Amortized Variational Deep Kernel Learning

Alan L. S. Matias [1] **César Lincoln C. Mattos** [1] **João P. P. Gomes** [1] **Diego P. P. Mesquita** [2]

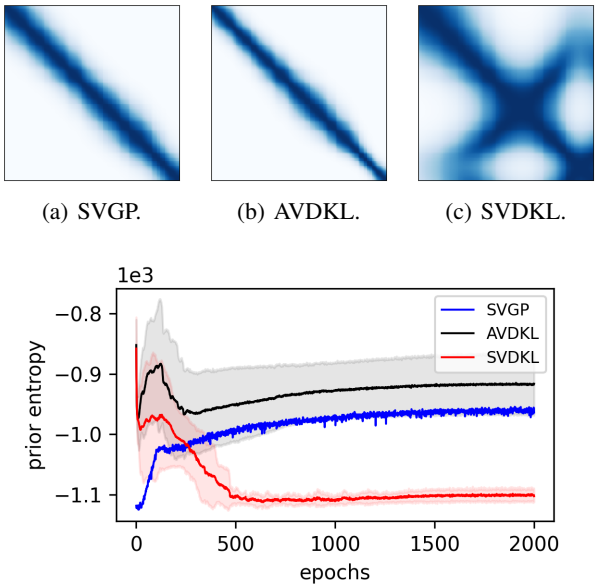
Abstract

Deep kernel learning (DKL) marries the uncertainty quantification of Gaussian processes (GPs) and the representational power of deep neural networks. However, training DKL is challenging and often leads to overfitting. Most notably, DKL often learns "non-local" kernels — incurring spurious correlations. To remedy this issue, we propose using amortized inducing points and a parameter-sharing scheme, which ties together the amortization and DKL networks. This design imposes an explicit dependency between the ELBO's model fit and capacity terms. In turn, this prevents the former from dominating the optimization procedure and incurring the aforementioned spurious correlations. Extensive experiments show that our resulting method, *amortized varitional DKL* (AVDKL), i) consistently outperforms DKL and standard GPs for tabular data; ii) achieves significantly higher accuracy than DKL in node classification tasks; and iii) leads to substantially better accuracy and negative log-likelihood than DKL on CIFAR100.

1. Introduction

In the late '90s, preceding the advent of the deep learning era, Gaussian Processes (GPs) gained popularity as alternatives to neural networks (NNs), owing to their inherent uncertainty quantification and frequently superior performance (McKay, 1998). During this period, benchmarking datasets primarily consisted of tabular data, fostering the perception that it was reasonable to completely bypass representation learning, concentrating solely on the relationship between vectors in the input space. However, in the early 2010s, a series of works harnessing the inductive biases of domain-specific NNs redirected the community's focus toward more intricate domains, encompassing images (Krizhevsky et al., 2012), sequences (Mikolov et al., 2010), and speech (Hinton et al., 2012). This shift in attention, however, did not favor GPs due mainly to the challenge of tailoring kernels that embody useful inductive biases for these diverse domains. In response, works proposed methods for learning GP kernels directly from data (e.g., Wilson & Adams, 2014; Duvenaud et al., 2013). These approaches aimed to enhance the adaptability of GPs to diverse datasets by capturing relevant features automatically.

In this context, deep kernel learning (DKL) emerged as a solution to the kernel versus NN dichotomy (Wilson et al., 2016a) along with its scalable variant: stochastic variational DKL (SVDKL) (Wilson et al., 2016b). DKL combines the strengths of both approaches, using an NN to extract



(a) SVGP. (b) AVDKL. (c) SVDKL.

(d) Prior entropy along the training epochs.

Figure 1. Example of the prior covariance matrix computed via the kernel function for SVGP, AVDKL and SVDKL, considering an 1D toy-dataset (see Figure 4(a)). Both AVDKL and SVDKL perform nonlinear input projection before applying the kernel function. One can notice that the SVDKL induces undesired correlations, since the original data comes from a smooth GP sample with squared exponential kernel. The same effect can also be seen in Figure 1(d), where the SVDKL, after a few training epochs, decreases too much the prior entropy.

[1]Federal University of Ceará [2]Getulio Vargas Foundation. Correspondence to: Alan Matias <matiasalsm@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

meaningful representations that are subsequently fed into a standard GP. Importantly, the DKL framework allows for learning all model parameters in end-to-end fashion. Both kernel hyperparameters and network parameters can be jointly trained, either by maximizing the marginal likelihood or using variational inference (Wilson et al., 2016b).

Nonetheless, the flexibility of deep NNs may be a double-edged sword when training DKL models. In particular, Ober et al. (2021) have identified a behavior in DKL leading to severe overfitting. DKL tends to correlate all input points, "hacking" the Gram matrix to reduce the impact of the complexity penalty term of the marginal likelihood, as illustrated in Figure 1. Ober et al. (2021) overcome this issue by carrying full Bayesian inference over the network parameters, for instance, by using Markov Chain Monte Carlo (MCMC) methods, such as Hamiltonian Monte Carlo (Neal et al., 2011). However, MCMC methods scale poorly for high-dimensional posteriors (e.g., over NN parameters), and monitoring their convergence can be challenging.

In this work, we propose amortized variational DKL (AVDKL), a novel method that counteracts the overfitting of DKL by locally smoothing predictions. More specifically, AVDKL uses deep NNs to both i) compute the embeddings which will be fed into the kernel function and ii) learn variational distributions over inducing points in an amortized manner. Such an amortization allows AVDKL to yield different variational mean and covariance structures for different regions of the input space, inspired by the recent proposition of input-dependent sparse GPs (Jafrasteh et al., 2022). To achieve this local effect, we share all parameters of both NN components underlying AVDKL, except for the last layers. We argue that the latter effectively avoids the spurious correlations of DKL and better adjusts the predictive variances. AVDKL reaps the natural uncertainty quantification of GPs, while avoiding the issues of DKL and leveraging the inductive biases provided by NNs in complex data domains.

Results show that AVDKL often outperforms DKL and conventional neural networks for tasks on tabular, image, and graph data. In particular, AVDKL usually results in better accuracy, negative log-likelihood, and calibration error than DKL or standard neural networks. Moreover, AVDKL often converges faster than DKL. The full code for AVDKL and the experiments is available at `https://github.com/alanlsmatias/amortized-variational-dkl`.

In summary, our contributions are:

1. We propose AVDKL, a hybrid GP/NN model that leverages NN-based representation learning without the caveat of overfitting;
2. We show that AVDKL effectively avoids DKL's typical issue, in which all points in the input space become correlated, causing overfitting;
3. We promote a varied experimental campaign showcasing the performance of AVDKL for tabular, image, and graph domains. In most cases, AVDKL outperforms DKL and standard NNs in terms of log-likelihood, accuracy, and calibration error.

2. Gaussian Process Background

We say a function $f : \mathcal{X} \to \mathbb{R}$ follows a GP if, for every finite set $\{\boldsymbol{x}_n\}_{n=1}^N \subset \mathcal{X}$, the sequence $\boldsymbol{f} = [f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_N)]^T$ follows a multivariate normal distribution $p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}, \boldsymbol{K})$, with $\mu_i = \mu(\boldsymbol{x}_i)$, $\mu : \mathcal{X} \to \mathbb{R}$, and $[\boldsymbol{K}]_{ij} = k_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $k : \mathcal{X}^2 \to \mathbb{R}$ is the covariance/kernel function and θ represents a set of kernel hyperparameters. Following the standard procedure within the GP literature, we assume μ is a constant function, assigning zero to any input. For Gaussian observations, the predictive distribution $p(f_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*) = \mathcal{N}(f_*|\mu_*, \sigma_*^2)$ is an analytical Gaussian with mean and variance given by

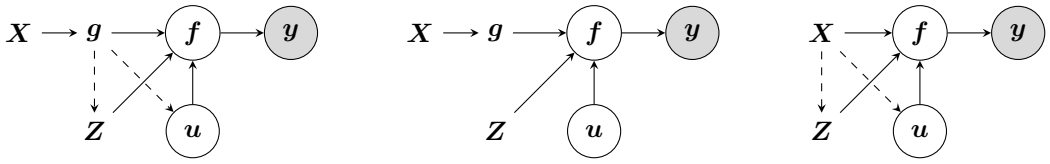
$$\mu_* = \boldsymbol{k}_{f_*}^T (\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I})^{-1} \boldsymbol{y}$$
$$\sigma_*^2 = K_{f_*} - \boldsymbol{k}_{f_*}^T (\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I})^{-1} \boldsymbol{k}_{f_*}.$$

Consequently, the predictive distribution for y_* is given by $p(y_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*) = \mathcal{N}(y_*|\mu_*, \sigma_*^2 + \sigma_y^2)$. Traditionally, k also has hyperparameters, which can be learned via maximization of the analytical marginal log-likelihood — also known as the *evidence*. We refer the interested reader to the classic reference by Rasmussen & Williams (2006) for a more thorough introduction to GPs.

