A Tighter Sparse Variational Bounds for GP Regression

As mentioned in section 3.1, another way to improve the variational distribution $q(\mathbf{u})p_\perp(\mathbf{f}_\perp)$ in SVGP is to make $\mathbf{u}$ and $\mathbf{f}_\perp$ dependent. The best possible approximation of this type is obtained by the setting $q(\mathbf{u})$ to the optimal exact posterior conditional $q^*(\mathbf{u}) = p(\mathbf{u}|\mathbf{f}_\perp, \mathbf{y})$. The corresponding collapsed bound for GP regression can be derived by analytically marginalising out $\mathbf{u}$ from the joint model in Eq. (6),

$$\begin{aligned} p(\mathbf{y}|\mathbf{f}_\perp) &= \int p(\mathbf{y}|\mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}}^{-1}\mathbf{u})p(\mathbf{u})\ d\mathbf{u} \\ &= \mathcal{N}(\mathbf{y}|\mathbf{f}_\perp, \mathbf{Q_{ff}} + \sigma^2\mathbf{I}), \end{aligned} \tag{12}$$

and then forcing the approximation $p_\perp(\mathbf{f}_\perp)$:

$$\mathbb{E}_{p_\perp(\mathbf{f}_\perp)} \log \mathcal{N}(\mathbf{y}|\mathbf{f}_\perp, \mathbf{Q_{ff}} + \sigma^2\mathbf{I}). \tag{13}$$

This bound has a closed-form as

$$\log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{Q_{ff}} + \sigma^2\mathbf{I}) - \frac{1}{2}\mathrm{tr}\left[(\mathbf{Q_{ff}} + \sigma^2\mathbf{I})^{-1}(\mathbf{K_{ff}} - \mathbf{Q_{ff}})\right], \tag{14}$$

Applying the matrix inversion lemma to $(\mathbf{Q_{ff}}+\sigma^2\mathbf{I})^{-1}$, we have an equivalent form that can be directly compared with Eq. (3):

$$\underbrace{\log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{Q_{ff}} + \sigma^2\mathbf{I}) - \frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}})}_{=\text{Eq. (3)}} + \frac{1}{2\sigma^4}\mathrm{tr}\left[\mathbf{K_{fu}}(\mathbf{K_{uu}} + \sigma^{-2}\mathbf{K_{uf}}\mathbf{K_{fu}})^{-1}\mathbf{K_{uf}}(\mathbf{K_{ff}} - \mathbf{Q_{ff}})\right], \tag{15}$$

where the first two terms recover Eq. (3), suggesting this is a tighter bound than the Titsias (2009) bound. This bound is not amenable to large-scale datasets because of $O(N^2)$ storage and $O(MN^2)$ computation time (dominated by the matrix multiplication $\mathbf{K_{uf}}\mathbf{K_{ff}}$) requirements. However, it is still of theoretical interest and can be applied to medium-sized regression datasets, just like the SGPR algorithm using the Titsias (2009) bound.

B Details of Orthogonal Decomposition

In section 3.2 we described the following orthogonal decomposition for $f \in \mathcal{H}$, where $\mathcal{H}$ is the RKHS induced by kernel k:

$$f = f_\parallel + f_\perp, \quad f_\parallel \in V \text{ and } f_\perp \perp V. \tag{16}$$

Here V is the subspace spanned by the inducing basis: $V = \{\sum_{j=1}^M \alpha_j k(\mathbf{z}_j, \cdot),\ \boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_M]^\top \in \mathbb{R}^M\}$. Since $f_\parallel \in V$, we let $f_\parallel = \sum_{j=1}^M \alpha'_j k(\mathbf{z}_j, \cdot)$. According to the properties of orthogonal projection, we have

$$\langle f, g\rangle_\mathcal{H} = \langle f_\parallel, g\rangle_\mathcal{H}, \quad \forall g \in V, \tag{17}$$

where $\langle\rangle_\mathcal{H}$ is the RKHS inner product that satisfies the reproducing property: $\langle f, k(\mathbf{x}, \cdot)\rangle_\mathcal{H} = f(\mathbf{x})$. Similarly let $g = \sum_{j=1}^M \beta_j k(\mathbf{z}_j, \cdot)$. Then $\langle f, g\rangle_\mathcal{H} = \sum_{j=1}^M \beta_j f(\mathbf{z}_j)$, $\langle f_\parallel, g\rangle_\mathcal{H} = \sum_{i=1}^M \sum_{j=1}^M \alpha'_i\beta_j k(\mathbf{z}_i, \mathbf{z}_j)$. Plugging into Eq. (17) and rearranging the terms, we have

$$\boldsymbol{\beta}^\top(f(\mathbf{Z}) - k(\mathbf{Z}, \mathbf{Z})\boldsymbol{\alpha}') = 0, \quad \forall \boldsymbol{\beta} \in \mathbb{R}^M, \tag{18}$$

