Deep Kernel Regression with Finite Learnable Kernels

Chunlin Ji✉ CHUNLIN.JI@KUANG-CHI.ORG
Kuang-Chi Institute of Advanced Technology
Shenzhen, China

Yuhao Fu YUHAO.FU@KUANG-CHI.COM
Kuang-Chi Institute of Advanced Technology
Shenzhen, China
Origin Artificial Intelligence Technology Co.
Shenzhen, China

Editors: Berrin Yanıkoğlu and Wray Buntine

Abstract

In this work, we study kernel regression that integrates with a modern deep neural network (DNN). The DNN projects the input into an embedding space, meanwhile a set of representative points is constructed in this embedding space. We build the regression using a finite set of kernels defined on the embedding space, where the DNN weights, the regression coefficients, and kernel hyperparameters are all learnable. We extend the model by introducing a location attention strategy and the multiple kernel technique. We provide effective ways to obtain representative points. The proposed model can be trained with an end-to-end learning algorithm with simple implementation. Simulation studies show that the proposed deep kernel regression is well scalable to large datasets and comparable to or superior to recent deep kernel models in various regression problems.

Keywords: Deep neural network; Kernel method; Kernel ridge regression; Deep kernel regression; Representative points; Multiple kernels; Location attention

1. Introduction

Deep neural networks have gained tremendous success in various difficult tasks such as image classification (Krizhevsky et al., 2012). However, directly using DNN on regression problems may cause overfitting, therefore requires extra regularization techniques, for example, introducing prior on network weights in Bayesian neural networks (MacKay, 1992; Hernández-Lobato and Adams, 2015). Kernel regression is an alternative approach to build a non-linear relation between the covariates and the response. Kernel models have a solid theoretical background and easy-to-handle regularization techniques, therefore, have been extensively applied in machine learning for classification and regression problems (Scholkopf and Smola, 2001). With the success of both approaches, there is growing interest in combining kernel regression models with flexible deep neural network structures.

There exist several ways to incorporate neural networks or network structures with kernel methods. A straightforward way is to utilize the DNN to extract feature-based representation from input, the kernel function is built on the extracted features, and then

apply the kernel method such as Gaussian processes (GP) or Support Vector Machine (SVM) for the following regression/classification task (Cho and Saul, 2009; Damianou et al., 2011; Wilson and Adams, 2013; Rebai et al., 2015; Wilson et al., 2016). The main idea behind this approach is to combine the flexibility of DNNs, in which the representative feature of the data is extracted completely automatically, with the approximation power of kernel methods, in which a feature map is determined by the chosen kernel. The DNN is helpful in discovering rich structures in data and building more expressive kernel functions for the kernel models. However, in most of the proposed methods, the DNN is trained separately from the kernel models. Therefore, it is difficult to guarantee that the extracted features will help the kernel models in the final regression/classification task.

An alternative approach extends kernel methods with a network structure. For example, the deep Gaussian processes (DGPs) (Damianou and Lawrence, 2013) is a hierarchical composition of GPs that can overcome the limitations of kernel expression in a single-layer GPs. In contrast to highly parameterized DNN models, DGPs learn a representation hierarchy nonparametrically with few hyperparameters to optimize. DGPs and its variants (Mattos et al., 2016; Dai et al., 2016) have been proven to gain superior performance in various regression/classification tasks. However, DGPs have complex posteriors (Salimbeni and Deisenroth, 2017; Aitchison et al., 2021) and require sophisticated variational methods for posterior approximation (Salimbeni and Deisenroth, 2017; Salimbeni et al., 2019).

In this work, we propose deep kernel regression (DKR) to sufficiently utilize the advantage of DNNs to extract the feature from the input data and then build a kernel regression on the extracted features. We allow DNN weights, regression coefficients, and kernel hyperparameters to be learnable. To be scalable for large datasets, we only require a small set of representative points in constructing the kernel functions, rather than involving all the training data in the regression formula. We provide an adaptive sampling strategy to select representative points and further propose a novel strategy to generate representative points using a generative neural network. We introduce a DNN for the coefficients with a location attention strategy to capture data localities and allow the multiple kernel technique to further enrich the expression of the model. Moreover, posterior prediction is preferred instead of only the conditional expectation; thus, we address the variance estimation approach, which enables the proposed model to be fully comparable with GPs or DGPs. The proposed DKR can be trained by the stochastic gradient descent (SGD) algorithm in an end-to-end style. We provide a detailed study to show the effectiveness of the proposed model improvements. We find that the proposed DKR is highly scalable to large datasets without increasing the complexity of the model. Our experiments demonstrate that the DKR achieves comparable or superior model performance compared with recent deep kernel models on various regression problems.

2. Related Works

In order to combine DNN with kernel methods, the most efficient approach is to employ a DNN to map the original space to an embedding space and then apply kernel methods on this embedding space. This is an effective way to take advantage of the capabilities of both techniques to gain a more flexible model. Previous work, such as (Cho and Saul, 2009; Damianou et al., 2011; Rebai et al., 2015; Wilson and Adams, 2013; Wilson et al.,

2016), explored this area thoroughly. Most of them choose GPs as the kernel method, as the tuning parameter in GPs is limited, and GPs provide a posterior estimation rather than a point estimation. However, training the hyperparameters in GPs and weights in the DNN requires additional derivation and is not in a simple end-to-end style.

As an alternative, deep kernel learning (or named multilayer kernel learning) uses kernel functions as an element of a multilayer network structure and concatenates one or more layers of such kernel functions to achieve a highly flexible new kernel function (Zhuang et al., 2011; Dinuzzo, 2011). This deep kernel learning method has a more solid theoretical background (Dinuzzo, 2011). However, training such deep kernel learning models is non-trivial and requires sophistical approximation methods (Salimbeni and Deisenroth, 2017; Salimbeni et al., 2019; Aitchison et al., 2021).

The proposed deep kernel regression follows the first kind of deep kernel methods, in which we utilize a DNN to explore the structure of input data and construct a kernel method on the extracted features. However, we choose kernel ridge regression (KRR) (Murphy, 2012) as the kernel model instead of GPs. One benefit of the kernel ridge regression is that it does not require the deriving of the marginal likelihood as in GPs, thus we can provide a SGD based end-to-end training for both the regression coefficients and the weights of networks, which is more favorable in the era of deep learning.