Stochastic Variational GP. Computation of GP predictions and evidence scales roughly cubically with the number of observations N. Sparse GPs (SGPs) (Titsias, 2009; Hensman et al., 2013) aim to reduce this cost by considering $M \ll N$ inducing points $\boldsymbol{u} \in \mathbb{R}$ from the same GP prior of $\boldsymbol{f}$, i.e., $p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{K}_u)$. The inducing points are related to pseudo-inputs $\boldsymbol{z}_m \in \mathbb{R}^D$, $m = 1, \dots, M$, which lie in the same space as $\boldsymbol{x}_i|_{i=1}^N$ and are collected in the matrix $\boldsymbol{Z}$. Defining the variational distributions $q(\boldsymbol{f}, \boldsymbol{u}) = q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})$ and $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{S})$, the variational parameters $\boldsymbol{m}$ and $\boldsymbol{S}$ and the kernel hyperparameters are learned by maximizing the evidence lower bound (ELBO) (Hensman et al., 2013):

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_{q(\boldsymbol{u})p(f_i|\boldsymbol{u})}[\log p(y_i|f_i)] - \mathrm{KL}[q(\boldsymbol{u})||p(\boldsymbol{u})], \quad (1)$$

where $p(f_i|\boldsymbol{u})$ is Gaussian, since f_i and $\boldsymbol{u}$ are jointly Gaussian, and $\mathrm{KL}[\cdot||\cdot]$ is the Kullback-Leibler divergence between two distributions. Since the ELBO factorizes over the observations and $M \ll N$, one can drastically reduce computational cost to train the model using mini-batches, resulting in the stochastic variational GP (SVGP) approach.



(a) AVDKL. (b) SVDKL. (c) IDSGP.

Figure 2. Models schematic dependencies. $\boldsymbol{g}$ is the latent representation of $\boldsymbol{X}$, $\boldsymbol{Z}$ are the inducing points and $\boldsymbol{u}$ is the random variable related to the variational distribution $q(\boldsymbol{u})$.

Stochastic Variational DKL. Both DKL (Wilson et al., 2016a) and SVDKL (Wilson et al., 2016b) estimate complex kernels by leveraging the property that for any valid kernel function $k_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and any function $g(\cdot)$, the expression $k_\theta(g(\boldsymbol{x}_i), g(\boldsymbol{x}_j))$ is also a valid kernel function (Rasmussen & Williams, 2006). DKL considers an NN to learn a mapping function $g(\boldsymbol{x}_i)$ whose outputs are fed to standard kernel functions, such as the squared exponential. SVDKL follows the ELBO in Equation (1) to enable stochastic training and achieve better scaling with the number of observations.

Input Dependent Sparse GP. The IDSGP method (Jafrasteh et al., 2022) follows the sparse GP variational framework (Titsias, 2009; Hensman et al., 2013), but instead of a fixed variational distribution $q(\boldsymbol{u})$, IDSGP computes inducing locations for each meta-point $\tilde{\boldsymbol{x}} \sim p(\tilde{\boldsymbol{x}})$, where $p(\tilde{\boldsymbol{x}})$ is a prior distribution with unknown analytical form. This approach yields $q(\boldsymbol{u}|\tilde{\boldsymbol{x}})$, making the inducing locations input dependent, resulting in an amortized variational inference (Kingma & Welling, 2014; Rezende et al., 2014). The full variational distribution is approximated by $q(\boldsymbol{f}, \boldsymbol{u}, \tilde{\boldsymbol{x}}) = p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u}|\tilde{\boldsymbol{x}})p(\tilde{\boldsymbol{x}})$, which yields the following ELBO (Tran et al., 2021; Jafrasteh et al., 2022):

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u},\tilde{\boldsymbol{x}})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u}|\tilde{\boldsymbol{x}})p(\tilde{\boldsymbol{x}})}{p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u}|\tilde{\boldsymbol{x}})p(\tilde{\boldsymbol{x}})}\right] \\ &= \sum_{i=1}^{N}\int p(\tilde{\boldsymbol{x}})\left[p(f_i|\boldsymbol{u})q(\boldsymbol{u}|\tilde{\boldsymbol{x}})\log p(y_i|f_i)\mathrm{d}\boldsymbol{f}\mathrm{d}\boldsymbol{u}\right. \\ &\left. - \frac{1}{N}\mathrm{KL}[q(\boldsymbol{u}|\tilde{\boldsymbol{x}})||p(\boldsymbol{u}|\tilde{\boldsymbol{x}})]\right]\mathrm{d}\tilde{\boldsymbol{x}},\end{aligned} \tag{2}$$

where $p(\tilde{\boldsymbol{x}})$ is assumed to be an implicit distribution. By sampling $\tilde{\boldsymbol{x}}_s$ from $p(\tilde{\boldsymbol{x}})$ we can approximate Equation (2), leading to the IDSGP's ELBO:

$$\begin{aligned}\mathcal{L}_{\text{IDSGP}} = &\sum_{i=1}^{N}\mathbb{E}_{p(f_i|\boldsymbol{u})q(\boldsymbol{u}|\tilde{\boldsymbol{x}}_s)}\left[\log p(y_i|f_i)\right] \\ &- \frac{1}{N}\sum_{i=1}^{N}\mathrm{KL}\left[q(\boldsymbol{u}|\tilde{\boldsymbol{x}}_s)||p(\boldsymbol{u}|\tilde{\boldsymbol{x}}_s)\right].\end{aligned} \tag{3}$$

The above ELBO is valid for any implicit distribution $p(\tilde{\boldsymbol{x}})$ (Tran et al., 2021) and the IDSGP can be optimized through stochastic optimization by evaluating its gradients. In particular, the IDSGP assumes a mini-batch training and sets $\tilde{\boldsymbol{x}}_s = \boldsymbol{x}_i$. Thus, the values of $\tilde{\boldsymbol{x}}$ are still random, since they are selected from randomized mini-batches. In this sense, the variational parameters become input-dependent (denoted by the subscript index): the pseudo-inputs $\boldsymbol{Z}_i \in \mathbb{R}^{M\times D}$, the mean $\boldsymbol{m}_i \in \mathbb{R}^M$, and the lower-triangular matrix $\boldsymbol{L}_i \in \mathbb{R}^{M\times M}$. In summary, the variational distribution for the input-dependent inducing points is $q(\boldsymbol{u}_i|\boldsymbol{x}_i) = \mathcal{N}(\boldsymbol{u}_i|\boldsymbol{m}_i, \boldsymbol{L}_i\boldsymbol{L}_i^T)$. Additionally, the KL divergence must be averaged across the inputs (see Equation (3)).

3. Amortized Variational DKL

As argued by Ober et al. (2021), the main issue with DKL is that the NN parameters do not receive a Bayesian treatment. In this sense, the NN is free to project $\boldsymbol{x}_i$ and increase the marginal log-likelihood without balancing fit and regularization. As an example, consider the evidence of the standard exact GP, with omitted constant term:

$$\log p(\boldsymbol{y}|\boldsymbol{X}) \propto -\underbrace{\log|\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I}|}_{\text{model capacity}} - \underbrace{\boldsymbol{y}^T(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{y}}_{\text{model fit}}. \tag{4}$$

Ober et al. (2021) points out that after some epochs the model fit term stabilizes and the DKL optimizer moves on to minimize the model capacity term in Equation (4). This is achieved by increasing the correlations between all the data points, which turns the log determinant term smaller. This effect is illustrated in Figure 1. Furthermore, the authors empirically showed that when DKL is not able to correlate the data points, the marginal likelihood becomes worse.

As previously stated, amortized variational inference (A-VI) aims to approximate posterior distributions in probabilistic models. With A-VI, one can learn a function that estimates local latent variables given an input. As mentioned in Section 2, this idea was applied to variational sparse GPs in the IDSGP work (Jafrasteh et al., 2022). One important advantage of IDSGP is that one can drastically reduce the number of inducing points without losing performance, since they become input dependent. On the other hand, the IDSGP design has the limitation that the pseudo-inputs must lie in the same space as the input vectors, since we need to compute the kernel between those. This renders dealing

with unstructured data domains challenging, as it implies that the amortizing NNs must output complex objects.

