pairs $\{D_i^t, D_i^v\}_{i=1}^b$, where each $(D_i^t, D_i^v) \sim \widetilde{p}(D^v, D^t)$, the objective function for learning (θ, w) becomes the empirical average, $\frac{1}{b}\sum_{i=1}^b \mathcal{F}_i(\theta, w)$, where

$$\begin{aligned} \mathcal{F}_i(w, \theta) = &\int q_w(Z_i|D_i^t) \log p_\theta(D_i^v|Z_i) dZ_i \qquad (22) \\ &- \beta \int q_w(Z_i|D_i^t) \log \frac{q_w(Z_i|D_i^t)}{p_\theta(Z_i)} dZ_i, \end{aligned}$$

where for the first term we made use of Eqs. 20 and 21 with $f(D^v, Z) = \log p_\theta(D^v|Z)$.

A.3 THE SUPERVISED META LEARNING VIB CASE

For the supervised meta learning case the joint density can be written as

$$\begin{aligned} &q_w(D^v, D^t, Z) \\ &= q_w(Z|Y^t, X^t, X^v)p(Y^t, Y^v|X^t, X^v)p(X^v, X^t), \\ &= q_w(Z|Y^t, X)p(Y^t, Y^v|X)p(X), \qquad (23) \end{aligned}$$

where $X = (X^t, X^v)$ and the encoding distribution $q_w(Z|Y^t, X)$ could depend on all inputs X but only on the training outputs Y^t. The derivation of the VIB objective is similar as the general case, with the difference that now we approximate the conditional information bottleneck $I(Z, Y^v|X) - \beta I(Z, Y^t|X)$, where we condition on the inputs X. In other words, both $I(Z, Y^v|X)$ and $I(Z, Y^t|X)$ are conditional mutual informations, i.e., they have the form

$$\begin{aligned} I(z, y|x) &= \int q(x) \left[\int q(z, y|x) \log \frac{q(z, y|x)}{q(z|x)q(y|x)} dz dy \right] dx \\ &= \int q(z, y, x) \log \frac{q(z, y|x)}{q(z|x)q(y|x)} dz dy dx. \end{aligned}$$

We can lower bound $I(Z, Y^v|X)$ as follows,

$$\begin{aligned} &\int p(X) \left[\int q_w(Z, Y^v|X) \log \frac{q_w(Z, Y^v|X)}{q_w(Z|X)p(Y^v|X)} dZ dY^v \right] dX \\ &= \int p(X) \int q_w(Z, Y^v|X) \log \frac{q_w(Y^v|Z, X)}{p(Y^v|X)} dZ dY^v dX \\ &\geq \int p(X) \int q_w(Z, Y^v|X) \log \frac{p_\theta(Y^v|Z, X)}{p(Y^v|X)} dZ Y^v dX \\ &= \int q_w(Z, Y^v, X) \log \frac{p_\theta(Y^v|Z, X)}{p(Y^v|X)} dZ dY^v dX \\ &= \int q_w(Z, Y^v, X) \log p_\theta(Y^v|Z, X) dZ dY^v dX \\ &\quad - \int p(Y^v, X) \log p(Y^v|X) dY^v dX. \end{aligned}$$

In the second line above, $q_w(Z|X)$ cancels, and in the third line we have applied Eq. 17. Note that $-\int p(Y^v, X) \log p(Y^v|X) dY^v dX$ is just a constant that does not depend on tunable parameters. Also

$$q_w(Z, Y^v, X) = \int q_w(Z|Y^t, X)p(Y^t, Y^v|X)p(X) dY^t, \qquad (24)$$

so that if we have a task sample $(Y_i^t, Y_i^v, X_i) \sim p(Y^t, Y^v|X)p(X)$, an unbiased estimate of the expectation $\int q_w(Z, Y^v, X) \log p_\theta(Y^v|Z, X) dZ dY^v dX$ is given by

$$\int q_w(Z|Y_i^t, X_i) \log p_\theta(Y_i^v|Z, X_i) dZ. \qquad (25)$$

We upper bound $I(Z, Y^t|X)$ as follows,

$$\begin{aligned} &\int p(X) \left[\int q_w(Z, Y^t|X) \log \frac{q_w(Z, Y^t|X)}{q_w(Z|X)p(Y^t|X)} dZ dY^t \right] dX \\ &= \int p(X) \left[\int q_w(Z, Y^t|X) \log \frac{q_w(Z|Y^t, X)}{q_w(Z|X)} dZ dY^t \right] dX, \\ &\leq \int p(X) \int q_w(Z, Y^t|X) \log \frac{q_w(Z|Y^t, X)}{p_\theta(Z|X)} dZ dY^t dX, \\ &= \int q_w(Z|Y^t, X)p(Y^t, X) \log \frac{q_w(Z|Y^t, X)}{p_\theta(Z|X)} dZ dY^t dX, \end{aligned}$$

