Deep Kernel Learning

Andrew Gordon Wilson[*]
CMU

Zhiting Hu[*]
CMU

Ruslan Salakhutdinov
University of Toronto

Eric P. Xing
CMU

Abstract

We introduce scalable deep kernels, which combine the structural properties of deep learning architectures with the non-parametric flexibility of kernel methods. Specifically, we transform the inputs of a spectral mixture base kernel with a deep architecture, using local kernel interpolation, inducing points, and structure exploiting (Kronecker and Toeplitz) algebra for a scalable kernel representation. These closed-form kernels can be used as drop-in replacements for standard kernels, with benefits in expressive power and scalability. We jointly learn the properties of these kernels through the marginal likelihood of a Gaussian process. Inference and learning cost $\mathcal{O}(n)$ for n training points, and predictions cost $\mathcal{O}(1)$ per test point. On a large and diverse collection of applications, including a dataset with 2 million examples, we show improved performance over scalable Gaussian processes with flexible kernel learning models, and stand-alone deep architectures.

1 Introduction

"How can Gaussian processes possibly replace neural networks? Have we thrown the baby out with the bathwater?" questioned MacKay (1998). It was the late 1990s, and researchers had grown frustrated with the many design choices associated with neural networks – regarding architecture, activation functions, and regularisation – and the lack of a principled framework to guide in these choices.

Gaussian processes had recently been popularised within the machine learning community by Neal (1996), who had shown that Bayesian neural networks with infinitely many hidden units converged to Gaussian processes with a particular kernel (covariance) function. Gaussian processes were subsequently viewed as flexible and interpretable alternatives to neural networks, with straightforward learning procedures. Where neural networks used finitely many highly adaptive basis functions, Gaussian processes typically used infinitely many fixed basis functions. As argued by MacKay (1998), Hinton et al. (2006), and Bengio (2009), neural networks could automatically discover meaningful representations in high-dimensional data by learning multiple layers of highly adaptive basis functions. By contrast, Gaussian processes with popular kernel functions were used typically as simple smoothing devices.

Recent approaches (e.g., Yang et al., 2015; Lloyd et al., 2014; Wilson, 2014; Wilson and Adams, 2013) have demonstrated that one can develop more expressive kernel functions, which are indeed able to discover rich structure in data without human intervention. Such methods effectively use infinitely many adaptive basis functions. The relevant question then becomes *not* which paradigm (e.g., kernel methods or neural networks) replaces the other, but whether we can combine the advantages of each approach. Indeed, deep neural networks provide a powerful mechanism for creating adaptive basis functions, with inductive biases which have proven effective for learning in many application domains, including visual object recognition, speech perception, language understanding, and information retrieval (Krizhevsky et al., 2012; Hinton et al., 2012; Socher et al., 2011; Kiros et al., 2014; Xu et al., 2015).

In this paper, we combine the non-parametric flexibility of kernel methods with the structural properties of deep neural networks. In particular, we use deep feedforward fully-connected and convolutional networks, in combination with spectral mixture covariance functions (Wilson and Adams, 2013), inducing points (Quiñonero-Candela and Rasmussen, 2005), structure exploiting algebra (Saatchi, 2011), and local kernel interpolation (Wilson and Nickisch, 2015; Wilson et al., 2015), to create scalable expressive closed form covariance kernels for Gaussian processes.

[*] Authors contributed equally. Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51. Copyright 2016 by the authors.

As a non-parametric method, the information capacity of our model grows with the amount of available data, but its complexity is automatically calibrated through the marginal likelihood of the Gaussian process, without the need for regularization or cross-validation (Rasmussen and Ghahramani, 2001; Rasmussen and Williams, 2006; Wilson, 2014). The flexibility and automatic calibration provided by the non-parametric layer typically provides a high standard of performance, while reducing the need for extensive hand tuning from the user.

We further build on the ideas in KISS-GP (Wilson and Nickisch, 2015) and extensions (Wilson et al., 2015), so that our deep kernel learning model can scale *linearly* with the number of training instances n, instead of $\mathcal{O}(n^3)$ as is standard with Gaussian processes (GPs), while retaining a fully non-parametric representation. Our approach also scales as $\mathcal{O}(1)$ per test point, instead of the standard $\mathcal{O}(n^2)$ for GPs, allowing for very fast prediction times. Because KISS-GP creates an approximate kernel from a user specified kernel for fast computations, independently of a specific inference procedure, we can view the resulting kernel as a *scalable deep kernel*. We demonstrate the value of this scalability in the experimental results section, where it is the large datasets that provide the greatest opportunities for our model to discover expressive statistical representations.

We begin by reviewing related work in section 2, and providing background on Gaussian processes in section 3. In section 4 we derive scalable closed form deep kernels, and describe how to perform efficient automatic learning of these kernels through the Gaussian process marginal likelihood. In section 5, we show substantially improved performance over standard Gaussian processes, expressive kernel learning approaches, deep neural networks, and Gaussian processes applied to the outputs of trained deep networks, on a wide range of datasets. We also interpret the learned kernels to gain new insights into our modelling problems.

2 Related Work

Given the intuitive value of combining kernels and neural networks, it is encouraging that various distinct forms of such combinations have been considered in different contexts.

The Gaussian process regression network (Wilson et al., 2012) replaces all weight connections and activation functions in a Bayesian neural network with Gaussian processes, allowing the authors to model input dependent correlations between multiple tasks. Alternatively, Damianou and Lawrence (2013) replace every activation function in a Bayesian neural network with a Gaussian process transformation, in an unsupervised setting. While promising, both models are very task specific, and require sophisticated approximate Bayesian inference which is much more demanding than what is required by standard Gaussian processes or deep learning models, and typically does not scale beyond a few thousand training points. Similarly, Salakhutdinov and Hinton (2008) combine deep belief networks (DBNs) with Gaussian processes, showing improved performance over standard GPs with RBF kernels, in the context of semi-supervised learning. However, their model is heavily relying on unsupervised pre-training of DBNs, with the GP component unable to scale beyond a few thousand training points. Likewise, Calandra et al. (2014) combine a feedforward neural network transformation with a Gaussian process, showing an ability to learn sharp discontinuities. However, similar to many other approaches, the resulting model can only scale to at most a few thousand data points.