where $f(\mathbf{Z}) = [f(\mathbf{z}_1), \dots, f(\mathbf{z}_M)]^\top$, $k(\mathbf{Z}, \mathbf{Z})$ is a matrix with the ij-th term as $k(\mathbf{z}_i, \mathbf{z}_j)$, and $\boldsymbol{\alpha}' = [\alpha'_1, \dots, \alpha'_M]^\top$. Therefore,

$$\boldsymbol{\alpha}' = k(\mathbf{Z}, \mathbf{Z})^{-1} f(\mathbf{Z}), \tag{19}$$

and it follows that $f_\parallel(\mathbf{x}) = k(\mathbf{x}, \mathbf{Z})k(\mathbf{Z}, \mathbf{Z})^{-1} f(\mathbf{Z})$, where $k(\mathbf{x}, \mathbf{Z}) = [k(\mathbf{z}_1, \mathbf{x}), \dots, k(\mathbf{z}_M, \mathbf{x})]$. The above analysis arrives at the decomposition:

$$f_\parallel = k(\cdot, \mathbf{Z})k(\mathbf{Z}, \mathbf{Z})^{-1} f(\mathbf{Z}), \quad f_\perp = f - f_\parallel. \tag{20}$$

Although the derivation from Eq. (16) to (19) relies on the fact $f \in \mathcal{H}$, such that the inner product is well-defined, the decomposition in Eq. (20) is valid for any function f on $\mathcal{X}$. This motivates us to study it for $f \sim \mathcal{GP}(0, k)$. Substituting $\mathbf{u}$ for $f(\mathbf{Z})$ and $\mathbf{K_{uu}}$ for $k(\mathbf{Z}, \mathbf{Z})$, we have for $f \sim \mathcal{GP}(0, k)$:

$$f_\parallel = k(\cdot, \mathbf{Z})\mathbf{K_{uu}}^{-1}\mathbf{u} \sim p_\parallel \equiv \mathcal{GP}(0, k(\mathbf{x}, \mathbf{Z})\mathbf{K_{uu}}^{-1}k(\mathbf{Z}, \mathbf{x}')), \tag{21}$$

$$f_\perp \sim p_\perp \equiv \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{Z})\mathbf{K_{uu}}^{-1}k(\mathbf{Z}, \mathbf{x}')). \tag{22}$$

C The Collapsed SOLVE-GP Lower Bound

We derive the collapsed SOLVE-GP lower bound in Eq. (9) by seeking the optimal $q(\mathbf{u})$ that is independent of $\mathbf{f}_\perp$. First we rearrange the terms in the uncollapsed SOLVE-GP bound (Eq. (8)) as

$$\mathbb{E}_{q(\mathbf{u})}\left\{\mathbb{E}_{q_\perp(\mathbf{f}_\perp)}\left[\log\mathcal{N}\left(\mathbf{y}|\mathbf{f}_\perp+\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u},\sigma^2\mathbf{I}\right)\right]\right\}-\mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right]-\mathrm{KL}\left[q(\mathbf{v}_\perp)\|p_\perp(\mathbf{v}_\perp)\right]. \tag{23}$$

where $q_\perp(\mathbf{f}_\perp)=\mathcal{N}(\mathbf{m}_{\mathbf{f}_\perp},\mathbf{S}_{\mathbf{f}_\perp})$, and $\mathbf{m}_{\mathbf{f}_\perp}=\mathbf{C}_{\mathbf{fv}}\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{m}_{\mathbf{v}}$, $\mathbf{S}_{\mathbf{f}_\perp}=\mathbf{C}_{\mathbf{ff}}+\mathbf{C}_{\mathbf{fv}}\mathbf{C}_{\mathbf{vv}}^{-1}(\mathbf{S}_{\mathbf{v}}-\mathbf{C}_{\mathbf{vv}})\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{C}_{\mathbf{vf}}$. In the first term we can simplify the expectation over $\mathbf{f}_\perp$ as:

$$\begin{aligned}
&\mathbb{E}_{q_\perp(\mathbf{f}_\perp)}\log\mathcal{N}\left(\mathbf{y}|\mathbf{f}_\perp+\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u},\sigma^2\mathbf{I}\right)\\
&=\mathbb{E}_{q_\perp(\mathbf{f}_\perp)}\left[-\frac{N}{2}\log 2\pi-\frac{N}{2}\log\sigma^2-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{f}_\perp-\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u})^\top(\mathbf{y}-\mathbf{f}_\perp-\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u})\right]\\
&=\left[-\frac{N}{2}\log 2\pi-\frac{N}{2}\log\sigma^2-\frac{1}{2\sigma^2}(\mathbf{y}-\mathbf{m}_{\mathbf{f}_\perp}-\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u})^\top(\mathbf{y}-\mathbf{m}_{\mathbf{f}_\perp}-\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u})\right]\\
&\quad-\mathbb{E}_{q_\perp(\mathbf{f}_\perp)}\left[\frac{1}{2\sigma^2}(\mathbf{f}_\perp-\mathbf{m}_{\mathbf{f}_\perp})^\top(\mathbf{f}_\perp-\mathbf{m}_{\mathbf{f}_\perp})\right]\\
&=\log\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}+\mathbf{m}_{\mathbf{f}_\perp},\sigma^2\mathbf{I})-\frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{S}_{\mathbf{f}_\perp}).
\end{aligned}\tag{24}$$