Then we obtain the overall objective,

$$\begin{aligned} &\mathcal{F}(\theta, w) = \int q_w(Z, Y^v, X) \log p_\theta(Y^v|Z, X) dZ dY^v dX \\ &- \beta \int q_w(Z|Y^t, X)p(Y^t, X) \log \frac{q_w(Z|Y^t, X)}{p_\theta(Z|X)} dZ dY^t dX, \end{aligned}$$

where $p(Y^t|X)$ cancels in the second line, we have used Eq. 18 in the third line, and we have dropped the constant term. Therefore, given a set of task pairs the objective becomes the empirical average, $\frac{1}{b}\sum_{i=1}^b \mathcal{F}_i(\theta, w)$, where

$$\begin{aligned} \mathcal{F}_i(\theta, w) = &\int q_w(Z|Y_i^t, X_i) \log p_\theta(Y_i^v|Z, X_i) dZ \qquad (26) \\ &- \beta \int q_w(Z|Y_i^t, X_i) \log \frac{q_w(Z|Y_i^t, X_i)}{p_\theta(Z|X_i)} dZ, \end{aligned}$$

where we made use of Eq. 25.

A.4 CONNECTION WITH VARIATIONAL INFERENCE

As mentioned in the main paper, the VIB for meta learning (where we consider for simplicity the general case from Appendix A.2) is similar to applying approximate variational inference to a certain joint model over the validation set,

$$p_\theta(D^v|Z)p_\theta(Z),$$

where $p_\theta(D^v|Z)$ is the decoder model, $p_\theta(Z)$ a prior model over the latent variables and where the corresponding marginal likelihood is

$$p(D^v) = \int p_\theta(D^v|Z)p_\theta(Z)dZ.$$

We can lower bound the log marginal likelihood with a variational distribution $q_w(Z|D^t)$ that depends on the training set D^t,

$$\mathcal{F}_{\beta=1}(w,\theta) = \int q_w(Z|D^t)\log p_\theta(D^v|Z)dZ \qquad (27)$$
$$- \int q_w(Z|D^t)\log\frac{q_w(Z|D^t)}{p_\theta(Z)}dZ,$$

which corresponds to the VIB objective with $\beta = 1$.

B TRANSDUCTIVE AND NON-TRANSDUCTIVE META LEARNING

Here, we discuss how the transductive and non-transductive settings that appear in few-shot image classification [Bronskill et al., 2020, Finn et al., 2017, Nichol et al., 2018], due to the use of batch-normalization, can be interpreted under our VIB framework by defining suitable encodings. We shall use MAML as an example, but the discussion is more generally relevant.

The transductive case occurs when the concatenated support and validation/test inputs $X = (X^t, X^v)$ of a single task (we ignore the task index i to keep the notation uncluttered) are used to compute batch-norm statistics (possibly at different stages) shared by all validation/test points, when predicting those points. For MAML this implies a deterministic parametric encoding, i.e., common to all individual validation inputs $x_j^v \in X^v$, obtained by a sequence of two steps: (i) Obtain first the task-specific parameter ψ in the usual way by the support loss, i.e., $\psi = \theta + \Delta(\theta, D^t)$. If batch-normalization is used here, then the statistics are computed only by X^t. (ii) Compute the validation loss by applying batch-normalization on X^v or the union $X = X^t \cup X^v$ (the union seems to be a better choice, but not used often in practice for computational reasons; e.g., Finn et al. [2017], Nichol et al. [2018] prefer to use only X^v). In both cases, the underlying encoder is parametric over the final effective task parameter $\widetilde{\psi} = BN(\psi, X)$, where BN denotes the final batch-norm operation that outputs a parameter vector, that predicts all validation points and it is a deterministic delta measure.