Our work is inspired by both DKL and IDSGP and aims to tackle the main DKL problem. In essence, while the DKL provides means for dealing with unstructured data, the IDSGP is scalable and capable of learning complex mappings for the variational parameters. The proposed AVDKL changes the SVDKL framework by using the NN embedding $\boldsymbol{g}_i = g(\boldsymbol{x}_i) \in \mathbb{R}^E$, where E is the embedding size, to compute the inducing locations and the variational parameters $\boldsymbol{Z}_i, \boldsymbol{m}_i, \boldsymbol{L}_i$, so that we have $q(\boldsymbol{u}_i|\boldsymbol{g}_i)$ with ELBO

$$\begin{aligned}\mathcal{L}_{\text{AVDKL}} = &\sum_{i=1}^{N} \mathbb{E}_{q(\boldsymbol{u}|\boldsymbol{g}_i)p(f_i|\boldsymbol{u})}\left[\log p(y_i|f_i)\right] \\ &- \frac{1}{N}\sum_{i=1}^{N} \text{KL}\left[q(\boldsymbol{u}|\boldsymbol{g}_i)||p(\boldsymbol{u}|\boldsymbol{g}_i)\right].\end{aligned} \tag{5}$$

Prediction for a new $\boldsymbol{x}_*$ is given by integating out $\boldsymbol{u}$:

$$\begin{aligned}q(f_*|\boldsymbol{x}_*) &= \int p(f_*|\boldsymbol{u})q(\boldsymbol{u}|\boldsymbol{g}_*)\mathrm{d}\boldsymbol{u} \\ &= \int \mathcal{N}(f_*|\,\boldsymbol{A}\boldsymbol{u}, K_* - \boldsymbol{A}\boldsymbol{k}_{*u}^\top)\mathcal{N}(\boldsymbol{u}|\boldsymbol{m}_*, \boldsymbol{S}_*)\mathrm{d}\boldsymbol{u} \\ &= \mathcal{N}\left(f_*\middle|\boldsymbol{A}\boldsymbol{m}_*, K_* - \boldsymbol{A}(\boldsymbol{K}_u - \boldsymbol{S}_*)\boldsymbol{A}^T\right),\end{aligned}$$

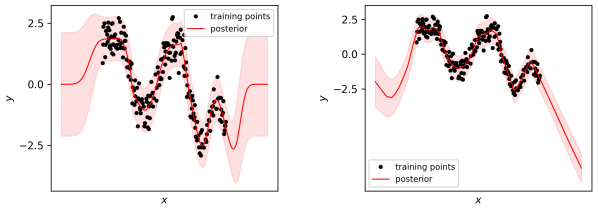
where $\boldsymbol{A} = \boldsymbol{k}_{*u}\boldsymbol{K}_u^{-1}$.

In Figure 2 we depict the graphical models of SVDKL, IDSGP and AVDKL — the dashed lines specify amortization. The difference between the AVDKL and the other models can be identified by the connection from $\boldsymbol{g}$ to $\boldsymbol{Z}$ and $\boldsymbol{u}$. In practice, we generate $\boldsymbol{Z}$ and the parameters of $q(\boldsymbol{u})$ through an additional mapping from $\boldsymbol{g}$. Such a mapping is comprised of a nonlinear layer and a final linear transformation. Importantly, the nonlinear activation function has a saturation, e.g., by applying a Sigmoid or Tanh function directly over $\boldsymbol{g}$ or its projection. The relevance of the saturation lies on the mapping of the variational parameters for points that are far from the training data. If no saturation function was applied, the amortization would try to extrapolate the variational parameters and the corresponding approximate posterior, which, as a consequence, would lead to poor uncertainty estimates. Figure 3 exemplifies the latter undesired behavior.

3.1. Attenuating the Overcorrelation Issue

In the AVDKL ELBO (Eq. (5)), the KL divergence between the variational posterior $q(\boldsymbol{u}|\boldsymbol{g}_i)$ and the prior $p(\boldsymbol{u}|\boldsymbol{g}_i)$ can be written as

$$\begin{aligned}\text{KL}\left[q(\boldsymbol{u}|\boldsymbol{g}_i)||p(\boldsymbol{u}|\boldsymbol{g}_i)\right] &= \int q(\boldsymbol{u}|\boldsymbol{g}_i)\log\frac{q(\boldsymbol{u}|\boldsymbol{g}_i)}{p(\boldsymbol{u}|\boldsymbol{g}_i)}\mathrm{d}\boldsymbol{u} \\ &= \mathcal{H}_q[p(\boldsymbol{u}|\boldsymbol{g}_i)] - \mathcal{H}_q[q(\boldsymbol{u}|\boldsymbol{g}_i)],\end{aligned}$$



(a) AVDKL with Tanh. (b) AVDKL with ReLU.

Figure 3. Example of the AVDKL applied to synthetic 1D data using different activation functions before the variational parameters mapping. In Fig. 3(a) a satured function is used, which prevents the variational parameters to perform undesired extrapolation.

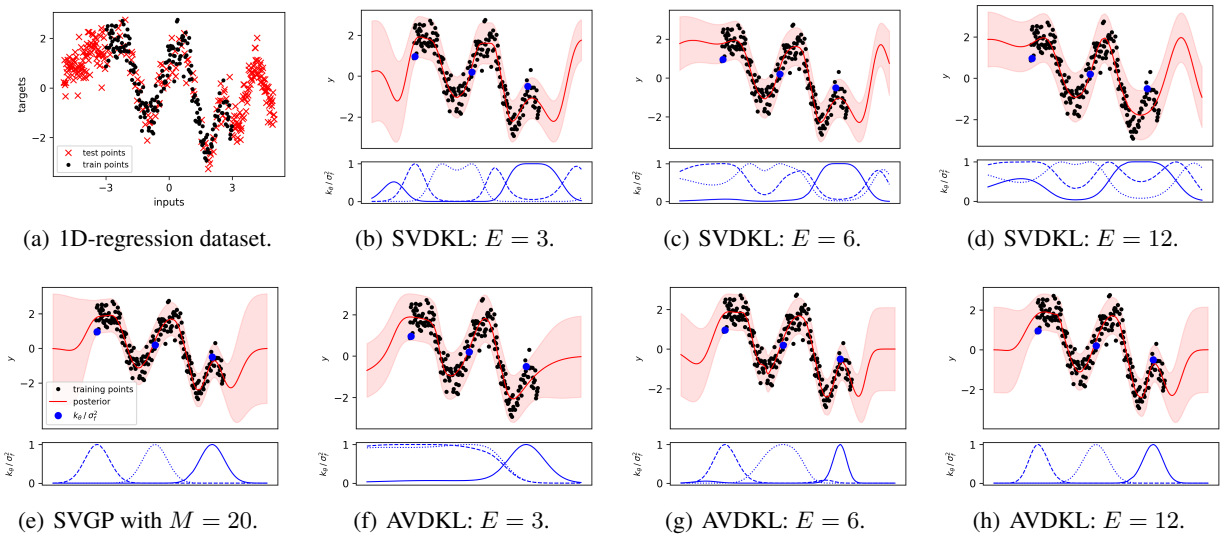
where $\mathcal{H}_q[\cdot]$ denotes the entropy operation with respect to the distribution $q(\boldsymbol{u}|\boldsymbol{g}_i)$. In the above, the first term is the cross-entropy between the variational posterior and the prior, while the second is the entropy of the variational posterior.

The maximization of the ELBO corresponds to the minimization of the above KL divergence, which in turn corresponds to the minimization of the cross-entropy and the maximization of the variational entropy. The former involves pulling the variational posterior closer to the prior, which regularizes the approximation. The latter, considering $q(\boldsymbol{u}|\boldsymbol{g}_i) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}_i, \boldsymbol{S}_i)$, where $\boldsymbol{S}_i = \boldsymbol{L}_i\boldsymbol{L}_i^T$, is proportional to $\log|\boldsymbol{S}_i|$. Thus, if the covariance matrix $\boldsymbol{S}_i$ correlates too much the inducing points, its determinant would decrease along with the entropy of $q(\boldsymbol{u}|\boldsymbol{g}_i)$, which would penalize the ELBO. Importantly, these effects only happen because $\boldsymbol{S}_i$ is not a model parameter, but a variational parameter.

In the stochastic version of the DKL, the variational parameters $\boldsymbol{m}$, $\boldsymbol{S}$ and $\boldsymbol{Z}$ (the latter which is used to compute $\boldsymbol{K}_u$) are not coupled with the input projection neural network (see Figure 2(b)), so their optimization do not directly affects the network optimization and vice-versa. In the AVDKL formulation, they are generated from the same neural feature extractor used in the kernel learning component, which yields $\boldsymbol{g}_i$ (see Figure 2(a)). Thus, the neural feature extractor and the amortization network share parameters, which become variational parameters that aid in guarding against overfitting (Titsias, 2009; Dai et al., 2016). Such a coupling is critical to allow the network to leverage the above balance effect of the KL term and avoid the issue of excessive correlation by the embedding layers.