As is known, GPs require computation of matrix inversion, and the cost becomes cubic in the number of training data. Therefore, a low-rank matrix approximation, such as the Nyström method (Williams and Seeger, 2000), is always required to deal with large-scale problems. In addition, traditional KRR also involves all training data in the regression formula. However, in this work, we select or learn a subset of the given train data as representative points in building the regression model instead of involving the entire dataset. We provide an effective adaptive sampling strategy to select representative points, inspired by the sampling scheme in Ma et al. (2015). Moreover, we propose a novel method to learn representative points using a generative neural network. To our knowledge, this generative strategy has not been explored before.

Furthermore, unlike traditional kernel ridge regression, the coefficient in our deep kernel regression can be implemented by a DNN, which is related to the self-attention strategy (Shaw et al., 2018), that allows the weights of the kernel to capture data localities, and therefore provide more flexibility in building the nonlinear relation between the covariates and response. Previous studies (Lanckriet et al., 2004; Gönen and Alpaydın, 2011) show that using multiple different kernels instead of a single kernel improves the model performance. Previous work on the multiple kernel method (Gönen and Alpaydin, 2008; Moeller et al., 2016) utilizes localized kernels and gating functions to choose different kernels. In this study, we apply the localized multiple kernel technique, allowing multiple kernels in the regression and extending the DNN output to weight them.

3. Background

Let $\Omega \in \mathbb{R}^d$ be an open domain and let the input data (covariates) $X := x_1, ..., x_N \in \Omega$ and the corresponding response $Y := \{y_1, ..., y_N\} \in \mathbb{R}$ be given. Let $\mathcal{H} := \mathcal{H}(\Omega, \mathbb{R})$ be a reproducing kernel Hilbert space of real-valued functions on Ω. In real-world applications, the values y_i, $i = 1, ..., N$ are usually not exactly given, but are observed with some noise

term. In this case, we formulate the problem by the regularized least-squares regression,

$$f^{\lambda}_{X,Y} := \underset{f\in\mathcal{H}}{argmin} \sum_{j=1}^{N} |f(x_i) - y_i|^2 + \lambda \|f\|^2_{\mathcal{H}} \tag{1}$$

where the side condition in Eq.(1) is substituted by a penalty term, where the Lagrange multiplier λ weights the importance of the norm minimization against the function evaluation error. The representer theorem (Schölkopf et al., 2001; Micchelli and Pontil, 2005; Steinwart and Christmann, 2008) provides that $f^{\lambda}_{X,Y}$ is of the form,

$$f^{\lambda}_{X,Y}(x) = \sum_{i=1}^{N} \alpha^{\lambda}_i K(x_i, x) \tag{2}$$

where $K : \Omega \times \Omega \to \mathbb{R}$ denotes the reproducing kernel of $\mathcal{H}$ and $\alpha \in \mathbb{R}$, $i = 1, ..., N$, are the corresponding coefficients. The coefficients α^{λ}_i , $i = 1, ..., N$, can be determined by the linear system $(\boldsymbol{M}_{X,X} + \lambda \boldsymbol{I})\boldsymbol{\alpha}^{\lambda} = \boldsymbol{y}$, where $\boldsymbol{M}_{X,X} := [K(x_i; x_j)]_{i=1:N,j=1:N}$, $\boldsymbol{\alpha} := [\alpha_1, ..., \alpha_N]^T$ and $\boldsymbol{y} := [y_1, ..., y_N]$, $\boldsymbol{I}$ denotes the $N \times N$ identity matrix.

Any Mercer kernel (continuous, symmetric, positive definite) K may be used as the reproducing kernel of $\mathcal{H}$. Here we list the common choices: the radial basis function (RBF) kernel $K_{\text{RBF}}(x, x^*) = \exp\left(-\frac{1}{2}(x - x^*)^T\Theta^{-2}(x - x^*)\right)$ with lengthscale parameter $\Theta \in \mathbb{R}^d$; the linear kernel $K_{\text{Linear}}(x, x^*) = \Theta x^T x^*$ with variance parameter $\Theta \in \mathbb{R}^d$; the Polynomial kernel $K_{\text{Poly}}(x, x^*) = (x^T x^* + c)^r$, with offset parameter $c \geq 0$ and power $r \geq 1$; the rational quadratic (RQ) kernel $K_{\text{RQ}}(x, ^*) = \left(1 + \frac{1}{2\alpha}(x - x^*)^T\Theta^{-2}(x - x^*)\right)^{-\alpha}$; and the Matern kernel $K_{\text{Matern}}(x, x^*) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}d\right)^{\nu} K_{\nu}\left(\sqrt{2\nu}d\right)$, where $d = (x - x^*)^T\Theta^{-2}(x - x^*)$, ν is a smoothness parameter, $K_v(\cdot)$ is a modified Bessel function and $\Gamma(\cdot)$ is the gamma function. These kernels measure the similarity between two data points $x, x^* \in \mathbb{R}^d$, accounting for possible differences in significance of the various dimensions.

4. Deep Kernel Regression

The propose deep kernel regression, we use a DNN $\phi(\cdot)$ to project the input data x into a low-dimensional embedding space that $\phi(x) : \mathbb{R}^d \to \mathbb{R}^p$, where $p < d$. We expect that the DNN $\phi(\cdot)$ can extract 'meaningful' feature which could benefit the kernel regression. To reduce computation efforts, here we use only a representative subset of the entire dataset, $\{x^*_j\}^J_{j=1}$, to build the regression model. We name these subset samples as *representative points* and will provide the selection or learning strategy for this subset in a later section. Given the representative data points and the embedding network $\phi(\cdot)$, we can evaluate the kernel $K(\phi(x), \phi(x^*_j))$ ($j = 1, ...J$) with respect to all the representative points. The deep kernel regression is defined as

$$\mu_f(x) = \sum_{j=1}^{J} a_j K(\phi(x), \phi(x^*_j)) \tag{3}$$

where a_j is the weight of each kernel $K(\cdot, \cdot)$, and $K(\cdot, \cdot)$ is a Mercer kernel.