In a frequentist setting, Yang et al. (2014) combine convolutional networks, with parameters pre-trained on ImageNet, with a scalable Fastfood (Le et al., 2013) expansion for the RBF kernel applied to the final layer. The resulting method is scalable and flexible, but the network parameters generally must first be trained separately from the Fastfood features, and the combined model remains parametric, due to the parametric expansion provided by Fastfood. Careful attention must still be paid to training procedures, regularization, and manual calibration of the network architecture. In a similar manner, Huang et al. (2015) and Snoek et al. (2015) have combined deep architectures with parametric Bayesian models. Huang et al. (2015) pursue an unsupervised pre-training procedure using deep autoencoders, showing improved performance over GPs using standard kernels. Snoek et al. (2015) show promising performance on Bayesian optimisation tasks, for tuning the parameters of a deep neural network, but do not use a Bayesian (marginal likelihood) objective for training network parameters.

Our approach is distinct in that we combine deep feedforward and convolutional architectures with spectral mixture covariances (Wilson and Adams, 2013), inducing points, Kronecker and Toeplitz algebra, and local kernel interpolation (Wilson and Nickisch, 2015; Wilson et al., 2015), to derive expressive and scalable *closed form* kernels, where all parameters are trained jointly with a unified supervised objective, as part of a *non-parametric* Gaussian process framework, without requiring approximate Bayesian inference. Moreover, the simple joint learning procedure in our approach can be applied in general settings. Indeed we show that the proposed model outperforms state of the art stand-alone deep learning architectures and Gaussian processes with advanced kernel learning procedures on a wide range of datasets, demonstrating its practical significance. We also show that jointly training all the weights of a deep kernel through the marginal likelihood of a non-parametric GP provides significant advantages over training a GP applied to the output layer of a trained deep neural network. Moreover, we achieve scalability while retaining non-parametric model structure by leveraging the very recent KISS-GP approach (Wilson and Nickisch, 2015) and extensions in Wilson et al. (2015) for efficiently representing kernel functions, to produce scalable deep kernels.

3 Gaussian Processes

We briefly review Gaussian processes (GPs) and the computational requirements for predictions and kernel learning (see e.g., Rasmussen and Williams (2006) for a comprehensive discussion of GPs).

We assume a dataset $\mathcal{D}$ of n input (predictor) vectors $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, each of dimension D, which index an $n \times 1$ vector of targets $\mathbf{y} = (y(\mathbf{x}_1), \dots, y(\mathbf{x}_n))^\top$. If $f(\mathbf{x}) \sim \mathcal{GP}(\mu, k_{\boldsymbol{\gamma}})$, then any collection of function values $\mathbf{f}$ has a joint Gaussian distribution,

$$\mathbf{f} = f(X) = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)]^\top \sim \mathcal{N}(\boldsymbol{\mu}, K_{X,X}) \,, \quad (1)$$

with mean vector and covariance matrix defined by the mean function and covariance kernel of the Gaussian process: $\boldsymbol{\mu}_i = \mu(\mathbf{x}_i)$, and $(K_{X,X})_{ij} = k_{\boldsymbol{\gamma}}(\mathbf{x}_i, \mathbf{x}_j)$, where the kernel $k_{\boldsymbol{\gamma}}$ is parametrized by $\boldsymbol{\gamma}$. Assuming additive Gaussian noise, $y(\mathbf{x})|f(\mathbf{x}) \sim \mathcal{N}(y(\mathbf{x}); f(\mathbf{x}), \sigma^2)$, the predictive distribution of the GP evaluated at the n_* test points indexed by X_*, is given by

$$\begin{aligned} \mathbf{f}_*|X_*, X, \mathbf{y}, \boldsymbol{\gamma}, \sigma^2 &\sim \mathcal{N}(\mathbb{E}[\mathbf{f}_*], \mathrm{cov}(\mathbf{f}_*)) \,, \quad (2) \\ \mathbb{E}[\mathbf{f}_*] &= \boldsymbol{\mu}_{X_*} + K_{X_*,X}[K_{X,X} + \sigma^2 I]^{-1}(\mathbf{y} - \boldsymbol{\mu}_X) \,, \\ \mathrm{cov}(\mathbf{f}_*) &= K_{X_*,X_*} - K_{X_*,X}[K_{X,X} + \sigma^2 I]^{-1} K_{X,X_*} \,. \end{aligned}$$

$K_{X_*,X}$, for example, represents the $n_* \times n$ matrix of covariances between the GP evaluated at X_* and X. $\boldsymbol{\mu}_X$ and $\boldsymbol{\mu}_{X_*}$ are mean vectors evaluated at X and X_*, and $K_{X,X}$ is the $n \times n$ covariance matrix evaluated at training inputs X. All covariance matrices implicitly depend on the kernel hyperparameters $\boldsymbol{\gamma}$.

GPs with RBF kernels correspond to models which have an infinite basis expansion in a dual space, and have compelling theoretical properties: these models are universal approximators, and have prior support to within an arbitrarily small epsilon band of any continuous function (Micchelli et al., 2006). Indeed the properties of the distribution over functions induced by a Gaussian process are controlled by the kernel function. For example, the popular RBF kernel,

$$k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2}||\mathbf{x} - \mathbf{x}'||/\ell^2) \quad (3)$$

encodes the inductive bias that function values closer together in the input space are more correlated. The complexity of the functions in the input space is determined by the interpretable length-scale hyperparameter ℓ. Shorter length-scales correspond to functions which vary more rapidly with the inputs $\mathbf{x}$.