Plugging into Eq. (23) and rearranging the terms, we have

$$\underbrace{\mathbb{E}_{q(\mathbf{u})}\left[\log\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}+\mathbf{m}_{\mathbf{f}_\perp},\sigma^2\mathbf{I})\right]-\mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right]}_{\leq\log\int\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}+\mathbf{m}_{\mathbf{f}_\perp},\sigma^2\mathbf{I})p(\mathbf{u})\,d\mathbf{u}}-\frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{S}_{\mathbf{f}_\perp})-\mathrm{KL}\left[q(\mathbf{v}_\perp)\|p_\perp(\mathbf{v}_\perp)\right]. \tag{25}$$

Clearly the leading two terms form a variational lower bound of the joint distribution $\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}+\mathbf{m}_{\mathbf{f}_\perp},\sigma^2\mathbf{I})p(\mathbf{u})$. The optimal $q(\mathbf{u})$ will turn it into the log marginal likelihood:

$$\log\int\mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}+\mathbf{m}_{\mathbf{f}_\perp},\sigma^2\mathbf{I})p(\mathbf{u})\,d\mathbf{u}=\log\mathcal{N}(\mathbf{y}|\mathbf{m}_{\mathbf{f}_\perp},\mathbf{Q}_{\mathbf{ff}}+\sigma^2\mathbf{I}). \tag{26}$$

Plugging this back, we have the collapsed SOLVE-GP bound in Eq. (9):

$$\log\mathcal{N}(\mathbf{y}|\mathbf{C}_{\mathbf{fv}}\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{m}_{\mathbf{v}},\mathbf{Q}_{\mathbf{ff}}+\sigma^2\mathbf{I})-\frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{S}_{\mathbf{f}_\perp})-\mathrm{KL}\left[\mathcal{N}(\mathbf{m}_{\mathbf{v}},\mathbf{S}_{\mathbf{v}})\|\mathcal{N}(\mathbf{0},\mathbf{C}_{\mathbf{vv}})\right], \tag{27}$$

Moreover, we could find the optimal $q^*(\mathbf{v})=\mathcal{N}(\mathbf{m}_{\mathbf{v}}^*,\mathbf{S}_{\mathbf{v}}^*)$ by setting the derivatives w.r.t. $\mathbf{m}_{\mathbf{v}}$ and $\mathbf{S}_{\mathbf{v}}$ to be zeros:

$$\mathbf{m}_{\mathbf{v}}^*=\mathbf{C}_{\mathbf{vv}}[\mathbf{C}_{\mathbf{vv}}+\mathbf{C}_{\mathbf{vf}}\mathbf{A}^{-1}\mathbf{C}_{\mathbf{fv}}]^{-1}\mathbf{C}_{\mathbf{vf}}\mathbf{A}^{-1}\mathbf{y}, \tag{28}$$

$$\mathbf{S}_{\mathbf{v}}^*=\mathbf{C}_{\mathbf{vv}}[\mathbf{C}_{\mathbf{vv}}+\sigma^{-2}\mathbf{C}_{\mathbf{vf}}\mathbf{C}_{\mathbf{fv}}]^{-1}\mathbf{C}_{\mathbf{vv}}, \tag{29}$$

where $\mathbf{A}=\mathbf{Q}_{\mathbf{ff}}+\sigma^2\mathbf{I}$. Then the collapsed bound with the optimal $q(\mathbf{v}_\perp)$ is

$$\log N(\mathbf{y}|\mathbf{C}_{\mathbf{fv}}\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{m}_{\mathbf{v}}^*,\mathbf{A})-\frac{1}{2\sigma^2}\mathrm{tr}[\mathbf{C}_{\mathbf{ff}}-\mathbf{B}(\mathbf{B}+\sigma^2\mathbf{I})^{-1}\mathbf{B}]-\mathrm{KL}\left[\mathcal{N}(\mathbf{m}_{\mathbf{v}}^*,\mathbf{S}_{\mathbf{v}}^*)\|\mathcal{N}(\mathbf{0},\mathbf{C}_{\mathbf{vv}})\right], \tag{30}$$

where $\mathbf{B}=\mathbf{C}_{\mathbf{fv}}\mathbf{C}_{\mathbf{vv}}^{-1}\mathbf{C}_{\mathbf{vf}}$.

D Computational Details

D.1 Training

To compute the lower bound in Eq. (8), we write it as

$$\sum_{n=1}^{N}\mathbb{E}_{q(f(\mathbf{x}_n);\Theta)}[\log p(y_n|f(\mathbf{x}_n))]-\mathrm{KL}\left[q(\mathbf{u})\|p(\mathbf{u})\right]\ -\mathrm{KL}\left[q(\mathbf{v}_\perp)\|p_\perp(\mathbf{v}_\perp)\right], \tag{31}$$

Algorithm 1 The SOLVE-GP lower bound via Cholesky decomposition. We parameterize the variational covariance matrices with their Cholesky factors $\mathbf{S_u} = \mathbf{L_u}\mathbf{L_u}^\top, \mathbf{S_v} = \mathbf{L_v}\mathbf{L_v}^\top$. $\mathbf{A} = \mathbf{L_u^0} \setminus \mathbf{K_{uv}}$ denotes the solution of $\mathbf{L_u^0 A} = \mathbf{K_{uv}}$. $\odot$ denotes elementwise multiplication. The differences from SVGP are shown in blue.