4.0.1. Location Attention

Inspired by the spatial attention strategy in DNNs (Shaw et al., 2018), we let the weight a_j depend on the input x_i to count the localization effort. We introduce a DNN for the coefficients to realize the localization attention strategy. To be specified, the regression coefficient network $\alpha(\cdot) : \mathbb{R}^d \to \mathbb{R}^J$ projects the input data x into the 'weight' for each $K(\phi(x), \phi(x_j^*))$ ($j = 1, ...J$), then we define the kernel regression as,

$$\mu_f(x) = \sum_{j=1}^{J} \alpha^{(j)}(x) K(\phi(x), \phi(x_j^*)) \tag{4}$$

where $\alpha^{(j)}(x)$ represents the j-dimension of the network output. As shown in Fig. 1, the output of the coefficient network $\alpha(\cdot)$ provides the attention scheme: the contribution of each term $K(\phi(x), \phi(x_j^*))$ (for $j = 1, ...J$) now depends on the 'location' of x.

4.0.2. Multiple kernel technique

The performance of kernel regression always depends on the choice of kernel functions, as some kernels may better match the data structure than others. Using the multiple kernel technique is an easy way to avoid choosing a kernel for each individual problem. Assuming that the data have underlying localities, we should give higher weights to appropriate kernel functions (i.e., kernels that match the complexity of data distribution) for each local region. The weights for difference kernels are generally designed to be localized, which means that the weights depend on the input variable. Therefore, we introduce another coefficient network $\eta(\cdot)$ to project the representative point x_j^* into a coefficient vector for each type of kernel function M, $\eta(\cdot) : \mathbb{R}^d \to \mathbb{R}^M$. To guarantee the weight $\sum_m \eta^{(m)}(\cdot) = 1$, we use the softmax function for the output of $\eta(\cdot)$, that $\eta^{(m)}(\cdot) \leftarrow \frac{e^{\widetilde{\eta}^{(m)}(\cdot)}}{\sum_{m'} e^{\widetilde{\eta}^{(m')}(\cdot)}}$, where $\widetilde{\eta}^{(m)}$ represents the output of the last layer in the network η. Then the deep kernel regression becomes

$$\mu_f(x) = \sum_{j=1}^{J} \alpha^{(j)}(x) \sum_{m=1}^{M} \eta^{(m)}(x_j^*) K_m(\phi(x), \phi(x_j^*)) \tag{5}$$

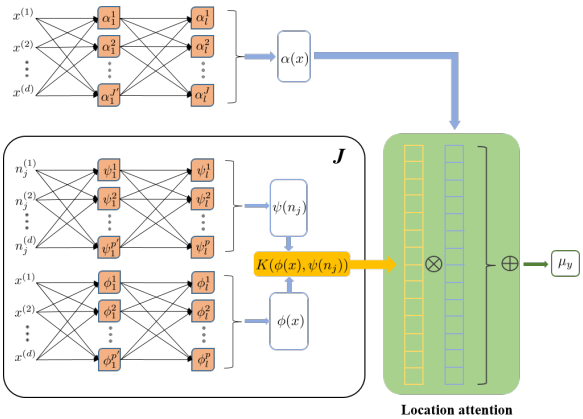


Figure 1: Illustration of the deep kernel regression model.

4.1. Selecting or Learning of Representative Points

Traditionally, to build the regression model, the kernel methods, such as KRR and GPs, use all the data from the training set. As a consequence, it may take heavy computation effort, particularly when the dataset becomes large. To reduce computation efforts and model complexity, we propose to use a small set of representative points to build the DKR. Here, we provide two simple but effective approaches to obtain representative points: 1) an effective adaptive sampling strategy for selecting the representative point from the given training data; 2) a generative approach that leverages a generative DNN to generate the set of representative points from unscented noise.

4.1.1. Adaptive Sampling

Following the adaptive sampling scheme in Ma et al. (2015), we first divide the range of responses $\{y_i\}_1^N$ into Γ disjoint intervals, S_1,..., S_Γ and let $|S_\gamma|$ denote the number of observations in S_γ; then for each S_γ form the collection of all pairs (x_i, y_i) where $y_i \in S_\gamma$ and draw a subsample of size n_γ (J/Γ) from this collection through uniform sampling of the sample index, denote the samples by $x^{*(\gamma)} = \{x_1^{*(\gamma)}, ..., x_{n_\gamma}^{*(\gamma)}\}$; Combine $\{x^{*(1)}, ..., x^{*(\Gamma)}\}$ to form the representative points set. The size of this set is $\sum_{\gamma=1}^{\Gamma} = J$.

4.1.2. Generative Method

Following the idea of GAN (Goodfellow et al., 2014), we generate a set of data that is distributed the same as the given dataset. We propose to learn representative points using a generative neural network from inputs of white noise. Unlike a GAN model, here we drop the discriminative network, and only keep the generative network, and let the regression loss functions guide the learning of the generative network. We define the generative network, $\psi(\cdot) : \mathbb{R}^d \to \mathbb{R}^p$, which transfers non-informative data, such as a uniformly distributed random variable, into a set of representative data points. With this generative method, the DKR defined in Eq.(5) is reformated as,

$$\mu_f(x) = \sum_{j=1}^{J} \alpha^{(j)}(x) \sum_{m=1}^{M} \eta^{(m)}(\psi(n_j)) K_m(\phi(x), \psi(n_j)) \tag{6}$$

where n_j is an uninformative random sample in which each dimension of n_j is sampled from a uniform distribution, $n_j^{(d_i)} \sim U(0, 1)$.

The DKR with the generative representative points defined by Eq.(6) is illustrated in Fig. 1. Without bothering with the selection of representative points, this generative method makes the model more concrete; simulation study will show that it always gains superior performance than the adaptive sampling strategy.

4.2. Covariance Estimation

Given the training pair (x_i, y_i) $(i = 1, ..., N)$, we can train the parameters in the DKR model with MSE loss, that

$$\mathcal{L}_{\text{MSE}} = \sum_{i=1}^{N} |y_i - \mu_f(x_i)|^2. \tag{7}$$

However, when the test point x' is given, the model can only predict the conditional expectation $\mu_f(x')$, which is only a point estimation. In comparison, GPs and DGPs models can output both the mean and the covariance of the prediction to calibrate a posterior.