The structure of our data is discovered through learning interpretable kernel hyperparameters. The marginal likelihood of the targets $\mathbf{y}$, the probability of the data conditioned only on kernel hyperparameters $\boldsymbol{\gamma}$, provides a principled probabilistic framework for kernel learning:

$$\log p(\mathbf{y}|\boldsymbol{\gamma}, X) \propto -[\mathbf{y}^\top (K_{\boldsymbol{\gamma}} + \sigma^2 I)^{-1} \mathbf{y} + \log |K_{\boldsymbol{\gamma}} + \sigma^2 I|] \,, \quad (4)$$

where we have used $K_{\boldsymbol{\gamma}}$ as shorthand for $K_{X,X}$ given $\boldsymbol{\gamma}$. Note that the expression for the log marginal likelihood in Eq. (4) pleasingly separates into automatically calibrated model fit and complexity terms (Rasmussen and Ghahramani, 2001). Kernel learning is performed by optimizing Eq. (4) with respect to $\boldsymbol{\gamma}$.

The computational bottleneck for inference is solving the linear system $(K_{X,X} + \sigma^2 I)^{-1}\mathbf{y}$, and for kernel learning is computing the log determinant $\log |K_{X,X} + \sigma^2 I|$. The standard approach is to compute the Cholesky decomposition of the $n \times n$ matrix $K_{X,X}$, which requires $\mathcal{O}(n^3)$ operations and $\mathcal{O}(n^2)$ storage. After inference, the predictive mean costs $\mathcal{O}(n)$, and the predictive variance costs $\mathcal{O}(n^2)$, per test point $\mathbf{x}_*$.

4 Deep Kernel Learning

In this section we show how we can contruct kernels which encapsulate the expressive power of deep architectures, and how to learn the properties of these kernels as part of a scalable probabilistic GP framework.

Specifically, starting from a base kernel $k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta})$ with hyperparameters $\boldsymbol{\theta}$, we transform the inputs (predictors) $\mathbf{x}$ as

$$k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta}) \to k(g(\mathbf{x}_i, \mathbf{w}), g(\mathbf{x}_j, \mathbf{w})|\boldsymbol{\theta}, \mathbf{w}) \,, \quad (5)$$

where $g(\mathbf{x}, \mathbf{w})$ is a non-linear mapping given by a deep architecture, such as a deep convolutional network, parametrized by weights $\mathbf{w}$. The popular RBF kernel (Eq. (3)) is a sensible choice of base kernel $k(\mathbf{x}_i, \mathbf{x}_j|\boldsymbol{\theta})$. For added flexibility, we also propose to use *spectral mixture* base kernels (Wilson and Adams, 2013):

$$k_{\mathrm{SM}}(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}) = \quad (6)$$
$$\sum_{q=1}^{Q} a_q \frac{|\Sigma_q|^{\frac{1}{2}}}{(2\pi)^{\frac{D}{2}}} \exp\left(-\frac{1}{2}||\Sigma_q^{\frac{1}{2}}(\mathbf{x} - \mathbf{x}')||^2\right) \cos\langle \mathbf{x} - \mathbf{x}', 2\pi\boldsymbol{\mu}_q \rangle \,.$$

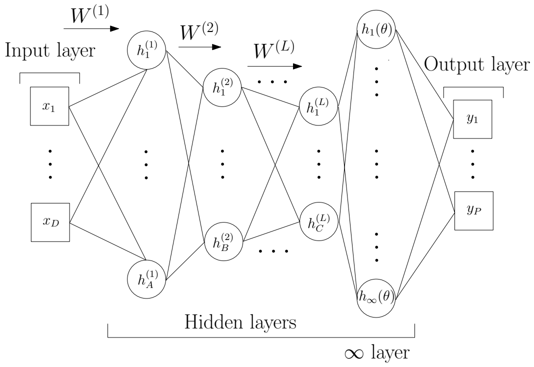

Figure 1: Deep Kernel Learning: A Gaussian process with a deep kernel maps D dimensional inputs $\mathbf{x}$ through L parametric hidden layers followed by a hidden layer with an infinite number of basis functions, with base kernel hyperparameters $\boldsymbol{\theta}$. Overall, a Gaussian process with a deep kernel produces a probabilistic mapping with an infinite number of adaptive basis functions parametrized by $\boldsymbol{\gamma} = \{\mathbf{w}, \boldsymbol{\theta}\}$. All parameters $\boldsymbol{\gamma}$ are learned *jointly* through the marginal likelihood of the Gaussian process.

The parameters of the spectral mixture kernel $\boldsymbol{\theta} = \{a_q, \Sigma_q, \boldsymbol{\mu}_q\}$ are mixture weights, bandwidths (inverse length-scales), and frequencies. The spectral mixture (SM) kernel, which forms an expressive basis for all stationary covariance functions, can discover quasi-periodic stationary structure with an interpretable and succinct representation, while the deep learning transformation $g(\mathbf{x}, \mathbf{w})$ captures non-stationary and hierarchical structure.

We use the deep kernel of Eq. (5) as the covariance function of a Gaussian process to model data $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$. Conditioned on all kernel hyperparameters, we can interpret our model as applying a Gaussian process with base kernel $k_{\boldsymbol{\theta}}$ to the final hidden layer of a deep network. Since a GP with (RBF or SM) base kernel $k_{\boldsymbol{\theta}}$ corresponds to an infinite basis function representation, our network effectively has a hidden layer with an *infinite* number of hidden units. The overall model is shown in Figure 1.