In contrast, the non-transductive setting occurs when each individual validation input x_j^v is concatenated with the support inputs X^t to form the sets $x_j^v \cup X^t$, $j = 1, \ldots, n^v$. Then, each set $x_j^v \cup X^t$ is used to compute point-specific batch-norm statistics when predicting the corresponding validation output y_j^v. Under the VIB framework this corresponds to a non-parametric encoding, which grows with the size of the validation set. The first deterministic step of this encoder is the same (i) above from the transductive case but the second step differs in the sense that now we get a validation point-specific task parameter $\widetilde{\psi}_j = BN(\psi, x_j^v \cup X^t)$ by computing the statistics using the set $x_j^v \cup X^t$. For MAML, this encoding becomes, $Z \equiv \{\widetilde{\psi}_j\}_{j=1}^{n^v}$, and the encoder distribution is a product of delta measures. i.e., $p(\{\widetilde{\psi}_j\}_{j=1}^{n^v}|Y^t, X) \equiv \prod_{j=1}^{n^v}\delta_{\widetilde{\psi}_j, BN(\theta+\Delta(\theta,D^t), x_j^v\cup X^t)}$.

Finally, note that under the VIB perspective it does not make much sense to meta train transductively and meta test non-transductively and vice versa, since this changes the encoding. That is, in meta testing we should do the same as in meta training.

C FURTHER DETAILS ABOUT THE GAUSSIAN PROCESS METHOD

For simplicity next we ignore the task index i to keep the notation uncluttered, and write for example $\mathbf{f}_i^t$ as $\mathbf{f}^t$.

C.1 DERIVATION OF THE VIB BOUND

The VIB objective for a single task from Eq. 26 in the main paper is computed as follows

$$\sum_{j=1}^{n^v}\mathbb{E}_{q(f_j^v)}[\log p(y_j^v|f_j^v)] - \beta\int p(\mathbf{f}^v|\mathbf{f}^t, X^v, X^t)q(\mathbf{f}^t|D^t)$$
$$\times\log\frac{p(\mathbf{f}^v|\mathbf{f}^t, X^v, X^t)q(\mathbf{f}^t|D^t)}{p(\mathbf{f}^v|\mathbf{f}^t, X^v, X^t)p(\mathbf{f}^t|X^t)}d\mathbf{f}^t d\mathbf{f}^v$$
$$=\sum_{j=1}^{n^v}\mathbb{E}_{q(f_j^v)}[\log p(y_j^v|f_j^v)] - \beta\int q(\mathbf{f}^t|D^t)\log\frac{q(\mathbf{f}^t|D^t)}{p(\mathbf{f}^t|X^t)}d\mathbf{f}^t$$
$$=\sum_{j=1}^{n^v}\mathbb{E}_{q(f_j^v)}[\log p(y_j^v|f_j^v)] - \beta\mathrm{KL}\left[q(\mathbf{f}^t|D^t)||p(\mathbf{f}^t|X^t)\right], \qquad (28)$$

where $q(f_j^v) = \int p(f_j^v|\mathbf{f}^t, x_j^v, X^t)q(\mathbf{f}^t|D^t)d\mathbf{f}^t$ is a marginal Gaussian over an individual validation function value f_j^v, as also explained in the main paper. Specifically, $q(f_j^v)$ depends on the training set (Y^t, X^t) and the single

validation input x_j^v, so intuitively from the training set and the corresponding function values $\mathbf{f}^t$ we extrapolate (through the conditional GP $p(f_j^v|\mathbf{f}^t, x_j^v, X^t)$) to the input x_j^v in order to predict its function value f_j^v.

Given the specific amortization of $q(\mathbf{f}^t|D^t)$:

$$q(\mathbf{f}^t|D^t) = \frac{\left(\prod_{j=1}^{n^t}\mathcal{N}(m_j^t|s_j^t)\right)\mathcal{N}(\mathbf{f}^t|0,\mathbf{K}^t)}{\mathcal{N}(\mathbf{m}^t|0,\mathbf{K}^t+\mathbf{S}^t)} \quad (29)$$
$$= \mathcal{N}(\mathbf{f}^t|\mathbf{K}^t(\mathbf{K}^t+\mathbf{S}^t)^{-1}\mathbf{m}^t, \mathbf{K}^t-\mathbf{K}^t(\mathbf{K}^t+\mathbf{S}^t)^{-1}\mathbf{K}^t),$$

the VIB objective, by using the middle part of Eq. 29, can be written in the following form,

$$\sum_{j=1}^{n^v}\mathbb{E}_{q(f_j^v)}[\log p(y_j^v|f_j^v)] - \beta\sum_{j=1}^{n^t}\mathbb{E}_{q(f_j^t)}[\log\mathcal{N}(m_j^t|f_j^t,s_j^t)]$$
$$+\beta\log\mathcal{N}(\mathbf{m}^t|\mathbf{0},\mathbf{K}^t+\mathbf{S}^t), \quad (30)$$