To obtain the variance estimation, here we introduce another branch of the model to estimate the covariance explicitly as follows,

$$\sigma_f(x) = \sum_{j=1}^{J} \beta^{(j)}(x) \sum_{m=1}^{M} \eta^{(m)}(x_j^*) K_m(\phi(x), \phi(x_j^*)) \tag{8}$$

where we introduce another coefficient network $\beta(\cdot) : \mathbb{R}^d \to \mathbb{R}_+^J$, note that we need an activation function, such as Softplus(), to enable the output of $\beta(\cdot)$ be positive. Moreover, we share the network $\phi(\cdot)$, $\eta(\cdot)$ and $\psi(\cdot)$ between the expression of $\mu_f(x)$ and $\sigma_f(x)$. If representative points are obtained by the generative method, we only need to change $\phi(x_j^*)$ in Eq.(8) to $\psi(n_j)$. With the variance $\sigma_f(x)$, we define the loss function as the negative log-likelihood instead of the MSE loss, that is,

$$\mathcal{L}_{\text{NLL}} = -\sum_{i=1}^{N} \log \mathcal{N}(y_i | \mu_f(x_i), \sigma_f^2(x_i)) \tag{9}$$

As shown in previous work (Detlefsen et al., 2019), estimation of a variance network is non-trivial: the maximum likelihood estimate (MLE) of σ_f^2 does not exist when only a single observation is available and μ_f is unknown, as the mean is not a free parameter. Therefore, if we optimize μ_f and σ_f^2 jointly (even with a warm-up of μ_f), Detlefsen et al. (2019) suggests that the joint update of mean and covariance is substandard as σ_f^2 is never sufficiently optimized when μ_f is unknown. Fortunately, Detlefsen et al. (2019) addresses a mean-variance split training strategy: sequentially optimizing μ_f by assuming σ_f^2 is known, and optimizing σ_f^2 by assuming μ_f is known. Moreover, an adjusted stochastic gradient for unbiased estimation is also suggested (Detlefsen et al., 2019); however, it requires heavy upfront computation and thus is not suitable for the DKR model to handle large datasets.

4.3. Learning and Inference

4.3.1. Regularization

To prevent model overfitting, traditional kernel models introduce regularization terms on the norm of the regression function with a Lagrange parameter to control this regularization. For the DKR with vector coefficients Eq.(3), we can estimation of norm by $\|\mu_f\|_{\mathcal{H}}^2 = \langle \sum_{i=1}^{J} \alpha^{(i)} K(\cdot, \phi(x_i^*)), \sum_{j=1}^{J} \alpha^{(j)} K(\cdot, \phi(x_j^*)) \rangle = \sum_{i=1}^{J} \sum_{j=1}^{J} \alpha^{(i)} \alpha^{(j)} \langle K(\cdot, \phi(x_i^*)), K(\cdot, \phi(x_j^*)) \rangle$ $= \sum_{i=1}^{J} \sum_{j=1}^{J} \alpha^{(i)} \alpha^{(j)} K(\phi(x_i^*), \phi(x_j^*))$, where $\langle \cdot, \cdot \rangle$ denotes the inner product. Furthermore, in case the multiple kernel technique is applied, the norm becomes $\|\mu_f\|_{\mathcal{H}}^2 = \sum_{i=1}^{J} \sum_{j=1}^{J} \alpha^{(i)} \alpha^{(j)}$ $\widetilde{K}(\phi(x_i^*), \phi(x_j^*))$ where $\widetilde{K}(\phi(x_i^*), \phi(x_j^*)) = \sum_m \eta^{(m)}(x_i^*) K_m(\phi(x_i^*), \phi(x_j^*)) \eta^{(m)}(x_j^*)$. If representative points are obtained by the generative method, we need to change $\phi(x_j^*)$ to $\psi(n_j)$ to estimate the norm. Similarly, we obtain $\|\sigma_f\|_{\mathcal{H}}^2$ for the variance function. Accordingly, the entire loss function can be expressed as follows,

$$\mathcal{L} = -\sum_{i=1}^{N} \log \mathcal{N}(y_i | \mu_f(x_i), \sigma_f^2(x_i)) + \lambda \|\mu_f\|_{\mathcal{H}}^2 + \lambda \|\sigma_f\|_{\mathcal{H}}^2 \tag{10}$$

However, the DKR with location attention does not have a closed-form solution for the norm. Given the model expression in Eqn (4), we have $\|\mu_f\|^2_{\mathcal{H}} = \langle \sum_{i=1}^{J} \alpha^{(i)}(x) K(\cdot, \phi(x_i^*)), \sum_{j=1}^{J} \alpha^{(j)}(x) K(\cdot, \phi(x_j^*)) \rangle = \sum_{i,j=1}^{J} \alpha^{(i)}(x)\alpha^{(j)}(x) K(\phi(x_i^*), \phi(x_j^*))$. Thus, the norm $||\mu_f||^2$ is still a function of input x rather than a conventional scaler value, which can not be served as the term for norm regularization; Therefore, the norm regularization does not apply to this case.

In practice, we find that with the help of norm regularization, the DKR without location attention is preferred for small-scale data, while the DKR with location attention is preferred for large-scale data. Simulation studies support this assumption.

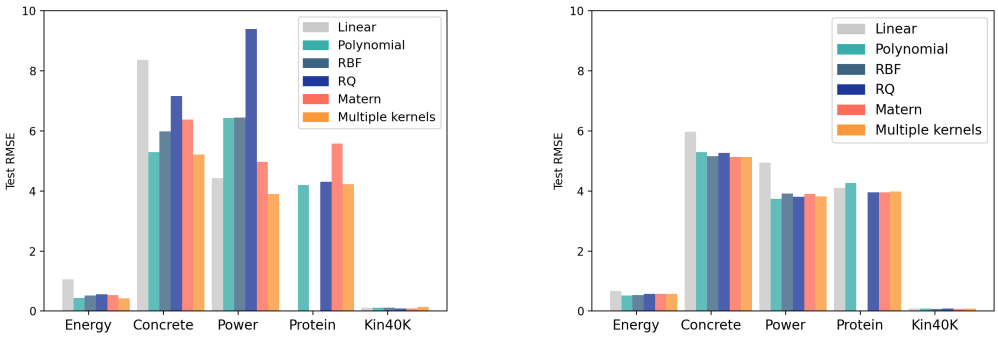
4.3.2. Training and Inference

The learnable parameters in the proposed model are $\phi(\cdot)$, $\psi(\cdot)$, $\eta(\cdot)$, $\alpha(\cdot)$, $\beta(\cdot)$, (or α, β), and hyperparameters in kernel functions (denoted as Θ_K). Note that we can leverage the auto differential package for kernel function, such as GPyTorch library (Gardner et al., 2018), to obtain the gradient of kernel hyperparameters. Given the NLL loss function (Eq.(9) or Eq.(10)), we can train the DNNs and kernel hyperparameters using a SGD algorithm in an end-to-end style. Remember that to apply the mean-variance split training, we group the parameters as $\{\phi(\cdot), \psi(\cdot), \eta(\cdot), \alpha(\cdot), \Theta_K\}$ and $\{\beta(\cdot)\}$, and train them separately assuming that the other group of parameters is fixed. In the inference stage, given the new data x', we can predict the conditional mean and variance using the trained DKR model.