We emphasize, however, that we *jointly* learn all deep kernel hyperparameters, $\boldsymbol{\gamma} = \{\mathbf{w}, \boldsymbol{\theta}\}$, which include $\mathbf{w}$, the weights of the network, and $\boldsymbol{\theta}$ the parameters of the base kernel, by maximizing the log *marginal likelihood* $\mathcal{L}$ of the Gaussian process (see Eq. (4)). Indeed compartmentalizing our model into a base kernel and deep architecture is for pedagogical clarity. When applying a Gaussian process one can use our deep kernel, which operates as a single unit, as a drop-in replacement for e.g., standard ARD or Matérn kernels (Rasmussen and Williams, 2006), since learning and inference follow the same procedures. In the experiments, we show that jointly learning all deep kernel parameters has advantages over training a GP applied to the output layer of a trained deep neural network.

For kernel learning, we use the chain rule to compute derivatives of the log marginal likelihood with respect to the deep kernel hyperparameters:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}}{\partial K_{\boldsymbol{\gamma}}} \frac{\partial K_{\boldsymbol{\gamma}}}{\partial \boldsymbol{\theta}}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial K_{\boldsymbol{\gamma}}} \frac{\partial K_{\boldsymbol{\gamma}}}{\partial g(\mathbf{x}, \mathbf{w})} \frac{\partial g(\mathbf{x}, \mathbf{w})}{\partial \mathbf{w}}.$$

The implicit derivative of the log marginal likelihood with respect to our $n \times n$ data covariance matrix $K_{\boldsymbol{\gamma}}$ is given by

$$\frac{\partial \mathcal{L}}{\partial K_{\boldsymbol{\gamma}}} = \frac{1}{2}(K_{\boldsymbol{\gamma}}^{-1}\mathbf{y}\mathbf{y}^\top K_{\boldsymbol{\gamma}}^{-1} - K_{\boldsymbol{\gamma}}^{-1}), \tag{7}$$

where we have absorbed the noise covariance $\sigma^2 I$ into our covariance matrix, and treat it as part of the base kernel hyperparameters $\boldsymbol{\theta}$. $\frac{\partial K_{\boldsymbol{\gamma}}}{\partial \boldsymbol{\theta}}$ are the derivatives of the deep kernel with respect to the base kernel hyperparameters (such as length-scale), conditioned on the fixed transformation of the inputs $g(\mathbf{x}, \mathbf{w})$. Similarly, $\frac{\partial K_{\boldsymbol{\gamma}}}{\partial g(\mathbf{x}, \mathbf{w})}$ are the implicit derivatives of the deep kernel with respect to g, holding $\boldsymbol{\theta}$ fixed. The derivatives with respect to the weight variables $\frac{\partial g(\mathbf{x}, \mathbf{w})}{\partial \mathbf{w}}$ are computed using standard backpropagation.

For scalability, we replace all instances of $K_{\boldsymbol{\gamma}}$ with the KISS-GP covariance matrix (Wilson and Nickisch, 2015; Wilson et al., 2015)

$$K_{\boldsymbol{\gamma}} \approx M K_{U,U}^{\text{deep}} M^\top := K_{\text{KISS}}, \tag{8}$$

where M is a sparse matrix of interpolation weights, containing only 4 non-zero entries per row for local cubic interpolation, and $K_{U,U}$ is a covariance matrix created from our deep kernel, evaluated over m latent inducing points $U = [\mathbf{u}_i]_{i=1\ldots m}$. We place the inducing points over a regular multidimensional lattice, and exploit the resulting decomposition of $K_{U,U}$ into a Kronecker product of Toeplitz matrices for extremely fast matrix vector multiplications (MVMs), without requiring any grid structure in the data inputs X or the transformed inputs $g(\mathbf{x}, \mathbf{w})$. Because KISS-GP operates by creating an approximate kernel which admits fast computations, and is independent from a specific inference and learning procedure, we can view the KISS approximation applied to our deep kernels as a *stand-alone* kernel, $k(\mathbf{x}, \mathbf{z}) = \mathbf{m}_{\mathbf{x}}^\top K_{U,U}^{\text{deep}} \mathbf{m}_{\mathbf{z}}$, which can be combined with Gaussian processes or other kernel machines for scalable learning.

For inference we solve $K_{\text{KISS}}^{-1}\mathbf{y}$ using linear conjugate gradients (LCG), an iterative procedure for solving linear systems which only involves matrix vector multiplications (MVMs). The number of iterations required for convergence to within machine precision is $j \ll n$, and in practice j depends on the conditioning of the KISS-GP covariance matrix rather than the number of training points n. For estimating the log determinant

Table 1: Comparative RMSE performance and runtime on UCI regression datasets, with n training points and d the input dimensions. The results are averaged over 5 equal partitions (90% train, 10% test) of the data ± one standard deviation. The *best* denotes the best-performing kernel according to Yang et al. (2015) (note that often the best performing kernel is GP-SM). Following Yang et al. (2015), as exact Gaussian processes are intractable on the large data used here, the Fastfood finite basis function expansions are used for approximation in GP (RBF, SM, Best). We verified on datasets with $n < 10,000$ that exact GPs with RBF kernels provide comparable performance to the Fastfood expansions. For datasets with $n < 6,000$ we used a fully-connected DNN with a [d-1000-500-50-2] architecture, and for $n > 6000$ we used a [d-1000-1000-500-50-2] architecture. DNN+GP is a GP applied to fixed pre-trained output layer of the DNN. We used RBF kernel and KISS-GP approximation for direct comparison with our proposed deep kernel learning (DKL). We consider scalable DKL with RBF and SM base kernels. For the SM base kernel, we set $Q = 4$ for datasets with $n < 10,000$ training instances, and use $Q = 6$ for larger datasets.