Input: $\mathbf{X}$ (training inputs), $\mathbf{y}$ (targets), $\mathbf{Z}, \mathbf{O}$ (inducing points), $\mathbf{m_u}, \mathbf{L_u}, \mathbf{m_v}, \mathbf{L_v}$ (variational parameters)

$\mathbf{K_{uu}} = k(\mathbf{Z}, \mathbf{Z})$, $\mathbf{K_{vv}} = k(\mathbf{O}, \mathbf{O})$

$\mathbf{L_u^0} = \underline{\text{Cholesky}(\mathbf{K_{uu}})}$, $\mathbf{K_{uv}} = k(\mathbf{Z}, \mathbf{O})$, $\mathbf{A} := \underline{\mathbf{L_u^0} \setminus \mathbf{K_{uv}}}$, $\mathbf{C_{vv}} = \mathbf{K_{vv}} - \underline{\mathbf{A}^\top \mathbf{A}}$, $\mathbf{L_v^0} = \underline{\text{Cholesky}(\mathbf{C_{vv}})}$

$\mathbf{K_{uf}} = k(\mathbf{Z}, \mathbf{X})$, $\mathbf{K_{vf}} = k(\mathbf{O}, \mathbf{X})$

$\mathbf{B} := \underline{\mathbf{L_u^0} \setminus \mathbf{K_{uf}}}$, $\mathbf{C_{vf}} = \mathbf{K_{vf}} - \mathbf{A}^\top \mathbf{B}$, $\mathbf{D} := \underline{\mathbf{L_v^0} \setminus \mathbf{C_{vf}}}$

$\mathbf{E} := \underline{\mathbf{L_u^0} \setminus \mathbf{B}}$, $\mathbf{F} := \underline{\mathbf{L_u}^\top \mathbf{E}}$, $\mathbf{G} := \underline{\mathbf{L_v^0} \setminus \mathbf{D}}$, $\mathbf{H} := \underline{\mathbf{L_v}^\top \mathbf{G}}$

$\boldsymbol{\mu}(\mathbf{X}) = \mathbf{E}^\top \mathbf{m_u} + \mathbf{G}^\top \mathbf{m_v}$

$\boldsymbol{\sigma}^2(\mathbf{X}) = \text{diag}(\mathbf{K_{ff}}) + (\mathbf{F} \odot \mathbf{F})^\top \mathbf{1} - (\mathbf{B} \odot \mathbf{B})^\top \mathbf{1} + (\mathbf{H} \odot \mathbf{H})^\top \mathbf{1} - (\mathbf{D} \odot \mathbf{D})^\top \mathbf{1}$

Compute LLD $= \sum_{n=1}^N \mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}(\mathbf{x}_n), \boldsymbol{\sigma}^2(\mathbf{x}_n))} \log p(y_n | f(\mathbf{x}_n))$ in closed form or using quadrature/Monte Carlo.

function COMPUTE_KL($\mathbf{m}$, $\mathbf{L}$, $\mathbf{L}^0$)

  $\mathbf{P} = \underline{\mathbf{L}^0 \setminus \mathbf{L}}$, $\mathbf{a} = \mathbf{L}^0 \setminus \mathbf{m}$

  return $\log(\text{diag}(\mathbf{L}^0))^\top \mathbf{1} - \log(\text{diag}(\mathbf{L}))^\top \mathbf{1} + 1/2((\mathbf{P} \odot \mathbf{P})^\top \mathbf{1} + \mathbf{a}^\top \mathbf{a} - M)$

end function

$\text{KL}_\mathbf{u} = \text{COMPUTE\_KL}(\mathbf{m_u}, \mathbf{L_u}, \mathbf{L_u^0})$, $\text{KL}_\mathbf{v} = \text{COMPUTE\_KL}(\mathbf{m_v}, \mathbf{L_v}, \mathbf{L_v^0})$

return $\text{LLD} - \text{KL}_\mathbf{u} - \text{KL}_\mathbf{v}$

where $\Theta := \{\mathbf{m_u}, \mathbf{S_u}, \mathbf{m_v}, \mathbf{S_v}, \mathbf{Z}, \mathbf{O}\}$ and $q(f(\mathbf{x}_n); \Theta)$ defines the marginal distribution of $\mathbf{f} = \mathbf{f}_\perp + \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}$ for the n-th data point given $\mathbf{u} \sim q(\mathbf{u})$ and $\mathbf{f}_\perp \sim q_\perp(\mathbf{f}_\perp)$. We can write $q(f(\mathbf{x}_n); \Theta)$ as