which is convenient from computational and programming point of view. Specifically, to compute this we need to perform a single Cholesky decomposition of $\mathbf{K}^t+\mathbf{S}^t$ which scales as $O((n^t)^3)$, i.e., cubically w.r.t. the size of the support set n^t. This is fine for small support sets (which is the standard case in few-shot learning) but it can become too expensive when n^t becomes very large. However, given that the kernel has the linear form $k_\theta(x,x') = \phi(x;\theta)^\top\phi(x';\theta)$ (ignoring any kernel variance σ_f^2 for notational simplicity), where $\phi(x_i;\theta)$ is M-dimensional and given that $M \ll n^t$, we can also carry out the computations based on the Cholesky decomposition of a matrix of size $M\times M$. This is because $\mathbf{K}^t = \Phi^t{\Phi^t}^\top$, where Φ^t is an $n^t\times M$ matrix storing as rows the features vectors on the support inputs X^t, and therefore we can apply the standard matrix inversion and determinant lemmas for the matrix $\Phi^t{\Phi^t}^\top+\mathbf{S}^t$ when computing $\log\mathcal{N}(\mathbf{m}^t|\mathbf{0},\mathbf{K}^t+\mathbf{S}^t)$. Such $O(M^3)$ computations also gives us the quantities $q(f_j^v)$ and $q(f_j^t)$, as explained next.

C.2 DATA EFFICIENT GP META TESTING PREDICTION WITH CONSTANT MEMORY

Once we have trained the GP meta learning system we can consider meta testing where a new fresh task is provided having a support set $D_*^t = (Y_*^t, X_*^t)$ based on which we predict at any arbitrary validation/test input x_*. This requires to compute quantities (such as the mean value $\mathbb{E}[y_*]$) associated with the predictive density

$$q(y_*) = \int p(y_*|f_*)p(f_*|\mathbf{f}_*^t,x_*,X_*^t)q(\mathbf{f}_*^t|D_*^t)df_*d\mathbf{f}_*^t$$
$$= \int p(y_*|f_*)q(f_*)df_*,$$

where $q(f_*)$ is an univariate Gaussian given by

$$q(f_*) = \mathcal{N}(f_*|\mathbf{k}_*^t(\mathbf{K}^t+\mathbf{S}^t)^{-1}\mathbf{m}^t, k_* - \mathbf{k}_*^t(\mathbf{K}^t+\mathbf{S}^t)^{-1}{\mathbf{k}_*^t}^\top),$$

$$\mathbf{k}_*^t = \phi_*^\top\Phi^t,\ \mathbf{K}^t = \Phi^t{\Phi^t}^\top,\ k_* = \phi_*^\top\phi_*,\ \phi_* = \phi(x_*;\theta).$$

Here, Φ^t is an $n_*^t\times M$ matrix storing as rows the features vectors on the support inputs X_*^t. Note that if we wish to evaluate $q(y_*)$ at certain value of y_*, and given that the likelihood $p(y_*|f_*)$ is not the standard Gaussian, we can use 1-D Gaussian quadrature or Monte Carlo by sampling from $q(f_*)$.

An interesting property of the above predictive density is that when the support set D_*^t can grow incrementally, e.g., individual data points or mini-batches are added sequentially, the predictive density can be implemented with constant memory without requiring to explicit memorize the points in the support. The reason is that the feature parameters θ remain constant at meta test time and the kernel function is linear, so we can apply standard tricks to update the sufficient statistics as in Bayesian linear regression.

More precisely, what we need to show is that we can sequentially update the mean and variance of $q(f_*)$ with constant memory. The distribution $q(f_*)$ can be written as

$$q(f_*) = \mathcal{N}\Big(f_*\Big|\phi_*^\top{\Phi^t}^\top(\Phi^t{\Phi^t}^\top+\mathbf{S}^t)^{-1}\mathbf{m}^t, \quad (31)$$
$$\phi_*^\top\left(I-{\Phi^t}^\top(\Phi^t{\Phi^t}^\top+\mathbf{S}^t)^{-1}\Phi^t\right)\phi_*\Big)$$
$$= \mathcal{N}\Big(f_*\Big|\phi_*^\top({\Phi^t}^\top[\mathbf{S}^t]^{-1}\Phi^t+I)^{-1}{\Phi^t}^\top[\mathbf{S}^t]^{-1}\mathbf{m}^t,$$
$$\phi_*^\top({\Phi^t}^\top[\mathbf{S}^t]^{-1}\Phi^t+I)^{-1}\phi_*\Big),$$