Datasets	n	d	RMSE							Runtime(s)		
			GP			DNN	DNN+GP	DKL		DNN	DKL	
			RBF	SM	best			RBF	SM		RBF	SM
Gas	2,565	128	0.21±0.07	0.14±0.08	0.12±0.07	0.11±0.05	0.11±0.05	0.11±0.05	**0.09±0.06**	7.4	7.8	10.5
Skillcraft	3,338	19	1.26±3.14	0.25±0.02	0.25±0.02	**0.25±0.00**	**0.25±0.00**	**0.25±0.00**	**0.25±0.00**	15.8	15.9	17.1
SML	4,137	26	6.94±0.51	0.27±0.03	0.26±0.04	0.25±0.02	0.25±0.01	0.24±0.01	**0.23±0.01**	1.1	1.5	1.9
Parkinsons	5,875	20	3.94±1.31	**0.00±0.00**	**0.00±0.00**	0.31±0.04	0.31±0.04	0.29±0.04	0.29±0.04	3.2	3.4	6.5
Pumadyn	8,192	32	1.00±0.00	0.21±0.00	**0.20±0.00**	0.25±0.02	0.25±0.02	0.24±0.02	0.23±0.02	7.5	7.9	9.8
PoleTele	15,000	26	12.6±0.3	5.40±0.3	4.30±0.2	3.42±0.05	3.36±0.04	3.28±0.04	**3.11±0.07**	8.0	8.3	27.0
Elevators	16,599	18	0.12±0.00	0.090±0.001	0.089±0.002	0.099±0.001	0.097±0.002	**0.084±0.002**	**0.084±0.002**	8.9	9.2	11.8
Kin40k	40,000	8	0.34±0.01	0.19±0.02	0.06±0.00	0.11±0.01	0.11±0.01	0.05±0.00	**0.03±0.01**	19.8	20.7	25.0
Protein	45,730	9	1.64±1.66	0.50±0.02	0.47±0.01	0.49±0.01	0.49±0.01	0.46±0.01	**0.43±0.01**	143	155	144
KEGG	48,827	22	0.33±0.17	0.12±0.01	0.12±0.01	0.12±0.01	0.12±0.00	0.11±0.00	**0.10±0.01**	31.3	34.2	61.0
CTslice	53,500	385	7.13±0.11	2.21±0.06	0.59±0.07	0.41±0.06	0.41±0.02	0.36±0.01	**0.34±0.02**	36.4	44.3	80.4
KEGGU	63,608	27	0.29±0.12	0.12±0.00	0.12±0.00	0.12±0.00	0.12±0.00	**0.11±0.00**	**0.11±0.00**	39.5	43.0	41.1
3Droad	434,874	3	12.9±0.1	10.3±0.2	9.90±0.10	7.36±0.07	7.04±0.06	**6.91±0.04**	**6.91±0.04**	239	256	292
Song	515,345	90	0.55±0.00	0.46±0.00	0.45±0.00	0.45±0.02	0.45±0.01	0.44±0.00	**0.43±0.01**	518	539	590
Buzz	583,250	77	0.88±0.01	0.51±0.01	0.51±0.01	0.49±0.00	0.49±0.00	0.48±0.00	**0.46±0.01**	486	523	770
Electric	2M	11	0.23±0.00	0.053±0.000	0.053±0.000	0.058±0.002	0.054±0.002	0.050±0.002	**0.048±0.002**	3458	3542	4881

in the marginal likelihood we follow the approach described in Wilson and Nickisch (2015) with extensions in Wilson et al. (2015).

KISS-GP training scales as $\mathcal{O}(n + h(m))$ (where $h(m)$ is typically close to linear in m), versus conventional scalable GP approaches which require $\mathcal{O}(m^2 n + m^3)$ (Quiñonero-Candela and Rasmussen, 2005) computations and need $m \ll n$ for tractability, which results in severe deteriorations in predictive performance. The ability to have large $m \approx n$ allows KISS-GP to have near-exact accuracy in its approximation (Wilson and Nickisch, 2015), retaining a non-parametric representation, while providing linear scaling in n and $\mathcal{O}(1)$ time per test point prediction (Wilson et al., 2015). We empirically demonstrate this scalability and accuracy in our experiments of section 5.

5 Experiments

We evaluate the proposed deep kernel learning method on a wide range of regression problems, including a large and diverse collection of regression tasks from the UCI repository (section 5.1), orientation extraction from face patches (section 5.2), magnitude recovery of handwritten digits (section 5.3), and step function recovery (section 5.4 and the supplementary material). We show that the proposed algorithm substantially outperforms GPs with expressive kernel learning approaches, and deep neural networks, without any significant increases in computational overhead.

All experiments were performed on a Linux machine with eight 4.0GHz CPU cores and 32GB RAM. We implemented DNNs based on Caffe (Jia et al., 2014), a general deep learning platform.

For our deep kernel learning model, we first train a deep neural network using SGD with the squared loss objective, and rectified linear activation functions. After the neural network has been pre-trained, a KISS-GP model was fitted using the top-level features of the DNN model as inputs. Using this pre-training initialization, our joint deep kernel learning (DKL) model of section 4 is then trained by optimizing *all* the hyperparameters $\boldsymbol{\gamma}$ of our deep kernel, by backpropagating derivatives through the marginal likelihood of the Gaussian process (see Eq. 7).

5.1 UCI regression tasks

We consider a large set of UCI regression problems of varying sizes and properties. Table 1 reports test root mean squared error (RMSE) for 1) many scalable Gaussian process kernel learning methods based on Fastfood (Yang et al., 2015); 2) stand-alone deep neural networks (DNNs); and 3) our proposed combined deep kernel learning (DKL) model using both RBF and SM base kernels.