3.2. Initial investigation

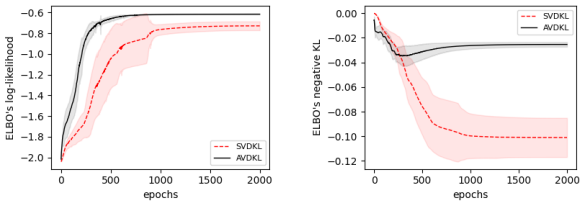
We perform an initial investigation considering an 1D-regression toy dataset, depicted in Figure 4(a). The data points were generated by sampling from a zero mean GP with a squared exponential kernel, hyperparameters $\sigma_f^2 = 1$, $l = 0.5$, and noise variance $\sigma_y^2 = 0.2$. We also include in



(a) 1D-regression dataset. (b) SVDKL: $E = 3$. (c) SVDKL: $E = 6$. (d) SVDKL: $E = 12$.

(e) SVGP with $M = 20$. (f) AVDKL: $E = 3$. (g) AVDKL: $E = 6$. (h) AVDKL: $E = 12$.

Figure 4. 1D-regression dataset posterior and correlation analysis. The dataset was generated from a GP with the squared exponential kernel parameterized by $\sigma_f^2 = 1$, $l = 0.5$ and noise covariance $\sigma_y^2 = 0.2$. The SVGP model is depicted in Figure 4(e) to act as a baseline for AVDKL and SVDKL. As one can see, the SVDKL is not suitable with the prior from where the data was generated. On the other hand, the AVDKL with high embedding sizes, $E = 6$ and $E = 12$, have correlations consistent with the prior. Furthermore, compared to the SVDKL, the AVDKL posterior is better calibrated in terms of uncertainty estimates.



(a) ELBO's log-likelihood. (b) ELBO's negative KL.

Figure 5. ELBO's log-likelihood and KL terms for SVDKL and AVDKL with $E = 12$ over 10 random weight initializations.

Figure 4(e) the SVGP posterior and the kernel function k_θ values between the test points and 3 fixed reference points. We consider SVGP as as baseline to compare both AVDKL and SVDKL with varying embedding sizes. The optimization was carried out using AdamW (Loshchilov & Hutter, 2019) with learning rate 0.01 and a weight decay of 0.001 for the NN's weights and biases. We also used a cosine annealing learning rate scheduler (Loshchilov & Hutter, 2017). Both SVGP and SVDKL were trained with 20 inducing points, while the AVDKL used only 2. Concerning the feature extractor, we used a NN with two hidden layers of sizes $[32, E]$, batch normalization (Ioffe & Szegedy, 2015), and SiLU (Swish) activation (Ramachandran et al., 2017) on the hidden layer. Finally, we used a Tanh activation at the output saturation layer of AVDKL's amortization component.

From Figure 4(e), one can notice that the SVGP presents a well calibrated posterior and the correlation computed using the kernel function behaves as expected for the squared exponential used in the original data generation step, with high correlation between near points and low correlations between distant points.

On the other hand, the correlations of the SVDKL are not consistent with the prior the data was sampled from. Moreover, for high dimensional embeddings, SVDKL suffers from poor fitting (see Figure 4(b), Figure 4(c), and Figure 4(d)). The irregular correlations presented by the SVDKL was also pointed out by Ober et al. (2021): the model tends to excessively correlate the projections $\boldsymbol{g}_i$.

Contrary to SVDKL, the AVDKL is able to better estimate the posterior for high embedding sizes, with correlations between the embeddings being more coherent with $E = 12$ (see Figure 4(f), Figure 4(g) and Figure 4(h)).

We also analyze the behavior of the KL divergence component of the ELBO. Unlike the SVDKL, the variational distribution of the AVDKL is coupled with feature extractor that generates $\boldsymbol{g}_i$. In this sense, while the SVDKL feature extractor is free for adjusting the fitting term of the ELBO without much interference of the KL divergence term, in the AVDKL, both terms are more closely related. Thus, the latter is able to jointly learn complex embeddings and variational parameters representations while still being regularized by the prior $p(\boldsymbol{u}|\boldsymbol{g}_i)$. This behavior can be seen in Figure 5, where the AVDKL has a KL divergence much closer to the prior (i.e., a KL closer to zero) and a better fitting (i.e., a higher ELBO).

Table 1. MNLL for AVDKL, SVDKL and SVGP on tabular classification and regression datasets. The AVDKL achieved the best test MNLL for 5 datasets out of 6. The most significant results can be seen in the regression tasks.

Model	Classification			Regression		
	MagicGamma	HTRU2	Letter	Protein	KeggD	KeggU
AVDKL	0.286 ± 0.013	$\mathbf{0.066 \pm 0.004}$	$\mathbf{0.084 \pm 0.003}$	$\mathbf{2.735 \pm 0.015}$	$\mathbf{-2.283 \pm 0.024}$	$\mathbf{-3.004 \pm 0.311}$
GDKL	$\mathbf{0.284 \pm 0.014}$	0.067 ± 0.004	0.101 ± 0.002	2.825 ± 0.014	-1.979 ± 0.010	-
DLVKL	0.298 ± 0.012	0.069 ± 0.003	0.205 ± 0.007	2.875 ± 0.006	-1.911 ± 0.011	-2.772 ± 0.030
SVDKL	0.299 ± 0.018	0.067 ± 0.005	0.190 ± 0.014	2.844 ± 0.011	-1.916 ± 0.016	-2.705 ± 0.036
SVGP	0.303 ± 0.011	0.068 ± 0.002	0.171 ± 0.004	2.854 ± 0.008	-1.889 ± 0.011	-2.581 ± 0.019

4. Experiments

We evaluate the proposed AVDKL on a set of different tasks: tabular data, semi-supervised node classification and image classification. We access the model performance in terms of accuracy (classification) and mean negative log-likelihood (MNLL) (regression and classification). For image classification we also report the Expected Calibration Error (ECE).

4.1. Tabular data

We start our experiments with popular classification and regression datasets from the UCI repository (Kelly et al., 2023). We summarize the datasets, including the mini-batch size used for training, in Table 4. For evaluation, we split the datasets into five different train/test sets, with each split having 80% of the data for training and 20% for testing.

We compared AVDKL to GDKL (Achituve et al., 2023), DLVKL (Liu et al., 2021), SVDKL, and SVGP. Importantly, the SVDKL for classification was implemented with the inducing grid interpolation strategy (Wilson et al., 2016b) (i.e., each embedding feature is a GP task and the output logits are computed by combining the GP tasks via a linear mapping). The SVDKL for regression was implemented as a regular SVGP model coupled with kernel learning (i.e., the kernel function is computed over the embedding $\boldsymbol{g}$).

We train all models with zero mean and the squared exponential kernel, and a Gaussian (Softmax) likelihood for regression (classification). For GDKL, DLVKL and SVGP, we used 500 inducing points for all datasets. The SVDKL was trained with 500 inducing points for regression and with a inducing grid of size 50 for classification. The AVDKL was trained with 3 inducing points on the binary classfication tasks and with 15 inducing points for the remaining datasets. For all NN-based models, AVDKL, GDKL, DLVKL and SVDKL, we used a NN with two layers of size $[64, D]$, where the first layer includes batch normalization and a SiLU activation. Finally, we trained all models for 200 epochs using AdamW optimizer with learning rate 0.005 (0.01 for the SVGP) and weight decay 0.001 (applied to AVDKL, GDKL, DLVKL, and SVDKL networks).

We report the test MNLL of all models for each dataset in Table 1. In summary, the AVDKL was able to get the best test MNLL in 5 datasets out of 6. The most significant improvements in terms of performance regarding the AVDKL are related to the regression tasks. Importantly, we trained the GDKL model with a pre-computed covariance matrix in order to improve the training time. As a consequence, for the KeggU dataset, we incurred in memory overflow and could not report its performance metrics.

4.2. Semi-supervised node classification

For the graph node classification task, we used three popular citation-network datasets: Cora, CiteSeer and PubMed. The datasets present countings as textual node features and classes denoting topics of papers (Yang et al., 2016). We consider the fixed splits defined by Yang et al. (2016), with 20 nodes per class for training, 500 nodes for validation and 1000 nodes for testing. The datasets are summarized in Table 5. We evaluate the AVDKL against SVDKL and a DeepGCN model in terms of test Accuracy and MNLL.

The feature extractor for both AVDKL and SVDKL is a DeepGCN with Simplified Graph Convolution (SGC) (Wu et al., 2019) layers and initial connection $\alpha = 0.1$ (Chen et al., 2020; 2022). The layer size, convolutions depth, dropout rate, learning rate and weight decay were chosen according to the "sweet point" described by Chen et al. (2022) (see Table 6). We trained the AVDKL with zero mean, $\text{Matern}_{5/2}$ kernel, Softmax likelihood, 2 inducing points and a Tahn activation for saturation. The SVDKL follows the same settings as the AVDKL, but with an inducing grid of size 50. Initially, the SVDKL was not able to converge on the CiteSeer dataset with the settings described in Table 6. To overcome this, we progressively reduced the dropout rate until the SVDKL was able to converge. In this sense, we used a dropout rate of 0.3 specifically for the SVDKL on the CiteSeer dataset. Furthermore, all models were trained from scratch using the Adam optimizer (Kingma & Ba, 2015) for 1000 epochs with early-stopping window of 100.