where we applied the matrix inversion lemma backwards to write $I-{\Phi^t}^\top(\Phi^t{\Phi^t}^\top+\mathbf{S}^t)^{-1}\Phi^t = ({\Phi^t}^\top[\mathbf{S}^t]^{-1}\Phi^t+I)^{-1}$ and also used that ${\Phi^t}^\top(\Phi^t{\Phi^t}^\top+\mathbf{S}^t)^{-1} = {\Phi^t}^\top(\Phi^t{\Phi^t}^\top[\mathbf{S}^t]^{-1}+I)^{-1}[\mathbf{S}^t]^{-1} = ({\Phi^t}^\top[\mathbf{S}^t]^{-1}\Phi^t+I)^{-1}{\Phi^t}^\top[\mathbf{S}^t]^{-1}$ (based on the identity $(AB+I)^{-1}A = A(BA+I)^{-1}$). Now observe that the M-dimensional vector $\mathbf{b}^t = {\Phi^t}^\top[\mathbf{S}^t]^{-1}\mathbf{m}^t = \sum_{j=1}^{n^t}\phi(x_j^t;\theta)\frac{m_j^t}{s_j^t}$ can grow incrementally without memorizing the feature vectors $\phi(x_j^t;\theta)$ based on the recursion $\mathbf{b}^t \leftarrow \mathbf{b}^t+\phi(x_j^t;\theta)\frac{m_j^t}{s_j^t}$ (with the initialization $\mathbf{b}^t=0$) as individual data points (similarly for mini-batches) are added in the support set: $D^t \leftarrow D^t\cup(x_j^t,y_j^t)$. Similarly, the $M\times M$ matrix $A^t = {\Phi^t}^\top[\mathbf{S}^t]^{-1}\Phi^t = \sum_{j=1}^{n^t}\frac{1}{s_j^t}\phi(x_j^t;\theta)\phi(x_j^t;\theta)^\top$ can also be computed recursively with constant $O(M^2)$ memory.

Finally, note that the above constant memory during meta testing can only be implemented when the feature vector θ is fixed.

C.3 MULTI-CLASS CLASSIFICATION

For multi-class classification meta learning problems we need to introduce as many latent functions as classes. For instance, when the number of classes for each task is N we

will need N latent functions $f_n(x)$ which all are independent draws from the same GP. The marginal GP prior on the training and validation function values for a certain task factorizes as

$$\prod_{n=1}^{N} p(\mathbf{f}_n^v|\mathbf{f}_n^t, X^v, X^t)p(\mathbf{f}_n^t|X^t).$$

We assume a factorized encoding distribution of the form

$$\prod_{n=1}^{N} p(\mathbf{f}_n^v|\mathbf{f}_n^t, X^v, X^t)q(\mathbf{f}_n^t|D^t),$$

where each

$$q(\mathbf{f}_n^t|D^t) \\ = \mathcal{N}(\mathbf{f}_n^t|\mathbf{K}^t(\mathbf{K}^t+\mathbf{S}^t)^{-1}\mathbf{m}_n^t, \mathbf{K}^t - \mathbf{K}^t(\mathbf{K}^t+\mathbf{S}^t)^{-1}\mathbf{K}^t).$$

Here, $\mathbf{m}_n^t = Y_n^t \circ \widetilde{\mathbf{m}}^t$, and Y_n^t is a vector obtaining the value 1 for each data point x_j^t that belongs to class n and -1 otherwise. Note that the encoding distributions share the covariance matrix and they only have different mean vectors. The representation of $\mathbf{m}_n^t$ makes the full encoding distribution permutation invariant to the values of the class labels. Since also we are using shared (i.e., independent of class labels) amortized functions $\widetilde{m}_w(x)$ and $s_w(x)$, the terms $(\mathbf{S}^t, \widetilde{\mathbf{m}}^t)$ are common to all N factors. This allows to compute the VIB objective very efficiently (in way that is fully scalable w.r.t. the number of classes N) by requiring only a single Cholesky decomposition of the matrix $\mathbf{K}^t+\mathbf{S}^t$. Specifically, by working similarly to Appendix C.1 we obtain the VIB objective per single task,