5. Simulation Study

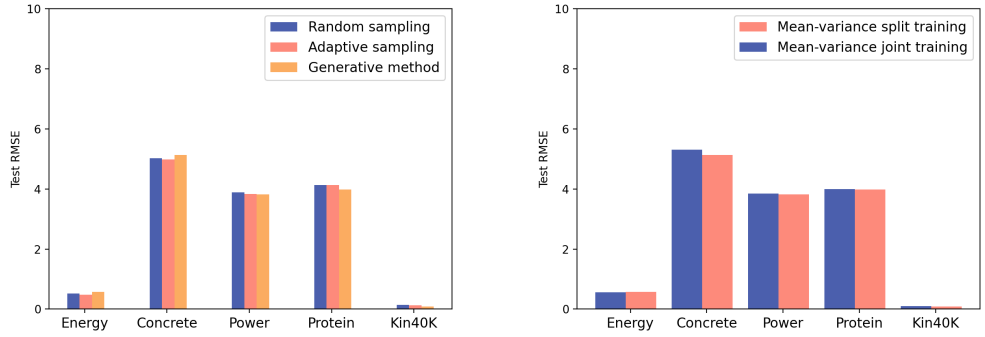
To test our methodologies, we conduct multiple experiments in various settings. All datasets studied in this work are real public data from the UCI Machine Learning Repository (Asuncion, 2007). We first provide a thorough study to show the effectiveness of the proposed improvements, including location attention strategy, kernel choice, learning of representative point and mean-variance split training. Then, we investigate the model performance on a batch of small to medium-scale UCI datasets. Various regression methods (Titsias, 2009; Bui et al., 2016; Salimbeni and Deisenroth, 2017) have been studied on these datasets, thus we can compare the performance with the state-of-the-art ones. To show that the proposed method is scalable to large datasets, we evaluate the model performance on a set of large-scale UCI datasets, compared with several advanced kernel models.

The neural network $\phi(\cdot)$, $\psi(\cdot)$, $\alpha(\cdot)$, $\beta(\cdot)$ and $\eta(\cdot)$ in the proposed models are chosen as Multi-Layer Perception (MLP). All network structures are detailed in the Appendix. We implement five kernels for the regression model: linear kernel, polynomial kernel($q = 2$), RBF kernel, RQ kernel and Matern kernel. The kernels are implemented by the GPyTorch library (Gardner et al., 2018). The end-to-end training algorithm is implemented in Pytorch(Paszke et al., 2017). The network weights and kernel hyperparameters are jointly optimized by the Adam algorithm (Kingma and Ba, 2015). We choose a batch size of 512 for small-scale dataset (N <2,000) and a batch size of 1,024 for other larger dataset. We set the learning rate to lr =2e-3 for networks, $\phi(\cdot)$, $\psi(\cdot)$, $\eta(\cdot)$, $\alpha(\cdot)$, and lr =0.5e-3 for the network $\beta(\cdot)$. We run 500 epochs for small-scale dataset (N <2,000), 200 epochs for medium-scale dataset and 20 epochs for large-scale dataset ($N > 1e5$). Following common practice, each dataset is randomly split into 80% training, 10% validating, and 10% held out



(a) DKR without location attention(**Case 1**). (b) DKR with location attention(**Case 2**).

Figure 2: RMSE of DKR with different kernels.



(a) DKR with different approaches to obtain the representative points (**Case 3**).

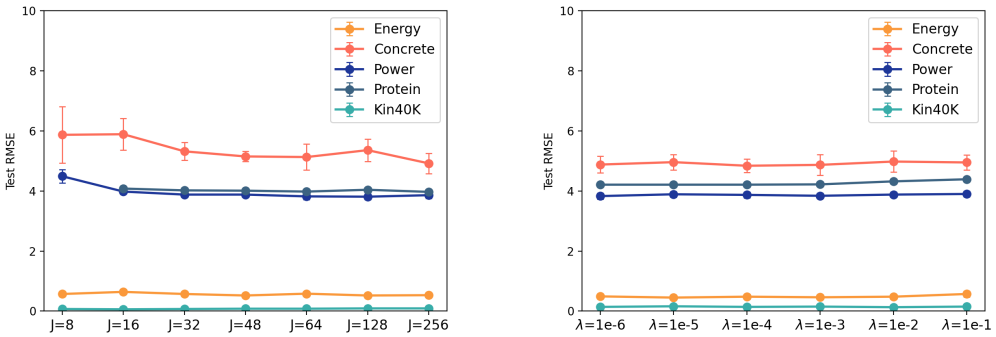
(b) DKR with mean-variance joint training vs split training (**Case 4**).

Figure 3: RMSE of DKR with different representative points and training methods.

test set. We scale the inputs and outputs to zero mean and unit standard deviation within the training set (we restore the output scaling for evaluation). There remain two important hyperparameters: the number of representative points and the weight λ for regularization terms in Eq.(10), of which the default values are $J = 64$ and λ =1e-4. We discuss the choice of J and λ in the ablation study. In all experiments, we report both the root mean square error (RMSE) and logarithmic likelihood (LL) as the metric of model performance.

5.1. Ablation Study

We investigate the following cases to verify the proposed improvements: **Case 1)**, since the choice of kernel always plays an important role in kernel regression, we compare the performance of the DKR model with five different kernels, including linear, polynomial, RBF, RQ and Matern. Additionally, for the multiple kernel model, we use polynomial, RQ and Matern. Notice that in this case study, we do not use the location attention strategy; **Case 2)**, one important contribution of the proposed work is the introduction of the location attention strategy; To demonstrate the effects of this improvement, we use the DKR model with location attention to carry out the same experiment as in Case 1; **Case 3)**, we compare



(*a*) DKR with a different number of representative points (**Case 5**).

(*b*) DKR training with different weights for the norm regularization (**Case 6**).