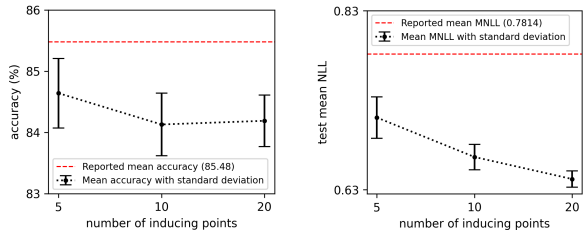
The results were computed over 30 random initialization and are reported in Table 2. Compared to the DeepGCN, the

Table 2. Accuracy and ECE for AVDKL, SVDKL and DeepGCN on citation-network datasets. The AVDKL is slightly better in accuracy in comparison to the DeepGCN. Concern the SVDKL, despite the lower test MNLL, the accuracy is much worse when compared to both AVDKL and DeepGCN.

Model	Cora		CiteSeer		PubMed	
	Acc. (%)	MNLL	Acc. (%)	MNLL	Acc. (%)	MNLL
AVDKL	**85.48 ± 0.51**	0.7814 ± 0.0174	**72.54 ± 0.42**	0.9438 ± 0.0089	**80.22 ± 0.30**	0.5317 ± 0.0041
SVDKL	79.83 ± 1.27	**0.6590 ± 0.0488**	69.96 ± 0.91	**0.9384 ± 0.0203**	78.05 ± 1.27	0.5996 ± 0.0196
DeepGCN	85.47 ± 0.44	0.8091 ± 0.0127	72.45 ± 0.50	0.9670 ± 0.0134	79.91 ± 0.30	**0.5186 ± 0.0041**

AVDKL is in general better in accuracy and better in MNLL for the Cora and CiteSeer datasets. However, against the SVDKL, the AVDKL obtained worse MNLL but a much better accuracy. We found in our experiments that the number of inducing points of the AVDKL, specifically for this task, trades-off between accuracy and MNLL. In this sense, we analysed the test accuracy and MNLL for different number of inducing points. We show in Figure 6 that by increasing the number of inducing points one can reach better MNLL at the cost of loosing accuracy – the AVDKL with 20 inducing points was able get an MNLL of 0.6420 ± 0.0091.

In Figure 7 we show the test MNLL over the training epochs averaged over 30 random initializations. It is possible to note how both AVDKL and DeepGCN curves are more stable than the SVDKL. Also, compared to the DeepGCN, the AVDKL convergence was in general faster and produced better models in terms of uncertainty calibration.

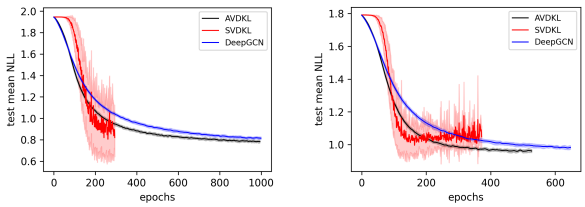


(a) Cora. (b) CiteSeer.

Figure 6. AVDKL test accuracy/MNLL for the Cora dataset with different number of inducing points. The red horizontal line represents the results reported in Table 2. While the test accuracy deteriorates with more inducing points, the test MNLL improves.

4.3. Image classification

In this section, we evaluate AVDKL's performance on image classification, comparing it to SVDKL and a standalone ResNet-18. For both AVDKL and SVDKL we use the same ResNet-18 architecture as feature extractor. We train all models from scratch on CIFAR10 and CIFAR100 datasets. Each dataset comprises 50K images for training and 10K images for testing. We compare all models in terms of test accuracy, MNLL, and ECE averaged over 3 repetitions.



(a) Cora. (b) CiteSeer.

Figure 7. Test MNLL over the training epochs for Cora and Citeseer (notice that early stopping was used). The curves are averages over 30 random initializations and the confidence intervals are related to 1 standard deviation.

We use the same recipe for training all models (details can be found in Table 7) : 300 epochs using SGD optimizer with initial learning rate 0.1, weight decay 0.0001, momentum 0.9, and cosine annealing learning rate scheduler. We train both AVDKL and SVDKL with zero mean, squared exponential kernel, and Softmax likelihood. We use 15 inducing points for AVDKL and a grid of size 64 for SVDKL. Concerning the AVDKL saturation, we use a module consisting of a linear layer with size $E/8$ followed by a Sigmoid activation.

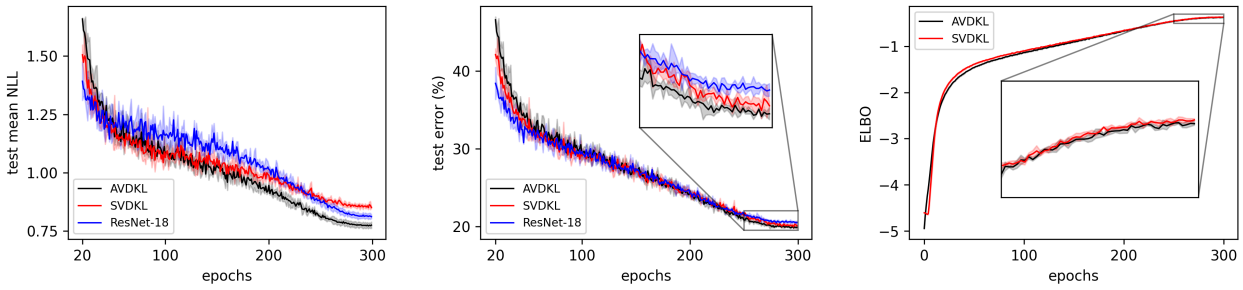
We report the results in Table 3. AVDKL was superior for both datasets in terms of MNLL, with a wider gap for CIFAR100. Regarding accuracy, AVDKL was on par with SVDKL on CIFAR10 and slightly better on CIFAR100. Besides having better uncertainty estimates, AVDKL is also better calibrated than its counterparts (smaller ECE).

In Figure 8 we report the test MNLL and error (Figure 8(a) and Figure 8(b), respectively) through the training phase along with the ELBO curve for both AVDKL and SVDKL (Figure 8(c)). As one can see, the AVDKL's test MNLL was superior for the most part of the training process. Interesting, the SVDKL ELBO was equivalent to the AVDKL ELBO during all the training process, but the test MNLL is worse – note SVDKL's test MNLL is even worse than ResNet-18's.

We have also carried out a experiment to measure the efficacy of the AVDKL for different dataset sizes. We used the CIFAR10 dataset and evaluated the test MNLL and error. The results are depicted in Figure 9. From Figure 9(a), the

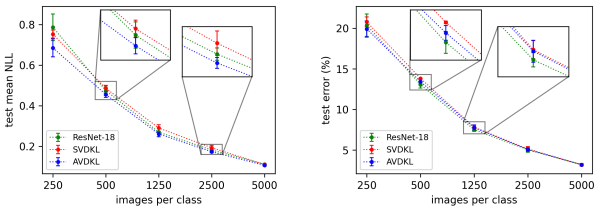
Table 3. Accuracy, MNLL and ECE for AVDKL, SVDKL and ResNet-18 on CIFAR10 and CIFAR100. The AVDKL has better test MNLL and ECE for both datasets. Its accuracy is also slightly better for the CIFAR100 dataset.

Model	CIFAR10			CIFAR100		
	MNLL	Acc. (%)	ECE (%)	MNLL	Acc. (%)	ECE (%)
AVDKL	**0.1068 ± 0.0037**	96.81 ± 0.11	**1.41 ± 0.11**	**0.7755 ± 0.0135**	**80.13 ± 0.16**	**7.18 ± 0.25**
SVDKL	0.1107 ± 0.0021	**96.82 ± 0.16**	1.44 ± 0.11	0.8510 ± 0.0038	79.92 ± 0.10	9.22 ± 0.02
ResNet-18	0.1122 ± 0.0027	96.81 ± 0.07	1.50 ± 0.05	0.8121 ± 0.0082	79.50 ± 0.17	8.00 ± 0.45



(a) Test MNLL. (b) Test Error (%). (c) ELBO.

Figure 8. Test MNLL, test error and ELBO (AVDKL and SVDKL) training curves for CIFAR100 averaged on 3 weight initialization. As one can see, the AVDKL has better generalization capabilities in terms of test MNLL and error along the training epochs. Interesting, both AVDKL and SVDKL have equivalent ELBO but highly different test MNLL.