For smaller datasets, where the number of training examples $n < 6,000$, we used a fully-connected neural network with a d-1000-500-50-2 architecture; for larger

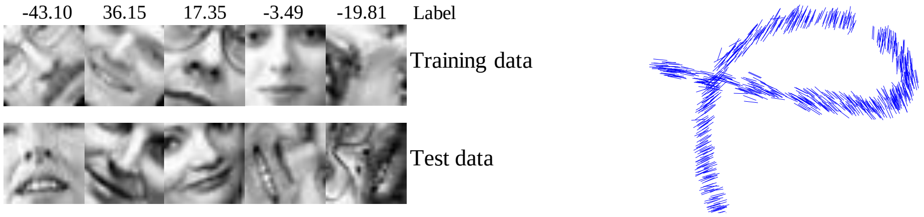

Figure 2: **Left**: Randomly sampled training and test examples. **Right**: The two dimensional outputs of the convolutional network on a set of test cases. Each point is shown using a line segment that has the same orientation as the input face.

datasets we used a d-1000-1000-500-50-2 architecture[1].

Table 1 shows that on most of the datasets, our DKL method strongly outperforms not only Gaussian processes with the standard RBF kernel, but also the best-performing kernels selected from a wide range of alternative kernel learning procedures (Yang et al., 2015).

We further compared DKL to stand-alone deep neural networks which have the exact same architecture as the DNN component of DKL, and DNN+GP which is a GP applied to a pre-trained DNN. We see that DNN+GP outperforms stand-alone DNNs, showing the non-parametric flexibility of kernel methods. By combining KISS-GP with DNNs as part of a *joint* DKL procedure, we obtain consistently better results than DNN and DNN+GP over all 16 datasets. Moreover, using a spectral mixture base kernel (Eq. (6)) to create a deep kernel provides notable additional performance improvements. By effectively learning the salient features from raw data, plain DNNs generally achieve competitive performance compared to expressive GPs. Combining the complementary advantages of these approaches into scalable deep kernels consistently brings substantial additional performance gains.

We next investigate the runtime of DKL. Table 1, right panel, compares DKL with a stand-alone DNN in terms of runtime for evaluating the objective and derivatives (i.e. one forward and backpropagation pass for DNN; one computation of the marginal likelihood and all relevant derivatives for DKL). We see that in addition to improving accuracy, combining KISS-GP with DNNs for deep kernels introduces only negligible runtime costs: KISS-GP imposes an additional runtime of about 10% over a stand-alone DNN. Overall, these results show the general applicability and practical significance of our scalable DKL approach.

5.2 Face orientation extraction

We now consider the task of predicting the orientation of a face extracted from a gray-scale image patch, explored in Salakhutdinov and Hinton (2008). We investigate our DKL procedure for efficiently learning meaningful representations from high-dimensional highly-structured image data.

Table 2: RMSE performance on Olivetti and MNIST. For comparison, in the face orientation extraction, we trained DKL on the same amount (12,000) of training instances as with DBN+GP, but used all labels; whereas DBN+GP (as with GP) scaled to only 1,000 labeled images and modeled the remaining data through unsupervised pretraining of DBN. CNN+GP is a GP applied to fixed pre-trained CNN. We used RBF base kernel within GPs.

Datasets	GP	DBN+GP	CNN	CNN+GP	DKL
Olivetti	16.33	6.42	6.34	6.42	**6.07**
MNIST	1.25	1.03	0.59	0.56	**0.53**

The Olivetti face data set contains ten 64×64 images of forty different people, for 400 images total. Following Salakhutdinov and Hinton (2008), we constructed datasets of 28×28 images by randomly rotating (uniformly from -90° to $+90^\circ$), cropping, and subsampling the original 400 images. We then randomly select 30 people uniformly and collect their images as training data, while using the images of the remaining 10 people as test data. Figure 2 shows randomly sampled examples from the training and test data.

For training DKL on the Olivetti face patches we used a convolutional network consisting of 2 convolutional layers followed by 4 fully-connected layers, mapping a face patch to a 2-dimensional feature vector, with a SM base kernel. We describe this convolutional architecture in detail in the supplementary material.

Table 2 shows the RMSE of the predicted face orientations using four models. The DBN+GP model, proposed by Salakhutdinov and Hinton (2008), first extracts features from raw data using a Deep Belief Network (DBN), and then applies a Gaussian process with an RBF kernel. However, their approach could only handle up to a few thousand labelled datapoints, due to the $O(n^3)$ complexity of standard Gaussian processes. The remaining data were modeled through unsupervised learning of a DBN, leaving the large amount of available labels unused.

Our proposed deep kernel methods, by contrast, scale linearly with the size of training data, and are capable of directly modeling the full labeled data to accurately recover salient patterns. Figure 2, right panel, shows that the deep kernel discovers features essential for orientation prediction, while filtering out irrelevant factors such as identities and scales.

[1] We found [d-1000-1000-500-50] architectures provide a similar level of performance, but scalable Kronecker algebra is most effective if the network maps into $D \leq 5$ dimensional spaces.

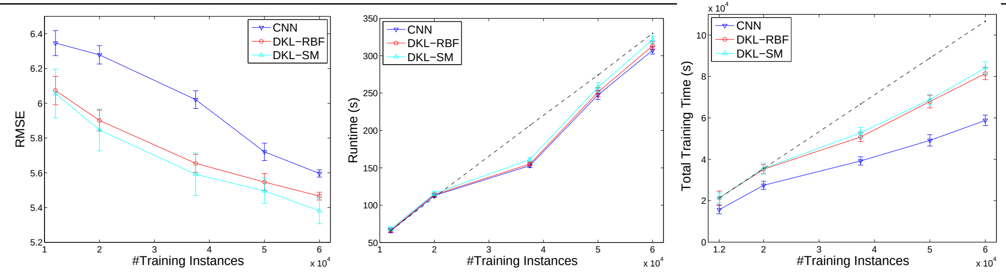

Figure 3: **Left**: RMSE vs. n, the number of training examples. **Middle**: Runtime vs n. **Right**: Total training time vs n. The dashed line in black indicates a slope of 1. CNNs are used within DKL. We set $Q = 4$ for the SM kernel.