Figure 4: RMSE of DKR with important hyperparameters.

three ways to obtain representative points: random sampling (randomly select 64 training data as representative points), adaptive sampling and the generative method introduced before. In this case study, we use the DKR with location attention and multiple kernels (polynomial, RQ and Matern); **Case 4)**, we compare the two variance estimation methods, that mean-variance joint training vs. split training. Here, we also use the DKR with location attention and multiple kernels; **Case 5)**, we compare the model performance with different $J = [8, 16, 32, 48, 64, 128, 256]$. Here, we also use the DKR with location attention and multiple kernels; **Case 6)**, we compare the model performance when learning the DKR (without location attention) with different $\lambda =$ [1e-6,1e-5,1e-4,1e-3,1e-2,1e-1]. Here, we also use the DKR with multiple kernels.

For all case studies, we evaluated the model performance on five datasets: Energy, Concrete, Power, Protein, Kin40K. For each dataset, we run five trials, and in each trial randomly select the validation/test datasets, and train the model from random initiation.

5.1.1. Results

Plots of RMSEs are shown in Fig.2-4. As shown in Fig.2(*a*)subfigure and Fig.2(*b*)subfigure, the DKR with a single kernel can obtain better performance on some datasets, as the kernel may fit the inner structure of the data distribution; however, the DKR with multiple kernels always has a more robust and favorable performance over all datasets. According to Fig. 3(*a*)subfigure, the DKR with the generative method obtains the best performance and the adaptive sampling method is slightly better than the random sampling method. According to Fig. 3(*b*)subfigure, the DKR with mean-variance split training obtains slightly better performance than mean-variance joint training. As shown in Fig. 4(*a*)subfigure, the DKR with medium-size representative points, for example $J = 64$, performs well and is scalable to large datasets. Fig. 4(*b*)subfigure shows that our model is robust against the weight of norm regularization, $\lambda =$1e-3 is applicable for practical problems.

5.2. Regression Benchmarks

We compare our DKR model with other regression methods on 8 standard small/medium-sized UCI benchmark datasets, as their state-of-the-art results of RMSE and LL are broadly known. The model for comparison includes, a Bayesian neural network with inference by probabilistic backpropagation (Hernández-Lobato and Adams, 2015) (PBP), GP models: Sparse Gaussian process regression (SGP) (Titsias, 2009) with 500 inducing points, here denoted as SGP1, and deep GP models: DGP with 1 hidden layer using approximate expectation propagation (Bui et al., 2016) (AEP-DGP), and DGP with 2 layers and 5 layers (100 inducing points) using doubly stochastic variational inference (Salimbeni and Deisenroth, 2017), here denoted as DSVI-DGP1 and DSVI-DGP2. Here, we directly cite their results on the benchmark datasets (Salimbeni and Deisenroth, 2017).

According to the ablation study, here we set two models as follows: **DKR1**–DKR with multiple kernels (polynomial, RQ and Matern) but without location attention, **DKR2**–DKR with multiple kernels (the same as DKR1) and location attention. For both models, use the generative method to gain representative points and the mean-variance split training strategy. We set $J = 64$ and choose λ =1e-3 for training **DKR1**. All trials are averaged over 10 trials with different splits.

5.2.1. Results

The test log-likelihood and RMSE are shown in Table 1 and 2 respectively. For both log-likelihood and RMSE, our performance is comparable to or even superior to these state-of-the-art methods, such as SGPs and DSVI-DGPs. Moreover, we observe that with the help of norm regularization, **DKR1** performs better on small-scale datasets, while **DKR2** is favorable for larger datasets.

Table 1: Regression test log-likelihood for small/medium scale datasets. Reported is the mean over 10 splits (with standard errors)

Dataset	PBP	SGP2	AEP-DGP	DSVI-DGP1	DSVI-DGP2	DKR1	DKR2
Boston	-2.57(0.09)	**-2.40(0.07)**	***-2.46(0.09)***	-2.47(0.05)	-2.49(0.05)	**-2.40(0.05)**	-2.47(0.06)
Energy	-2.04(0.02)	***-0.63(0.03)***	-1.65(0.15)	-0.73(0.02)	-0.74(0.02)	**-0.46(0.04)**	-0.68(0.06)
Concrete	-3.16(0.02)	-3.09(0.02)	-3.30(0.12)	-3.12(0.01)	-3.13(0.01)	**-2.98(0.12)**	***-3.03(0.05)***
Kin8nm	0.90(0.01)	1.15(0.00)	1.15(0.03)	***1.34(0.01)***	**1.38(0.01)**	1.24(0.01)	1.23(0.01)
Power	-2.84(0.01)	***-2.75(0.01)***	-2.78(0.01)	***-2.75(0.01)***	**-2.73(0.01)**	-2.83(0.04)	***-2.75(0.02)***
Naval	3.73(0.01)	**7.01(0.05)**	4.37(0.23)	***6.76(0.19)***	6.41(0.28)	5.40(0.05)	4.84(0.02)
Protein	-2.97(0.00)	-2.83(0.00)	-2.81(0.01)	-2.81(0.00)	***-2.71(0.00)***	***-2.71(0.02)***	**-2.64(0.01)**
Wine_red	-0.97(0.01)	**-0.93(0.01)**	-1.51(0.09)	-0.95(0.01)	-0.95(0.01)	***-0.94(0.01)***	**-0.93(0.01)**

Table 2: Regression test RMSE for small/medium scale datasets. Reported is the mean over 10 splits (with standard errors).