(a) Test MNLL (b) Test error (%)

Figure 9. Test MNLL and error on the CIFAR10 dataset for different number of images per class. Note that the AVDKL has better test MNLL for all training dataset sizes.

AVDKL achieved the best MNLL for all training dataset sizes. Even with 2500 images per class, the AVDKL was superior to the SVDKL. In Figure 9(b), one can see that the AVDKL has smaller error for 250 images per classes, and that the ResNet-18 test error becomes slightly lower than the AVDKL error in the 500 and 1250 images per classes settings. However, at the levels with 2500 and 5000 images per class, all models become equivalent.

5. Related Work

The use of neural networks within GPs formulation has been the subject of several previous efforts. As follows we summarize some of them.

Deep kernel learning for GPs. The DKL model was first introduced by Wilson et al. (2016a) working on ideas of KISS-GP (Wilson & Nickisch, 2015), by combining deep neural networks and standard covariance functions with local kernel interpolation to derive complex kernel functions. Despite DKL's ability of estimating complex kernels for GPs, it still had some important limitations: it was not suitable for classification and stochastic gradient training. In this sense, Wilson et al. (2016b) proposed the stochastic variational DKL (SVDKL), enabling the model for classification, multi-task learning, and stochastic gradient training. In a more general formulation, Calandra et al. (2016) proposed manifold GP by applying the kernel function over a parameterized transformation performed on the inputs; similar to DKL, such transformation can be parameterized by a neural network. The learning of complex kernels was also the topic of specific domains; Al-Shedivat et al. (2017) used a recurrent NN to learn kernels with recurrent structures, while Achituve et al. (2021) considered a NN for learning a shared kernel function in a personalized federated learning setting. Most recently, there was some efforts towards the regularization of DKL-based models. Liu et al. (2021) proposed a latent-variable framework that incorporates a stochastic encoding of the inputs for regularized representation learning. Achituve et al. (2023) also proposed a novel approach for training DKL models. The approach consists in leveraging the uncertainty estimation of a GP with an infinite-width deep neural

network kernel to guide the DKL optimization process.

Amortized variational inference for GPs. Amortization was initially investigated in the context of GPs as a back constraint approach (Lawrence & Quinonero-Candela, 2006; Bui & Turner, 2015). The terms "back constraint" and "amortization" are essentially interchangeable, as in both cases a neural network is employed to impose constraints on local smoothing. Lawrence & Quinonero-Candela (2006) used amortization to locally preserve distances in GP Latent Variable Models (GP-LVM). Bui & Turner (2015) revisited VI for GP-LVM and applied amortization to estimate variational parameters. Dai et al. (2016) applied amortization to reparameterize the variational posterior distribution. Mattos & Barreto (2019) made use of A-VI to constrain the model's local variables of recurrent GPs. Villacampa-Calvo et al. (2021) applied amortization on the variational posterior in order to reduce the variational distribution complexity. Jafrasteh et al. (2022) introduced the IDSGP by performing A-VI on the inducing locations.

Comparison to SVDKL and IDSGP. While AVDKL marries ideas from SVDKL and IDSGP, there are crucial design differences among these methods that are worth highlighting. In the DKL framework, the network only interacts directly with the kernel function. On the other hand, AVDKL is fully coupled with the GP on both terms of the ELBO by interacting with both kernel and variational parameters. Moreover, IDSGP performs the amortization using the input points and thus is not suitable for unstructured data domains. AVDKL takes advantage of SVDKL and IDSGP formulations to achieve scalable and complex representations for both kernel function and variational parameters.

6. Conclusion

In this work we proposed the use of amortized variational inference for learning scalable and complex kernels representations. With our formulation, the NN which projects the embeddings of $\boldsymbol{X}$ shares its parameters with the amortization mapping of the inducing points and variational distribution, ensuring that the NN have impact on both fitting and regularization terms of the evidence lower bound. We show on a set of extensive analysis and experiments that our model, AVDKL, is capable of attenuating the DKL tendency of over correlating the NN output features, producing well-regularized, calibrated and more accurate models.

Acknowledgments

Alan Matias, César Mattos and João Gomes acknowledge the support of FUNCAP (grant ITR-0214-00009.01.00/23). Diego Mesquita also acknowledges the support of CNPq (grant 404336/2023-0), FAPERJ (grant 200.151/2023), FAPESP (grant 2023/00815-6), and SVCF, through the Ripple impact fund.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Achituve, I., Shamsian, A., Navon, A., Chechik, G., and Fetaya, E. Personalized federated learning with gaussian processes. *Advances in Neural Information Processing Systems*, 34:8392–8406, 2021.

Achituve, I., Chechik, G., and Fetaya, E. Guided deep kernel learning. In *Uncertainty in Artificial Intelligence*, pp. 11–21. PMLR, 2023.

Al-Shedivat, M., Wilson, A., Saatchi, Y., Hu, Z., and Xing, E. Learning scalable deep kernels with recurrent structure. *The Journal of Machine Learning Research*, 18(1):2850–2886, 2017.

Bui, T. D. and Turner, R. E. Stochastic variational inference for gaussian process latent variable models using back constraints. In *Black Box Learning and Inference NIPS workshop*, 2015.

Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P. Manifold gaussian processes for regression. In *2016 International joint conference on neural networks (IJCNN)*, pp. 3338–3345. IEEE, 2016.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 2020.

Chen, T., Zhou, K., Duan, K., Zheng, W., Wang, P., Hu, X., and Wang, Z. Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2769–2781, 2022.

Dai, Z., Damianou, A., González, J., and Lawrence, N. Variational auto-encoded deep gaussian processes. In *International Conference on Learning Representations (ICLR)*, San Juan Puerto Rico, 2016. ICLR.

Duvenaud, D. K., Lloyd, J. R., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning (ICML)*, pp. 1166–1174, 2013.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

Han, I., Zandieh, A., Lee, J., Novak, R., Xiao, L., and Karbasi, A. Fast neural kernel embeddings for general activations. In *Advances in Neural Information Processing Systems*, 2022. URL https://github.com/google/neural-tangents.

Hensman, J., Fusi, N., and Lawrence, N. Gaussian processes for big data. In *Conference on Uncertainty in Artificial Intelligence*, 2013.

Hinton, G. E., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 2012.

Hron, J., Bahri, Y., Sohl-Dickstein, J., and Novak, R. Infinite attention: Nngp and ntk for deep attention networks. In *International Conference on Machine Learning*, 2020. URL https://github.com/google/neural-tangents.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Jafrasteh, B., Villacampa-Calvo, C., and Hernández-Lobato, D. Input dependent sparse Gaussian processes. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 9739–9759. PMLR, 2022.

Kelly, M., Longjohn, R., and Nottingham, K. Uci machine learning repository, https://archive.ics.uci.edu, 2023.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*. ICLR, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 2012.

Lawrence, N. and Quinonero-Candela, J. Local distance preservation in the GP-LVM through back constraints. In *international conference on Machine learning*, 2006.

Liu, H., Ong, Y.-S., Jiang, X., and Wang, X. Deep latent-variable kernel learning. *IEEE Transactions on Cybernetics*, 52(10):10276–10289, 2021.

Loshchilov, I. and Hutter, F. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Mattos, C. L. C. and Barreto, G. A stochastic variational framework for recurrent Gaussian processes models. *Neural Networks*, 112:54–72, 2019.

McKay, D. Bayesian semi-supervised learning with graph Gaussian processes. In Bishop, C. (ed.), *Neural Networks and Machine Learning*, chapter 11, pp. 133–165. Springer-Verlag, 1998.

Mikolov, T., Karafiát, M., Burget, L., Cernocky, J., and Khudanpur, S. Recurrent neural network based language model. *INTERSPEECH*, 2010.

Milios, D., Camoriano, R., Michiardi, P., Rosasco, L., and Filippone, M. Dirichlet-based gaussian processes for large-scale calibrated classification. *Advances in Neural Information Processing Systems*, 31, 2018.

Neal, R. M. et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL https://github.com/google/neural-tangents.

Novak, R., Sohl-Dickstein, J., and Schoenholz, S. S. Fast finite width neural tangent kernel. In *International Conference on Machine Learning*, 2022. URL https://github.com/google/neural-tangents.

Ober, S. W., Rasmussen, C. E., and van der Wilk, M. The promises and pitfalls of deep kernel learning. In *Uncertainty in Artificial Intelligence*, pp. 1206–1216. PMLR, 2021.

Ramachandran, P., Zoph, B., and Le, Q. V. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 2017.

Rasmussen, C. and Williams, C. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA, 1 edition, 2006.