Figure 3, left panel, further validates the benefit of scaling to large data. As more training data are used, our model continues to increase in accuracy. Indeed, it is the large datasets that will provide the greatest opportunities for our model to discover expressive statistical representations.

In Figure 4 we show the spectral density (the Fourier transform) of the *base* kernels learned through our deep kernel learning method. The expressive spectral mixture (SM) kernel discovers a structure with two peaks in the frequency domain. The RBF kernel is only able to use a single Gaussian in the spectral domain, centred at the origin. In an attempt to capture the significant mass near a frequency of 25, the RBF kernel spectral density spreads itself across the whole frequency domain, missing the important local correlations near a frequency $s = 0$, thus erroneously discarding much of the network features as white noise, since a broad spectral peak corresponds to a short lengthscale. This result provides intuition for why spectral mixture base kernels generally perform much better than RBF base kernels, despite the flexibility of the deep architecture.

We further see the benefit of an SM base kernel in Figure 5, where we show the learned covariance matrices constructed from the whole deep kernels (composition of base kernel and deep architecture) for RBF and SM base kernels. The covariance matrix is evaluated on a set of test inputs, where we randomly sample 400 instances from the test set and sort them in terms of the *orientation angles* of the input faces. We see that the deep kernels with both RBF and SM base kernels discover that faces with similar rotation angles are highly correlated, concentrating their largest entries on the diagonal (i.e., face pairs with similar orientations). Deep kernel learning with an SM base kernel captures these correlations more strongly than the RBF base kernel, which is somewhat more diffuse.

In Figure 5, right panel, we also show the learned covariance matrix for an RBF kernel with a standard Gaussian process applied to the raw data inputs. We see that the entries are very diffuse. In essence, through deep kernel learning, we can learn a metric where faces with similar rotation angles are highly correlated, and thus overcome the fundamental limitations of a Euclidean distance metric (used by standard kernels), where similar rotation angles are not particularly correlated, regardless of what hyper-parameters are learned with Euclidean kernels.

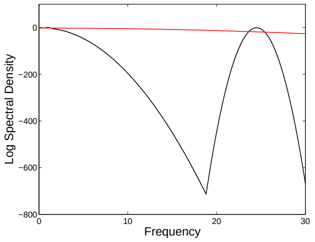

Figure 4: The log spectral densities of the DKL-SM and DKL-SE base kernels are in black and red, respectively.

We next measure the scalability of our model. Figure 3, middle panel, shows the runtimes in seconds, as a function of training instances, for evaluating the objective and any relevant derivatives. We see that, with the scalable KISS-GP, the joint model achieves a roughly linear asymptotic scaling, with a slope of 1. In Figure 3, right panel, we show how the total training time (i.e., the time for CNN pre-training plus the time for DKL with CNN architecture joint training) changes with varying the data size n. In addition to the linear scaling which is necessary for modeling large data, the fixed added time in combining KISS-GP with CNNs is modest, especially considering the gains in performance and expressive power.

5.3 Digit magnitude extraction

We map images of handwritten digits to a single real-value that is as close as possible to the integer represented by the digit in the image, as in Salakhutdinov and Hinton (2008). The MNIST digit dataset contains 60,000 training data and 10,000 test 28×28 images of ten handwritten digits (0 to 9). We used a convolutional neural network with a similar architecture as the LeNet (LeCun et al., 1998) (detailed in the supplementary material). Table 2 shows that a CNN performs considerably better than GP and DBN+GP, and DKL (with CNN architecture) further improves over CNN.

5.4 Step function recovery

We have so far considered RMSE for comparison to alternative methods where posterior predictive distri-

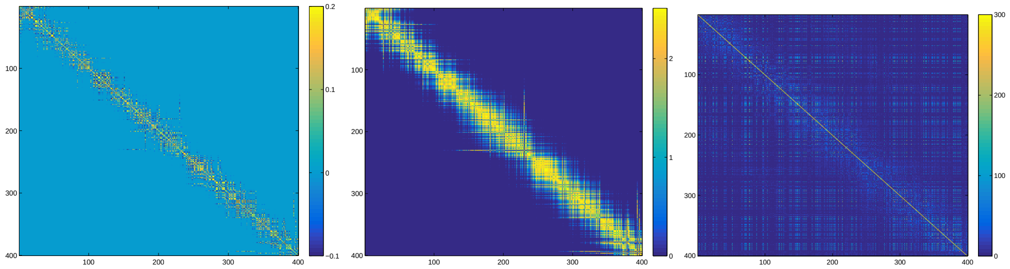

Figure 5: **Left**: The induced covariance matrix using DKL-SM kernel on a set of test cases, where the test samples are ordered by the *orientations* of the input faces. **Middle**: The respective covariance matrix using DKL-RBF kernel. **Right**: The respective covariance matrix using regular RBF kernel. The models are trained with $n = 12,000$, and $Q = 4$ for the SM base kernel.

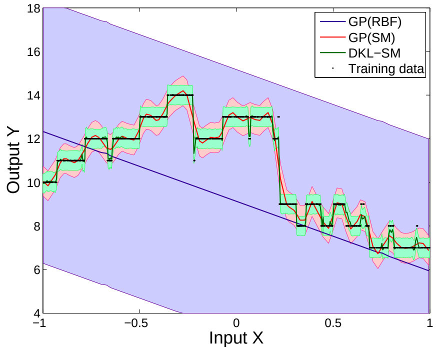

Figure 6: Recovering a step function. We show the predictive mean and 95% of the predictive probability mass for regular GPs with RBF and SM kernels, and DKL with SM base kernel. We set $Q = 4$ for SM kernels.

butions are not readily available, or on problems where RMSE has historically been used as a benchmark. However, an advantage of DKL over stand-alone deep architectures is the ability to naturally produce a posterior predictive distribution, which is especially useful in applications such as reinforcement learning and Bayesian optimisation. In Figure 6, we consider an example where we use DKL to learn the posterior predictive distribution for a step function with many challenging discontinuities. This problem is particularly difficult for conventional GP approaches, due to strong smoothness assumptions intrinsic to popular kernels.