Dataset	PBP	SGP2	AEP-DGP	DSVI-DGP1	DSVI-DGP2	DKR1	DKR2
Boston	3.01(0.18)	***2.73(0.12)***	3.42(0.37)	2.90(0.17)	2.92(0.17)	**2.72(0.15)**	3.13(0.19)
Energy	1.80(0.05)	***0.47(0.02)***	1.70(0.42)	***0.47(0.01)***	***0.47(0.01)***	**0.43 (0.03)**	0.57(0.03)
Concrete	5.67(0.09)	5.53(0.12)	7.68(0.90)	5.61(0.10)	5.61(0.10)	**5.11(0.42)**	***5.13(0.40)***
Kin8nm	0.10(0.00)	0.08(0.00)	0.08(0.00)	**0.06(0.00)**	**0.06(0.00)**	0.075(0.00)	***0.074(0.00)***
Power	4.12(0.03)	**3.79(0.03)**	3.99(0.03)	**3.79(0.03)**	**3.79(0.03)**	3.90(0.06)	***3.82(0.10)***
Naval	***0.01(0.00)***	**0.00(0.00)**	**0.00(0.00)**	**0.00(0.00)**	**0.00(0.00)**	**0.00(0.00)**	**0.00(0.00)**
Protein	4.37(0.01)	4.10(0.03)	4.54(0.02)	***4.00(0.03)***	***4.00(0.03)***	4.22(0.06)	**3.99(0.03)**
Wine_red	0.64(0.01)	***0.62(0.01)***	0.64(0.01)	0.63(0.01)	0.63(0.01)	**0.61(0.01)**	***0.62(0.01)***

5.3. Large-Scale Regression

We compare against four scalable GPs: A GP model using the Black-Box Matrix-Matrix Multiplication (BBMM) inference procedure (Exact GP) (Gardner et al., 2018), Sparse Gaussian process regression (SGP) (Titsias, 2009), Stochastic Variational Gaussian Processes (SVGP) (Hensman et al., 2013), and Latent Variables based three layers Gaussian Process (LV-3GP) (Salimbeni et al., 2019). We choose these methods because of their popularity and general applicability. Notice that SGPR and SVGP are two scalable GP approximations, we quote their results with a typical value for the inducing point ($m = 512$ for SGPR and $m = 1024$ for SVGP) (Wang et al., 2019). LV-3GP is an advanced DGPs, which has obtained state-of-the-art results on a wide range of UCI datasets (Salimbeni et al., 2019). We use **DKR1** and **DKR2** addressed in last section. All trials are averaged over five trials with different splits.

Results. The test log-likelihood and RMSE are shown in Table 3 and 4 respectively. Our experiments demonstrate that both **DKR1** and **DKR2** are well scalable for large-scale problems. The DKR2 outperforms popular GPs methods in most benchmarking datasets and achieves several state-of-the-art results. For the RMSE, our proposed models also gain favorable performance. Moreover, we see that the **DKR2** always performs better than **DKR1** on these large-scale datasets.

Table 3: Regression test log-likelihood results for large dataset. Reported is the mean over 5 splits (with standard errors)

Dataset	Exact GP	SGPR	SVGP	LV-3GP	DKR1	DKR2
Bike	-0.119(0.044)	-0.291(0.032)	-0.272(0.018)	**3.95 (0.01)**	3.049(0.052)	***3.510(0.031)***
Kin40K	0.258(0.084)	-0.087(0.067)	-0.236(0.077)	***1.27 (0.00)***	1.222(0.033)	**1.480(0.014)**
KeggDirected	0.199(0.381)	1.123(0.016)	0.940(0.020)	**2.26 (0.01)**	1.635(0.082)	***1.675(0.032)***
CTslice	0.894(0.188)	0.073(0.097)	-1.422(0.005)	***2.02 (0.00)***	1.830(0.046)	**2.074(0.034)**
3DRoad	-0.909(0.001)	-0.943(0.002)	-0.697(0.002)	-0.48 (0.00)	***-0.325(0.027)***	**-0.265(0.022)**
Song	-1.206(0.024)	-1.213(0.003)	-1.417(0.000)	**-1.07 (0.00)**	-1.109(0.002)	***-1.105(0.002)***
Buzz	-0.267(0.028)	-0.106(0.008)	-0.224(0.050)	0.04 (0.00)	***0.081(0.010)***	**0.098(0.005)**
HouseElectric	0.152(0.001)	—	1.010(0.039)	***2.03 (0.00)***	1.938(0.048)	**2.047(0.016)**

Table 4: Regression test RMSE for large scale dataset. Reported is the mean over 5 splits (with standard errors).

Dataset	Exact GP	SGPR	SVGP	LV-3GP	DKR1	DKR2
Bike	0.220(0.002)	0.362(0.004)	0.303(0.004)	—	***0.210(0.072)***	**0.149(0.001)**
Kin40K	***0.099(0.001)***	0.273(0.025)	0.268(0.022)	—	0.141(0.012)	**0.085(0.002)**
KeggDirected	***0.086(0.005)***	0.104(0.003)	0.096(0.001)	—	0.101(0.006)	**0.083(0.002)**
CTslice	0.262(0.448)	0.218(0.011)	1.003(0.005)	—	***0.118(0.023)***	**0.045(0.003)**
3DRoad	**0.110(0.007)**	0.661(0.010)	0.481(0.002)	—	0.500(0.008)	***0.465(0.008)***
Song	0.807(0.024)	0.803(0.002)	0.998(0.000)	—	***0.776(0.001)***	**0.773(0.001)**
Buzz	0.288(0.018)	0.300(0.004)	0.304(0.012)	—	***0.245(0.002)***	**0.240(0.000)**
HouseElectric	0.055(0.000)	—	0.084(0.005)	—	***0.048(0.000)***	**0.045(0.000)**

6. Discussion

Our experiments show that on a wide range of tasks, the proposed model is effective and scalable. We observe that on small to medium scale dataset the models do not overfit, the DKR1 is preferred for this case and obtain superior results. Meanwhile, on large datasets

both DKR1 and DKR2 perform well and DKR2 gains superior performance than other kernel methods. Crucially, we do not need to increase the model complexity to fit large-scale problems. Unlike the DGPs require sophisticated approximate inference, such as doubly stochastic variational inference and importance-weighted variational inference, our models work like a black-box and can be trained easily by a SGD algorithm in an end-to-end style and enable a simple implementation.

We consider the algorithm complexity at the beginning of the algorithm design, and that is why we introduce the representative points instead of involving all training instances/data in the model. The major computation comes from the kernel function evaluation: for single kernel DKR, the computation complexity is $O(N * J)$, where N is the number of training instances and J is the number of representative points; for multiple kernel DKR, computation complexity is $O(N * J * M)$, where M is the number of Kernel types. In our recommended model setting, $J = 64$, and $M = 3$, the computation complexity is smaller than that of conventional kernel methods, which is known as between $O(N^2)$ and $O(N^3)$. Moreover, according to the simulation studies, we have found that the proposed DKR models are highly computation-friendly.