Rezende, D., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Sohl-Dickstein, J., Novak, R., Schoenholz, S. S., and Lee, J. On the infinite width limit of neural networks with a standard parameterization, 2020. URL https://github.com/google/neural-tangents.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial intelligence and statistics*. PMLR, 2009.

Tran, G.-L., Milios, D., Michiardi, P., and Filippone, M. Sparse within sparse gaussian processes using neighbor information. In *International Conference on Machine Learning*, pp. 10369–10378. PMLR, 2021.

Villacampa-Calvo, C., Zaldívar, B., Garrido-Merchán, E. C., and Hernández-Lobato, D. Multi-class gaussian process classification with noisy inputs. *The Journal of Machine Learning Research*, 22(1):1696–1747, 2021.

Wilson, A. and Nickisch, H. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International conference on machine learning*, pp. 1775–1784. PMLR, 2015.

Wilson, A., Hu, Z., Salakhutdinov, R., and Xing, E. Deep kernel learning. In *Artificial intelligence and statistics*, 2016a.

Wilson, A., Hu, Z., Salakhutdinov, R., and Xing, E. Stochastic variational deep kernel learning. *Advances in Neural Information Processing Systems*, 2016b.

Wilson, A. G. and Adams, R. P. Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning (ICML)*, pp. 1067–1075, 2014.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 2016.

A. Understanding of the AVDKL Architecture

We can summarize the AVDKL architecture in three modules: the feature extractor, the variational module, composed by a saturation module and a projection layer, and the sparse GP. The feature extractor is the deep neural network responsible for encoding $\boldsymbol{x}_i$ into $\boldsymbol{g}_i$. The variational module is responsible for saturating $\boldsymbol{g}_i$ and projecting the inducing locations $\boldsymbol{Z}_i$, $\boldsymbol{m}_i$ and $\boldsymbol{L}_i$. Finally, the sparse GP accommodates the current inducing locations and computes the posterior distribution based on $\boldsymbol{g}_i$. An schematic representation of the AVDKL is depicted in Figure 10.

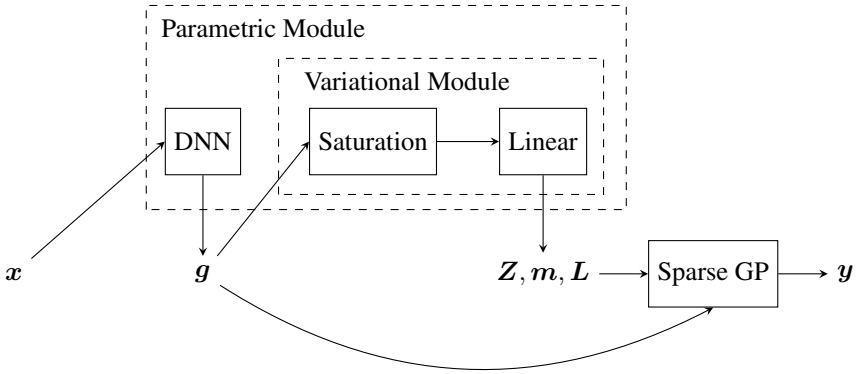


Figure 10. Schematic representation of the AVDKL architecture. The DNN is a deep neural network representing the feature extractor module. In the above figure, the inputs $\boldsymbol{x}$ are projected into $\boldsymbol{g}$ through the feature extractor (DNN); in the variational module, $\boldsymbol{g}$ is saturated and projected into the inducing locations, $\boldsymbol{Z}, \boldsymbol{m}, \boldsymbol{L}$; the inducing locations are fed into the sparse GP module, which receives as input $\boldsymbol{g}$, the representation learning of $\boldsymbol{x}$.

A.1. A Note About the Saturation Module

The saturation module is of great importance for the AVDKL model. As shown in Figure 3, without the saturation module the AVDKL would make bad extrapolation with poor uncertainty estimates.

Such module compose the first operation of the variational module and, as shown in our experiments, one can just apply a Sigmoid/Tanh activation or design more complex saturation modules. As an example, in the image classification scenario we designed a saturation module composed by a linear layer followed by a Sigmoid activation. The linear layer was responsible for compressing $\boldsymbol{g}_i \in \mathbb{R}^E$ into $E/8$ dimensions. In our experiments, such design greatly improved the posterior estimation, giving better uncertainty estimates and better calibration.

In practice, the saturation module comes as a straightforward solution for the absence of a well defined prior distribution for the inputs. During the learning phase, with the right saturation module, the model is capable of learning to bound the representation learning so that the estimations of the inducing locations, which depend on the representation learning, are not freely extrapolated. In this sense, in a situation where the representation learning falls outside the prior knowledge, it will saturate and, therefore, the estimation of the inducing locations will fall in a region where the model is capable of generalizing well.

B. Extra Implementation Details

All models were implemented using PyTorch, PyTorch Geometric (Fey & Lenssen, 2019), GPyTorch (Gardner et al., 2018) and Neural Tangents (Novak et al., 2020; 2022; Han et al., 2022; Hron et al., 2020; Sohl-Dickstein et al., 2020), depending on the evaluation scenario. For instance, PyTorch Geometric was only used on the semi-supervised node classification scenario, while the Neural Tangents was used to define the infinite-width network for the GDKL model. Furthermore, all models were trained on a NVIDIA GeForce RTX 3060 with 12GB, and 16GB of RAM. A summary about the tabular and graph datasets are presented in Tables 4 and 5.

Classification	*N*	*mini-batch*	*D*	*# Tasks*
MagicGamma	15216	256	10	2
HTRU2	14318	256	8	2
Letter	16000	256	16	26
Regression				
Protein	36584	256	12	1
KeggD	42730	512	12	1
KeggU	51686	512	12	1

Table 4. Tabular datasets summary.

Datasets	*# Nodes*	*# Edges*	*# Features*	*# Classes*
Cora	2,708	5,429	1,433	7
CiteSeer	3,327	4,732	3,703	6
PubMed	19,717	44,338	500	3

Table 5. Semi-supervised node classification datasets summary.

Dataset	size/depth	dropout	lr	weight decay
Cora	64/64	0.6	5e-3	5e-4
CiteSeer	256/32	0.6	5e-3	5e-4
PubMed	256/32	0.5	1e-2	5e-4

Table 6. DeepGCN architecture and training settings.

B.1. UCI Classification and Regression

We opted for keeping the NN architectures small. We used 2-hidden layer network with sizes [64, D], so that the latent representation of the inputs $\boldsymbol{g}_i$ is of same size as $\boldsymbol{x}_i$. We trained all NN-based models using AdamW (Loshchilov & Hutter, 2019) with learning rate of 0.005 and weight decay of 0.001 applied only on the network parameters, and the cosine annealing learning rate scheduler (Loshchilov & Hutter, 2017). Concern the AVDKL's inducing locations projection, we used a saturated non-linear activation applied on $\boldsymbol{g}_i$ – Sigmoid for MagicGamma and HTRU2 and Tahn for the remaining datasets – followed by a linear layer. The SVGP model was trained using Adam with a learning rate of 0.01. For the GDKL, an extra step is required during the training phase: train an infinite-width network GP-based model (NNGP) responsible for guiding the DKL optimization process. For this, we defined the infinite-width network with the same architecture of the DKL model. However, we used the GELU activation instead of SiLU, because the neural-tangents library, used for implementing the infinite-width network, only provided the implementation for the GELU activation. The NNGP model was trained in a full-GP regime on 10% inputs sampled from the training set for 300 epochs using the Adam optimizer with a learning rate of 0.01. For classification tasks, the NNGP model was trained with the Dirichlet likelihood with $\alpha_\epsilon = 0.01$ (Milios et al., 2018).

B.2. Semi-Supervised Node Classification

For both AVDKL and SVDKL we used the architecture described by Chen et al. (2022) (see Table 6). For the AVDKL's inducing locations projection, we used the Tanh activation followed by a linear layer. The models were optimized using Adam (Kingma & Ba, 2015) with learning rate and weight decay as in Table 6.

B.3. Image Classification

For both AVDKL and SVDKL we used the ResNet-18 architecture as the feature extractor in order to compute $\boldsymbol{g}_i$, so that the dimension of $\boldsymbol{g}_i$ is of size $E = 512$. For the AVDKL we built a saturation module that first compress $\boldsymbol{g}_i$ into $E/8$ through a linear layer and then applies a Sigmoid non-linear activation on the compressed latent representation. The saturated

Optimization	*Values*
Optimizer	SGD
Initial lr.	0.1
Initial kernel output/length-scale lr.	0.001
Weight decay	0.0001
Warmup epochs	5
Scheduler	cosine annealing
Epochs	300
Data Transforms	-
RandomCrop	size=32, padding=4
HorizontalFlip	p=0.5
Augmentation	trivial augment wide
RandomErasing	p=0.2

Table 7. Details of the recipe used for training the models for image classification on CIFAR10 and CIFAR100.