GPs with SM kernels improve upon RBF kernels, but neither can properly adapt to the many sharp changes in covariance structure. By contrast, DKL-SM accurately encodes the discontinuities of the function, and has reasonable uncertainty over the whole domain. Further details are in the supplement.

6 Discussion

We have explored scalable deep kernels, which combine the structural properties of deep architectures with the non-parametric flexibility of kernel methods. In particular, we transform the inputs of a base kernel with a deep architecture, and then leverage local kernel interpolation, inducing points, and structure exploiting algebra (e.g., Kronecker and Toeplitz methods) for a scalable kernel representation. These scalable kernels can then be combined with Gaussian process inference and learning procedures for $\mathcal{O}(n)$ training and $\mathcal{O}(1)$ testing time. Moreover, we use spectral mixture covariances as a base kernel, which provides a significant additional boost in representational power. Overall, our scalable deep kernels can be used in place of standard kernels, following the same inference and learning procedures, but with benefits in expressive power and efficiency. We show on a wide range of experiments the general applicability and practical significance of our approach, consistently outperforming scalable GPs with expressive kernels, stand-alone DNNs, and GPs applied to the outputs of trained DNNs.

A major challenge in developing expressive kernel learning approaches is the Euclidean and absolute distance based metrics which are pervasive in most families of kernel functions, such as the ARD and Matérn kernels. Indeed, although intuitive in some cases, one cannot expect Euclidean and absolute distance as measures of similarity to be generally applicable, and they are especially problematic in high dimensional input spaces (Aggarwal et al., 2001). Modern approaches attempt to learn a flexible parametric family, for example, through weighted combinations of known kernels (e.g., Gönen and Alpaydın, 2011), but are still fundamentally limited to these standard notions of distance.

As we have seen in the Olivetti faces examples, our approach allows for the whole functional form of the metric to be learned in a flexible manner, through expressive transformations of the input space. We expect such metric learning to be particularly valuable in high dimensional classification problems, which we view as a promising direction for future research. We hope that this work will help bring together research on neural networks and kernel methods, to inspire many new models and unifying perspectives which combine the complementary advantages of these approaches.

Acknowledgements: We thank NIH R01GM093156 and NIH R01GM087694, NSF IIS-1218282, and ONR N000141410684 grants for support.

References

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). *On the surprising behavior of distance metrics in high dimensional space*. Springer.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*.

Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P. (2014). Manifold gaussian processes for regression. *arXiv preprint arXiv:1402.5876*.

Damianou, A. and Lawrence, N. (2013). Deep Gaussian processes. In *Artificial Intelligence and Statistics*.

Gönen, M. and Alpaydın, E. (2011). Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268.

Hinton, G. E., Deng, L., Yu, D., Dahl, G. E., rahman Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Huang, W., Zhao, D., Sun, F., Liu, H., and Chang, E. (2015). Scalable gaussian process regression using deep neural networks. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3576–3582. AAAI Press.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Kiros, R., Salakhutdinov, R., and Zemel, R. (2014). Unifying visual-semantic embeddings with multimodal neural language models. *TACL*.

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

Le, Q., Sarlos, T., and Smola, A. (2013). Fastfood-computing Hilbert space expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning*, pages 244–252.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lloyd, J. R., Duvenaud, D., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. (2014). Automatic construction and Natural-Language description of nonparametric regression models. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

MacKay, D. J. (1998). Introduction to Gaussian processes. In Bishop, C. M., editor, *Neural Networks and Machine Learning*, chapter 11, pages 133–165. Springer-Verlag.

Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal kernels. *The Journal of Machine Learning Research*, 7:2651–2667.

Neal, R. (1996). *Bayesian Learning for Neural Networks*. Springer Verlag.

Quiñonero-Candela, J. and Rasmussen, C. (2005). A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959.

Rasmussen, C. E. and Ghahramani, Z. (2001). Occam's razor. In *Neural Information Processing Systems (NIPS)*.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for Machine Learning*. The MIT Press.

Saatchi, Y. (2011). *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge.

Salakhutdinov, R. and Hinton, G. (2008). Using deep belief nets to learn covariance kernels for Gaussian processes. *Advances in Neural Information Processing Systems*, 20:1249–1256.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Ali, M., and Adams, R. P. (2015). Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*.

Socher, R., Huang, E., Pennington, J., Ng, A., and Manning, C. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24*, pages 801–809.

Wilson, A. G. (2014). *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*. PhD thesis, University of Cambridge.

Wilson, A. G. and Adams, R. P. (2013). Gaussian process kernels for pattern discovery and extrapolation. *International Conference on Machine Learning (ICML)*.

Wilson, A. G., Dann, C., and Nickisch, H. (2015). Thoughts on massively scalable Gaussian processes. *Technical Report, Carnegie Mellon University*. `http://www.cs.cmu.edu/~andrewgw/msgp.html`.

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2012). Gaussian process regression networks. In *International Conference on Machine Learning (ICML)*, Edinburgh. Omnipress.

Wilson, A. G. and Nickisch, H. (2015). Kernel interpolation for scalable structured Gaussian processes (KISS-GP). *International Conference on Machine Learning (ICML)*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *ICML*.

Yang, Z., Moczulski, M., Denil, M., de Freitas, N., Smola, A., Song, L., and Wang, Z. (2014). Deep fried convnets. *arXiv preprint arXiv:1412.7149*.

Yang, Z., Smola, A. J., Song, L., and Wilson, A. G. (2015). A la carte - learning fast kernels. *Artificial Intelligence and Statistics*.