$$q(f(\mathbf{x}_n); \Theta) = \mathcal{N}(\mu(\mathbf{x}_n), \sigma^2(\mathbf{x}_n)), \tag{32}$$

where

$$\mu(\mathbf{x}_n) = k(\mathbf{x}_n, \mathbf{Z})\mathbf{K_{uu}^{-1}}\mathbf{m_u} + c(\mathbf{x}_n, \mathbf{O})\mathbf{C_{vv}^{-1}}\mathbf{m_v}, \tag{33}$$

$$\sigma^2(\mathbf{x}_n) = k(\mathbf{x}_n, \mathbf{Z})\mathbf{K_{uu}^{-1}}\mathbf{S_u}\mathbf{K_{uu}^{-1}}k(\mathbf{Z}, \mathbf{x}_n) + c(\mathbf{x}_n, \mathbf{x}_n) + c(\mathbf{x}_n, \mathbf{O})\mathbf{C_{vv}^{-1}}(\mathbf{S_v} - \mathbf{C_{vv}})\mathbf{C_{vv}^{-1}}c(\mathbf{O}, \mathbf{x}_n). \tag{34}$$

Here $c(\mathbf{x}, \mathbf{x}') := k(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{Z})\mathbf{K_{uu}^{-1}}k(\mathbf{Z}, \mathbf{x})$ denotes the covariance function of $p_\perp$. The univariate expectation of $\log p(y_n | f(\mathbf{x}_n))$ under $q(f(\mathbf{x}_n); \Theta)$ can be computed in closed form (e.g., for Gaussian likelihoods) or using quadrature (Hensman et al., 2015b). It can also be estimated by Monte Carlo with the reparameterization trick (Kingma and Welling, 2013; Titsias and Lázaro-Gredilla, 2014; Rezende et al., 2014) to propagate gradients. For large datasets, an unbiased estimate of the sum can be used for mini-batch training: $\frac{N}{|B|} \sum_{(\mathbf{x},y) \in B} \mathbb{E}_{q(f(\mathbf{x}); \Theta)}[\log p(y | f(\mathbf{x}))]$, where B denotes a small batch of data points.

Besides the log-likelihood term, we need to compute the two KL divergence terms:

$$\text{KL}\left[q(\mathbf{u}) \| p(\mathbf{u})\right] = \frac{1}{2}\left[\log\det \mathbf{K_{uu}} - \log\det \mathbf{S_u} - M + \text{tr}(\mathbf{K_{uu}^{-1}}\mathbf{S_u}) + \mathbf{m_u}^\top \mathbf{K_{uu}^{-1}}\mathbf{m_u}\right], \tag{35}$$

$$\text{KL}\left[q(\mathbf{v}_\perp) \| p_\perp(\mathbf{v}_\perp)\right] = \frac{1}{2}\left[\log\det \mathbf{C_{vv}} - \log\det \mathbf{S_v} - M + \text{tr}(\mathbf{C_{vv}^{-1}}\mathbf{S_v}) + \mathbf{m_v}^\top \mathbf{C_{vv}^{-1}}\mathbf{m_v}\right]. \tag{36}$$

We note that if the blue parts in Eqs. (31) to (34) are removed, then we recover the SVGP lower bound in Eq. (2). An implementation of the above computations using the Cholesky decomposition is shown in algorithm 1.

D.2 Prediction

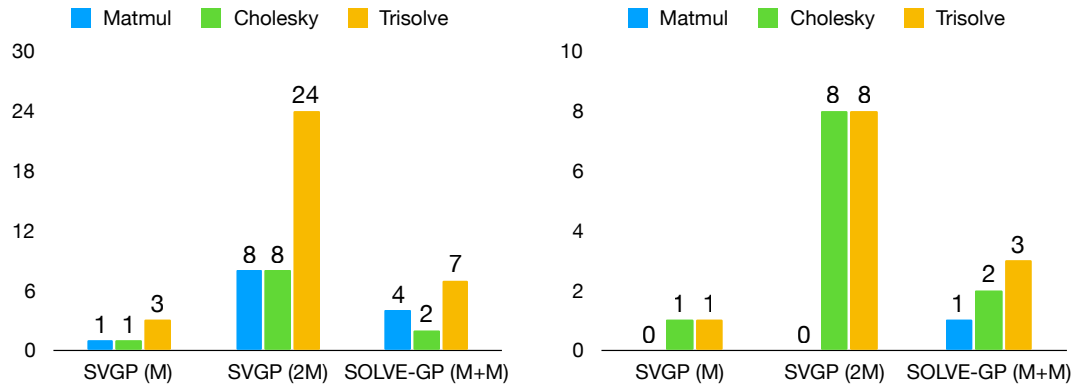
We can predict the function value at a test point $\mathbf{x}^*$ with the approximate posterior by substituting $\mathbf{x}^*$ for $\mathbf{x}_n$ in Eq. (32). For multiple test points $\mathbf{X}^*$, we denote the joint predictive density by $\mathcal{N}(\mathbf{f}^* | \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$, where the predicted mean and covariance are

$$\boldsymbol{\mu}^* = \mathbf{K_{*u}}\mathbf{K_{uu}^{-1}}\mathbf{m_u} + \mathbf{C_{*v}}\mathbf{C_{vv}^{-1}}\mathbf{m_v}, \tag{37}$$

$$\boldsymbol{\Sigma}^* = \mathbf{K_{*u}}\mathbf{K_{uu}^{-1}}\mathbf{S_u}\mathbf{K_{uu}^{-1}}\mathbf{K_{u*}} + \mathbf{C_{**}} - \mathbf{C_{*v}}\mathbf{C_{vv}^{-1}}(\mathbf{C_{vv}} - \mathbf{S_v})\mathbf{C_{vv}^{-1}}\mathbf{C_{v*}}. \tag{38}$$