Model	MagicGamma		HTRU2		Letter	
	Acc. (%)	ECE	Acc. (%)	ECE	Acc. (%)	ECE
AVDKL	88.12 ± 0.64	1.25 ± 0.35	97.97 ± 0.15	$\mathbf{0.35 \pm 0.07}$	$\mathbf{97.42 \pm 0.17}$	$\mathbf{0.61 \pm 0.12}$
GDKL	$\mathbf{88.39 \pm 0.61}$	$\mathbf{1.09 \pm 0.37}$	97.98 ± 0.25	0.55 ± 0.20	96.91 ± 0.10	1.40 ± 0.24
DLVKL	87.73 ± 068	1.27 ± 0.16	98.01 ± 0.13	0.64 ± 0.04	94.09 ± 0.38	4.92 ± 0.29
SVDKL	87.79 ± 0.60	1.59 ± 0.46	$\mathbf{98.07 \pm 0.14}$	0.49 ± 0.08	94.17 ± 0.27	0.92 ± 0.32
SVGP	87.43 ± 0.60	1.05 ± 0.46	98.04 ± 0.11	0.45 ± 0.10	95.63 ± 0.27	5.25 ± 0.24

Table 8. UCI experiments: accuracy and ECE for the classification datasets.

compressed latent representation is fed to a linear layer that computes the inducing locations. Both AVDKL and SVDKL are compared against a standalone ResNet-18. The training recipe for all models is described in Table 7.

There is a crucial difference between the AVDKL and SVDKL architectures: in the AVDKL the latent representation $\boldsymbol{g}_i$ is fed to the GP as is, while the SVDKL normalizes $\boldsymbol{g}_i$ by putting it in a grid, since the inducing locations are also in a grid – note that the values that limit such grid are set manually. In the AVDKL, we have noticed in our first investigations that, due to the high dimensionality of $\boldsymbol{g}_i$ – recall that $E = 512$ –, initializing the kernel with small lengthscales was harmful to the model's convergence. As a solution for this issue, we smoothed the kernel computation by initializing the lengthscales with $l = 15$.

C. Extra Experimental Results

C.1. UCI Classification and Regression

We show in Table 8 the accuracy and ECE for the classification datasets. In Table 9, we show the RMSE for the regression datasets. Finally, in Table 10, we show the training time of one of the independent runs in terms of seconds per epoch.

In Table 8, the accuracy obtained for both MagicGamma and HTRU2 were very similar, except for the SVGP model, which obtained a much lower accuracy when compared to the GDKL and the AVDKL. On the other hand, the AVDKL was superior in the Letter dataset – note that the AVDKL's mean accuracy is far from the GDKL mean accuracy, by more than 1 standard deviation. Furthermore, the AVDKL achieved better calibration on HTRU2 and Letter. In Table 9, the AVDKL achieved a much better RMSE compared to its counterparts. Finally, in Table 10, while the SVDKL was fast in the classification datasets, the AVDKL was faster in the regression datasets – this is expected, since in the regression datasets the SVDKL is not based on the grid approach, having much more inducing points when compared to the AVDKL.

Model	Protein	KeggD	KeggU
AVDKL	**3.7212 ± 0.0530**	**0.0242 ± 0.0004**	**0.0108 ± 0.0017**
GDKL	4.0799 ± 0.0584	0.0334 ± 0.0004	-
DLVKL	4.2869 ± 0.0266	0.0356 ± 0.0007	0.0151 ± 0.0004
SVDKL	4.1588 ± 0.0470	0.0357 ± 0.0006	0.0164 ± 0.0006
SVGP	4.2027 ± 0.0388	0.0364 ± 0.0005	0.0191 ± 0.0006

Table 9. UCI experiments: RMSE for the regression datsets.

Model	MagicGamma	HTRU2	Letter	Protein	KeggD	KeggU
AVDKL	1.323 ± 0.059	1.391 ± 0.119	1.430 ± 0.120	**3.110 ± 0.193**	**1.962 ± 0.109**	**2.386 ± 0.143**
GDKL	4.135 ± 0.089	3.840 ± 0.073	4.631 ± 0.049	7.822 ± 0.221	5.024 ± 0.155	-
DLVKL	1.786 ± 0.116	1.901 ± 0.183	2.172 ± 0.075	4.118 ± 0.238	2.709 ± 0.044	3.313 ± 0.114
SVDKL	**1.036 ± 0.031**	**0.970 ± 0.035**	**1.158 ± 0.055**	3.412 ± 0.033	2.383 ± 0.045	2.873 ± 0.043
SVGP	1.551 ± 0.096	1.351 ± 0.028	1.956 ± 0.082	3.339 ± 0.032	2.315 ± 0.035	2.794 ± 0.043

Table 10. UCI experiments: training time in terms of seconds per epoch.

Model	Cora	CiteSeer	PubMed
AVDKL	30.97 ± 1.20	19.09 ± 0.93	7.58 ± 0.76
SVDKL	**6.99 ± 2.55**	**5.95 ± 1.18**	5.29 ± 1.11
DeepGCN	32.92 ± 0.80	22.00 ± 1.44	**4.52 ± 0.76**

Table 11. Semi-supervised node classification: calibration results in terms of ECE (%) for each dataset.

Model	Cora		CiteSeer		PubMed	
	# Parameters	sec./epoch	# Parameters	sec./epoch	# Parameters	sec./epoch
AVDKL	102K	0.086 ± 0.014	1.1M	0.106 ± 0.013	263K	0.226 ± 0.010
SVDKL	255K	0.089 ± 0.018	1.6M	**0.068 ± 0.029**	782K	0.229 ± 0.010
DeepGCN	**92K**	**0.053 ± 0.017**	**949K**	0.075 ± 0.039	**129K**	**0.206 ± 0.033**

Table 12. Semi-supervised node classification: number of trainable parameters and training time terms of seconds per epoch across all independent runs for each model.

C.2. Semi-Supervised Node Classification

In Table 11 we report the ECE for each model on each graph dataset. In Table 12 we show the model size in terms of number of parameters and the training time in seconds per epochs.

Looking at the Table 11, one can see that the SVDKL obtained a better ECE on the Cora and CiteSeer datasets, but with poorer accuracy values (see Table 2). Also, excepting for PubMed, the AVDKL obtained a better calibration when compared to the DeepGCN. Concern the size of the models an the training time, note in Table 12 that the AVDKL model was always smaller in terms of number of parameters when compared to the SVDKL model, but with similar training time in the Cora and PubMed datsets.

C.3. Image Classification

We provide an extra experiment for image classification with small datasets subsampled from CIFAR10. We considered two scenarios: 10 images per class em 50 images per class. The results can be seen in Table 13. Additionally, we show in Table 14 the number of trainable parameters for each model and the training time in seconds per epoch per scenario.

As one can see in Table 13, the AVDKL was much superior than the SVDKL in terms of accuracy and, compared to the standalone ResNet-18, the AVDKL achieved better results in all metrics, showing the efficiency of the AVDKL model in

samples/class	Metric	AVDKL	SVDKL	ResNet-18
10	Acc. (%)	**29.89 ± 1.48**	25.75 ± 1.74	29.37 ± 0.63
	ECE (%)	18.44 ± 1.64	**2.67 ± 1.33**	24.99 ± 3.89
	MNLL	2.0815 ± 0.0439	**1.9936 ± 0.0342**	2.3891 ± 0.0590
50	Acc. (%)	**49.11 ± 0.81**	43.80 ± 1.37	48.59 ± 4.04
	ECE (%)	21.21 ± 0.93	**14.62 ± 1.22**	26.27 ± 2.70
	MNLL	**1.6682 ± 0.0210**	1.7191 ± 0.0254	1.9922 ± 0.1755

Table 13. Accuracy, ECE and MNLL computed over 3 independent runs for small dataset scenarios based on the CIFAR10 dataset: 10 images per class and 50 images per class.

Model	CIFAR10		CIFAR100	
	# Parameters	sec./epoch	# Parameters	sec./epoch
AVDKL	11.8M	34.306 ± 1.325	12.6M	46.449 ± 13.635
SVDKL	13.3M	58.869 ± 2.967	13.4M	60.141 ± 2.835
ResNet-18	**11.2M**	**27.115 ± 1.328**	**11.2M**	**27.126 ± 1.307**

Table 14. Image classification: number of trainable parameters and training time terms of seconds per epoch across all independent runs for each model.

very small datasets scenarios.

In Table 14, one can see that the ResNet-18 was the smaller model in terms of number of parameter – which is expected, since the AVDKL adds an extra variational amortization layer and the SVDKL adds a set of variational parameters. However, compared to the SVDKL, the AVDKL was much more scalable in terms of both number of parameters and training time, showing the benefits of the amortized variational inference approach.