7. Conclusion

In this study, we present a simple but effective way to incorporate the kernel regression with a deep neural network. We propose effective strategies to construct a small finite set of representative points. As a consequence, the proposed DKR models only require a small number of kernels, and thus possess a low model complexity. The proposed improvements, such as location attention strategy, multiple kernel technique, and norm regularization, enable the DKR model to perform more robustly and favorably on a wide range of problems. Furthermore, the proposed DKR models can be implemented and trained in a simple way using the SGD-type algorithm in an end-to-end style. Compared to recent advanced deep kernel models, the proposed DKR models gain superior performance on both log-likelihood and RMSE, particularly for large-scale datasets.

Acknowledgments

This work was supported by National Key R&D Program of China No. 2021YFB3802103.

References

Laurence Aitchison, Adam Yang, and Sebastian W Ober. Deep kernel processes. In *ICML*, pages 130–140. PMLR, 2021.

A. Asuncion. Uci machine learning repository, university of california, irvine. *http://www.ics.uci.edu/ mlearn/MLRepository.html*, 2007.

Thang D. Bui, Daniel Hernández-Lobato, José Miguel Hernández-Lobato, Yingzhen Li, and Richard E. Turner. Deep gaussian processes for regression using approximate expectation propagation. In *ICML*, 2016.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *NeurIPS*, 2009.

Zhenwen Dai, Andreas C Damianou, Javier González, and Neil D Lawrence. Variational auto-encoded deep gaussian processes. In *ICLR*, 2016.

Andreas Damianou and Neil D Lawrence. Deep gaussian processes. In *AISTATS*, pages 207–215. PMLR, 2013.

Andreas C. Damianou, Michalis K. Titsias, and Neil Lawrence. Variational gaussian process dynamical systems. In *NeurIPS*, 2011.

Nicki Skafte Detlefsen, Martin Jørgensen, and Søren Hauberg. Reliable training and estimation of variance networks. In *NeurIPS*, 2019.

Francesco Dinuzzo. *Learning functions with kernel methods*. PhD thesis, University of Pisa Pisa, Italy, 2011.

Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *NeurIPS*, 31, 2018.

Mehmet Gönen and Ethem Alpaydin. Localized multiple kernel learning. In *ICML*, 2008.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

James Hensman, Nicoló Fusi, and Neil Lawrence. Gaussian processes for big data. *ArXiv*, abs/1309.6835, 2013.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012.

Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine learning research*, 5(Jan):27–72, 2004.

Ping Ma, Jianhua Z Huang, and Nan Zhang. Efficient computation of smoothing splines via adaptive basis sampling. *Biometrika*, 102(3):631–645, 2015.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

César Lincoln C. Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A. Barreto, and Neil D. Lawrence. Recurrent Gaussian processes. In *ICLR*, 2016.

Charles A. Micchelli and Massimiliano Pontil. On learning vector-valued functions. *Neural Computation*, 17:177–204, 2005.

John Moeller, Sarathkrishna Swaminathan, and Suresh Venkatasubramanian. A unified view of localized kernel learning. In *SDM*, pages 252–260. SIAM, 2016.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zach DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Ilyes Rebai, Yassine Ben Ayed, and Walid Mahdi. Deep multilayer multiple kernel learning. *Neural Computing and Applications*, 27:2305–2314, 2015.

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes. *NeurIPS*, 30, 2017.

Hugh Salimbeni, Vincent Dutordoir, James Hensman, and Marc Deisenroth. Deep gaussian processes with importance-weighted variational inference. In *ICML*, pages 5589–5598. PMLR, 2019.

B. Scholkopf and A. J. Smola. *Learning with Kernels: Support vector machines, regularization, optimization, and beyond*. Cambridge: The MIT Press, 2001.

Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL*, 2018.

Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

Michalis K. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *AISTATS*, 2009.

Ke Alexander Wang, Geoff Pleiss, Jacob R. Gardner, Stephen Tyree, Kilian Q. Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. In *NeurIPS*, 2019.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *NeurIPS*, 13, 2000.

Andrew Gordon Wilson and Ryan P. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *ICML*, 2013.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *AISTATS*, 2016.

Jinfeng Zhuang, Ivor Wai-Hung Tsang, and Steven C. H. Hoi. Two-layer multiple kernel learning. In *AISTATS*, 2011.

Appendix A. Network structures

We provide the neural network structure used in the proposed models, $\phi(x) : \mathbb{R}^d \to \mathbb{R}^p$, $\psi(x) : \mathbb{R}^d \to \mathbb{R}^p$, $\alpha(\cdot) : \mathbb{R}^d \to \mathbb{R}^J$, $\beta(\cdot) : \mathbb{R}^d \to \mathbb{R}^J_+$, $\eta(\cdot) : \mathbb{R}^d \to \mathbb{R}^M$, where d is the dimension of input data, p is the dimension of extracted features, J is the number of representative points and M is the number of different types of kernels. Moreover, X denotes one input data, n denotes a number of J uniform distributed random data. All of these neural networks are chosen as MLP networks. The detailed structures are shown in Table 5-9.

Table 5: Network structure of $\phi(\cdot)$.

inputs	layer	outputs	dimension of outputs	activation function
X	-	-	1*d	-
X	Linear	h_1	1*128	ReLU
h_1	Linear	h_2	1*64	ReLU
h_2	Linear	$\phi(X)$	1*p	-

Table 6: Network structure of $\psi(\cdot)$.

inputs	layer	outputs	dimension of outputs	activation function
n	-	-	J*d	-
n	Linear	h_1	J*64	ReLU
h_1	Linear	h_2	J*32	ReLU
h_2	Linear	$\psi(n)$	J*p	-

Table 7: Network structure of $\alpha(\cdot)$.

inputs	layer	outputs	dimension of outputs	activation function
X	-	-	1*d	-
X	Linear	h_1	1*64	ReLU
h_1	Linear	h_2	1*128	ReLU
h_2	Linear	$\alpha(X)$	1*J	-

Table 8: Network structure of $\beta(\cdot)$.

inputs	layer	outputs	dimension of outputs	activation function
X	-	-	1*d	-
X	Linear	$\beta(X)$	1*J	ReLU

Table 9: Network structure of $\eta(\cdot)$.

inputs	layer	outputs	dimension of outputs	activation function
$\psi(n)$	-	-	J*p	-
$\psi(n)$	Linear	h_1	J*32	ReLU
h_1	Linear	h_2	J*3	ReLU
h_2	Softmax	$\eta(\psi(n))$	J*M	-