Table 4: Cubic-cost operations in SOLVE-GP and SVGP, following the implementation in algorithm 1.

	SVGP	SOLVE-GP
Matrix multiplication	$\mathcal{O}(NM^2) \times 1$	$\mathcal{O}(NM^2) \times 1$ $\mathcal{O}(NM_2^2) \times 1$ $\mathcal{O}(NMM_2) \times 1$ $\mathcal{O}(MM_2^2) \times 1$
Cholesky	$\mathcal{O}(M^3) \times 1$	$\mathcal{O}(M^3) \times 1$ $\mathcal{O}(M_2^3) \times 1$
Solving triangular matrix equations	$\mathcal{O}(M^3) \times 1$ $\mathcal{O}(NM^2) \times 2$	$\mathcal{O}(M^3) \times 1$ $\mathcal{O}(NM^2) \times 2$ $\mathcal{O}(M_2^3) \times 1$ $\mathcal{O}(NM_2^2) \times 2$ $\mathcal{O}(M_2M^2) \times 1$



(a) $N \approx M$ (b) $M \gg N$

Figure 4: Comparison of computational cost for SVGP and SOLVE-GP. For each method and each type of cubic-cost operation, we plot the factor of increase in cost compared to a single operation on $M \times M$ matrices.

D.3 Computational Complexity

As mentioned in section 3.3, the time complexity of SOLVE-GP is $\mathcal{O}(N\bar{M}^2 + \bar{M}^3)$ per gradient update, where $\bar{M} = \max(M, M_2)$ and N is the batch size. Here we provide a more fine-grained analysis by counting cubic-cost operations and compare to the standard SVGP method. We underlined all the cubic-cost operations in algorithm 1, including matrix multiplication, Cholesky decomposition, and solving triangular matrix equations. We count them for SVGP and SOLVE-GP. The results are summarized in Table 4.

For comparison purposes, we study two cases of mini-batch training: (i) $N \approx M$ and (ii) $M \gg N$. We consider SOLVE-GP with $M_2 = M$, which has $2M$ inducing points in total, and then compare to SVGP with M and $2M$ inducing points. For each method and each type of operation, we plot the factor of increase in cost compared to a single operation on $M \times M$ matrices. For instance, when $N \approx M$ (Fig. 4a), SVGP with M inducing points requires solving three triangular matrix equations for $M \times M$ matrices. Doubling the number of inducing points in SVGP increases the cost by a factor of 8, plotted as 24 for SVGP ($2M$). In contrast, in SOLVE-GP with M orthogonal inducing points we only need to solve 7 triangular matrix equations for $M \times M$ matrices. The comparison under the case of $M \gg N$ is shown in Fig. 4b. In this case SOLVE-GP additionally introduces one $\mathcal{O}(M^3)$ matrix multiplication operation, but overall the algorithm is still much faster than SVGP ($2M$) given the speed-up in Cholesky decomposition and solving matrix equations.

E Details of Eq. (11)

The variational distribution in Eq. (11) is defined as:

$$q(\mathbf{u}^L, \mathbf{f}_\perp^L) = \int \prod_{\ell=1}^{L} \left[p_\perp(\mathbf{f}_\perp^\ell | \mathbf{v}_\perp^\ell, \mathbf{f}_\perp^{\ell-1}, \mathbf{u}^{\ell-1}) q(\mathbf{v}_\perp^\ell) q(\mathbf{u}^\ell) d\mathbf{u}^\ell d\mathbf{v}_\perp^\ell \right] \prod_{\ell=1}^{L-1} d\mathbf{f}_\perp^\ell. \tag{39}$$