$$\sum_{j=1}^{n^v} \mathbb{E}_{q(\{f_{n,j}^v\}_{n=1}^N)}[\log p(y_j^v|\{f_{n,j}^v\}_{n=1}^N)] \\ - \beta \sum_{n=1}^{N}\sum_{j=1}^{n^t} \mathbb{E}_{q(f_{n,j}^t)}[\log \mathcal{N}(m_{n,j}^t|f_{n,j}^t, s_j^t)] \\ + \beta \sum_{n=1}^{N} \log \mathcal{N}(\mathbf{m}_n^t|\mathbf{0}, \mathbf{K}^t+\mathbf{S}^t),$$

where $q(\{f_{n,j}^v\}_{n=1}^N) = \prod_{n=1}^N q(f_{n,j}^v)$ and each univariate Gaussian $q(f_{n,j}^v)$ is given by the same expression as provided in Appendix C.2. The last two terms of the bound (i.e., the ones multiplied by the hyperparameter β) are clearly analytically computed, while the first term involves an expectation of a log softmax since the likelihood is

$$p(y_j^v = n|\{f_{n',j}^v\}_{n'=1}^N) = \frac{e^{f_{n,j}^v}}{\sum_{n'=1}^N e^{f_{n',j}^v}}.$$

To evaluate this expectation we apply first the reparametrization trick to move all tunable parameters of $q(\{f_{n,j}^v\}_{n=1}^N)$ inside the log-likelihood (so that we get a new expectation under a product of N univariate standard normals) and then we apply Monte Carlo by drawing 200 samples.

Finally, note that to compute the predictive density we need to evaluate,

$$q(y_*) = \mathbb{E}_{q(\{f_{n,*}\}_{n=1}^N)}\left[p(y_*|\{f_{n,*}\}_{n=1}^N)\right],$$

which again is done by applying Monte Carlo by drawing 200 samples from $q(\{f_{n,*}\}_{n=1}^N)$. To decide the classification label based on the maximum class predictive probability (in order to compute, e.g., accuracy scores), we take advantage of the fact that all N univariate predictive Gaussians $q(f_{n,*})$ have the same variance but different means, thus the predicted class can be equivalently obtained by taking the argmax of the means of these N distributions.

C.4 SPECIFIC GP IMPLEMENTATION AND AMORTIZATION FOR FEW-SHOT CLASSIFICATION

For all few-shot multi-class classification experiments in order to implement the GP-VIB method we need to specify the feature vector $\phi(x;\theta)$ and the amortized variational functions $\widetilde{m}_w(x)$ and $s_w(x)$. The feature vector is specified to have exactly the same neural architecture used in previous works for all datasets. Note that when computing the GP kernel function, the feature vector $\phi(x;\theta)$ is also augmented with the value 1 to automatically account for a bias term.

Regarding the two amortized variational functions needed to obtain the encoder, we consider a shared (with the GP functions) representation by adding two heads to the same feature vector $\phi(x;\theta)$: the first head corresponds to a linear output function $\widetilde{m}_w(x)$ and the second applies at the end the softplus activation $s_w(x) = \log(1+\exp(a(x)))$ (since $s_w(x)$ represents variance) where the pre-activation $a(x)$ is obtained by a linear function of the feature vector. For numerical stability we also apply a final clipping by bounding these functions so that $\widetilde{m}_w(x) \in [-20, 20]$ and $s_w(x) \in [0.001, 20]$. The bounds -20 and 20 are almost never realized during optimization, so they are not so crucial, in contrast the lower bound 0.001 on $s_w(x)$ is rather crucial regarding numerical stability since it ensures that the minimum eigenvalue of the matrix $\mathbf{K}^t+\mathbf{S}^t$ (i.e., the matrix we need to decompose using Cholesky) is bounded below by 0.001.

For the simplified encoder where $(\widetilde{m}_w(x_j^t), s_w(x_j^t)) := (\widetilde{m}, \sigma^2)$ we simply learn two independent scalar parameters $(\widetilde{m}, \sigma^2)$, where $\sigma^2 = \log(1+\exp(a))$ and a is the actual parameter optimised. For $(\widetilde{m}, \sigma^2)$ we use the same bounds mentioned above.

References

David Barber and Felix Agakov. The IM algorithm: A variational approach to information maximization. In *Advances in Neural Information Processing Systems*, 2003.

John Bronskill, Jonathan Gordon, James Requeima, Sebastian Nowozin, and Richard E. Turner. TaskNorm: Rethinking batch normalization for meta-learning. *arXiv preprint arXiv:2003.03284*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.