F Whitening

Similar to the practice in SVGP methods, we can apply the "whitening" trick (Murray and Adams, 2010; Hensman et al., 2015b) to SOLVE-GP. The goal is to improve the optimization of variational approximations by reducing correlation in the posterior distributions. Specifically, we could "whiten" $\mathbf{u}$ by using $\mathbf{u}' = \mathbf{K}_{\mathbf{uu}}^{-1/2}\mathbf{u}$, where $\mathbf{K}_{\mathbf{uu}}^{-1/2}$ denotes the Cholesky factor of the prior covariance $\mathbf{K}_{\mathbf{uu}}$. Then posterior inference for $\mathbf{u}$ turns into inference for $\mathbf{u}'$, which has an isotropic Gaussian prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Then we parameterize the variational distribution w.r.t. $\mathbf{u}'$: $q(\mathbf{u}') = \mathcal{N}(\mathbf{m}_\mathbf{u}, \mathbf{S}_\mathbf{u})$. Whitening $q(\mathbf{v}_\perp)$ is similar to whitening $q(\mathbf{u})$, i.e., we parameterize the variational distribution w.r.t. $\mathbf{v}'_\perp = \mathbf{C}_{\mathbf{vv}}^{-1/2}\mathbf{v}_\perp$ and set $q(\mathbf{v}'_\perp) = \mathcal{N}(\mathbf{m}_\mathbf{v}, \mathbf{S}_\mathbf{v})$. The algorithm can be derived by replacing $\mathbf{m}_\mathbf{u}, \mathbf{m}_\mathbf{v}$ with $\mathbf{L}_\mathbf{u}^0\mathbf{m}_\mathbf{u}$, $\mathbf{L}_\mathbf{v}^0\mathbf{m}_\mathbf{v}$, and $\mathbf{S}_\mathbf{u}, \mathbf{S}_\mathbf{v}$ with $\mathbf{L}_\mathbf{u}^0\mathbf{S}_\mathbf{u}{\mathbf{L}_\mathbf{u}^0}^\top, \mathbf{L}_\mathbf{v}^0\mathbf{S}_\mathbf{v}{\mathbf{L}_\mathbf{v}^0}^\top$ in algorithm 1 and removing the canceled terms.

G Experiment Details

For all experiments, we use kernels with a shared lengthscale across dimensions. All model hyperparameters, including kernel parameters, patch weights in convolutional GP models, and observation variances in regression experiments, are optimized jointly with variational parameters using ADAM. The variational distributions $q(\mathbf{u})$ and $q(\mathbf{v}_\perp)$ are by default initialized to the prior distributions. Unless stated otherwise, no "whitening" trick (Murray and Adams, 2010; Hensman et al., 2015b) is used for SVGP or SOLVE-GP.

G.1 1D Regression

We randomly sample 100 training data points from Snelson's dataset (Snelson and Ghahramani, 2006) as the training data. All models use Gaussian RBF kernels and are trained for 10K iterations with learning rate 0.01 and mini-batch size 20. The GP kernel is initialized with lengthscale 1 and variance 1. The Gaussian likelihood is initialized with variance 0.1.

G.2 Convolutional GP Models

All models are trained for 300K iteration with learning rate 0.003 and batch size 64. The learning rate is annealed by 0.25 every 50K iterations to ensure convergence. We use a zero mean function and the robust multi-class classification likelihood (Hernández-Lobato et al., 2011). The Gaussian RBF kernels for the patch response GPs in all layers are initialized with lengthscale 5 and variance 5. We used the TICK kernel (Dutordoir et al., 2019) for the output layer GP, for which we use a Matérn32 kernel between patch locations with lengthscale initialized to 3. We initialize the inducing patch locations to random values in $[0, H] \times [0, W]$, where $[H, W]$ is the shape of the output feature map in patch extraction.

Convolutional GPs We set patch size to 5×5 and stride to 1. We use the whitening trick in all single-layer experiments for $\mathbf{u}$ (and $\mathbf{v}_\perp$) since we find it consistently improves the performance. Inducing points are initialized by cluster centers which are generated from running K-means on $M \times 100$ (for SVGP) or $(M + M_2) \times 100$ (for SOLVE-GP) image patches. The image patches are randomly sampled from 1K images randomly selected from the dataset.

Deep Convolutional GPs The detailed model configurations are summarized in Table 5. No whitening trick is used for multi-layer experiments because we find it hurts performance. Inducing points in the input layer are initialized in the same way as in the single-layer model. In Blomqvist et al. (2018); Dutordoir et al. (2019), three-layer models were initialized with the trained values of a two-layer model to avoid getting stuck in bad

local minima. Here we design an initialization scheme that allows training deeper models without the need of pretraining. We initialize the inducing points in deeper layers by running K-means on $M \times 100$ (for SVGP) or $(M + M_2) \times 100$ (for SOLVE-GP) image patches which are randomly sampled from the projections of 1K images to these layers. The projections are done by using a convolution operation with random filters generated using Glorot uniform (Glorot and Bengio, 2010). We also note that when implementing the forward sampling for approximating the log-likelihood term, we follow the previous practice (Dutordoir et al., 2019) to ignore the correlations between outputs of different patches to get faster sampling, which works well in practice. While it is also possible to take into account the correlation when sampling as this only increases the computation cost by a constant factor, doing this might require multi-GPU training due to the additional memory requirements.

Table 5: Model configurations of deep convolutional GPs.

	2-layer	3-layer
Layer 0	patch size 5×5, stride 1, out channel 10,	patch size 5×5, stride 1, out channel 10
Layer 1	patch size 4×4, stride 2	patch size 4×4, stride 2, out channel 10
Layer 2	-	patch size 5×5, stride 1

G.3 Regression Benchmarks

The experiment settings are followed from Wang et al. (2019), where we used GPs with Matérn32 kernels and 80% / 20% training / test splits. A 20% subset of the training set is used for validation. We repeat each experiment 5 times with random splits and report the mean and standard error of the performance metrics. For all datasets we train for 100 epochs with learning rate 0.01 and mini-batch size 1024.

H Additional Results

H.1 Regression Benchmarks

Due to space limitations in the main text, we include the Root Mean Squared Error (RMSE) on test data in Table 6. The results on Elevators and Bike are shown in Table 7.

Table 6: Test RMSE values of regression datasets. The numbers in parentheses are standard errors. Best mean values are highlighted, and asterisks indicate statistical significance.

		Kin40k	Protein	KeggDirected	KEGGU	3dRoad	Song	Buzz	HouseElectric
	N	25,600	29,267	31,248	40,708	278,319	329,820	373,280	1,311,539
	d	8	9	20	27	3	90	77	9
SVGP	1024	0.193(0.001)	0.630(0.004)	0.098(0.003)	**0.123**(0.001)	0.482(0.001)	0.797(0.001)	0.263(0.001)	0.063(0.000)
	1536	0.182(0.001)	0.621(0.004)	0.098(0.002)	**0.123**(0.001)	0.470(0.001)	0.797(0.001)	0.263(0.001)	0.063(0.000)
ODVGP	$1024 + 1024$	0.183(0.001)	0.625(0.004)	0.176(0.012)	0.156(0.004)	0.467(0.001)	0.797(0.001)	0.263(0.001)	0.062(0.000)
	$1024 + 8096$	0.180(0.001)	0.618(0.004)	0.157(0.009)	0.157(0.004)	**0.462**(0.002)	0.797(0.001)	0.263(0.001)	0.062(0.000)
SOLVE-GP	$1024 + 1024$	***0.172**(0.001)	0.618(0.004)	**0.095**(0.002)	**0.123**(0.001)	0.464(0.001)	**0.796**(0.001)	**0.261**(0.001)	***0.061**(0.000)
SVGP	2048	0.177(0.001)	**0.615**(0.004)	0.100(0.003)	0.124(0.001)	0.467(0.001)	**0.796**(0.001)	0.263(0.000)	0.063(0.000)

H.2 Convolutional GP Models

We include here the full tables for CIFAR-10 classification, where we also report the accuracies and predictive log-likelihoods on the training data. Table 8 contains the results by convolutional GPs. Table 9 and Table 10 include the results of 2/3-layer deep convolutional GPs.

Table 7: Regression results on Elevators and Bike. Best mean values are highlighted.

		Elevators $N = 10,623,\ d = 18$		Bike $N = 11,122,\ d = 17$	
		Test LL	RMSE	Test LL	RMSE
SVGP	1024	-0.516(0.006)	0.398(0.004)	-0.218(0.006)	0.283(0.003)
	1536	-0.511(0.007)	0.396(0.004)	-0.203(0.006)	0.279(0.003)
ODVGP	$1024 + 1024$	-0.518(0.006)	0.397(0.004)	-0.191(0.006)	0.272(0.003)
	$1024 + 8096$	-0.523(0.006)	0.399(0.004)	**-0.186**(0.006)	**0.270**(0.003)
SOLVE-GP	$1024 + 1024$	-0.509(0.007)	**0.395**(0.004)	-0.189(0.006)	0.272(0.003)
SVGP	2048	**-0.507**(0.007)	**0.395**(0.004)	-0.193(0.006)	0.276(0.003)

Table 8: Convolutional GPs for CIFAR-10 classification.

		Train Acc	Train LL	Test Acc	Test LL	Time
SVGP	1000	77.81%	-1.36	66.07%	-1.59	0.241 s/iter
	1600	78.44%	-1.26	67.18%	-1.54	0.380 s/iter
SOLVE-GP	$1000 + 1000$	79.32%	-1.20	**68.19%**	**-1.51**	0.370 s/iter
SVGP	2000	79.46%	-1.22	68.06%	**-1.48**	0.474 s/iter

Table 9: 2-layer deep convolutional GPs for CIFAR-10 classification.

	Inducing Points	Train Acc	Train LL	Test Acc	Test LL	Time
SVGP	$384, 1K$	84.86%	-0.82	76.35%	-1.04	0.392 s/iter
SOLVE-GP	$384 + 384, 1K + 1K$	87.59%	-0.72	**77.80%**	**-0.98**	0.657 s/iter
SVGP	$768, 2K$	87.25%	-0.74	77.46%	**-0.98**	1.104 s/iter

Table 10: 3-layer deep convolutional GPs for CIFAR-10 classification.

	Inducing Points	Train Acc	Train LL	Test Acc	Test LL	Time
SVGP	$384, 384, 1K$	87.70%	-0.67	78.76%	-0.88	0.418 s/iter
SOLVE-GP	$(384 + 384) \times 2, 1K + 1K$	89.88%	-0.57	**80.30%**	**-0.79**	0.752 s/iter
SVGP	$768, 768, 2K$	90.01%	-0.58	**80.33%**	-0.82	1.246 s/iter