Fast Kernel Learning for Multidimensional Pattern Extrapolation

Andrew Gordon Wilson[*]
Carnegie Mellon University
andrewgw@cs.cmu.edu

Elad Gilboa[*]
Washington University St. Louis
gilboae@ese.wustl.edu

Arye Nehorai
Washington University St. Louis
nehorai@ese.wustl.edu

John P. Cunningham
Columbia University
jpc2181@columbia.edu

Abstract

The ability to automatically discover patterns and perform extrapolation is an essential quality of intelligent systems. Kernel methods, such as Gaussian processes, have great potential for pattern extrapolation, since the kernel flexibly and interpretably controls the generalisation properties of these methods. However, automatically extrapolating large scale multidimensional patterns is in general difficult, and developing Gaussian process models for this purpose involves several challenges. A vast majority of kernels, and kernel learning methods, currently only succeed in smoothing and interpolation. This difficulty is compounded by the fact that Gaussian processes are typically only tractable for small datasets, and scaling an expressive kernel learning approach poses different challenges than scaling a standard Gaussian process model. One faces additional computational constraints, and the need to retain significant model structure for expressing the rich information available in a large dataset. In this paper, we propose a Gaussian process approach for large scale multidimensional pattern extrapolation. We recover sophisticated out of class kernels, perform texture extrapolation, inpainting, and video extrapolation, and long range forecasting of land surface temperatures, all on large multidimensional datasets, including a problem with 383,400 training points. The proposed method significantly outperforms alternative scalable and flexible Gaussian process methods, in speed and accuracy. Moreover, we show that a distinct combination of expressive kernels, a fully non-parametric representation, and scalable inference which exploits existing model structure, are critical for large scale multidimensional pattern extrapolation.

1   Introduction

Our ability to effortlessly extrapolate patterns is a hallmark of intelligent systems: even with large missing regions in our field of view, we can see patterns and textures, and we can visualise in our mind how they generalise across space. Indeed machine learning methods aim to automatically learn and generalise representations to new situations. Kernel methods, such as Gaussian processes (GPs), are popular machine learning approaches for non-linear regression and classification [1, 2, 3]. Flexibility is achieved through a kernel function, which implicitly represents an inner product of arbitrarily many basis functions. The kernel interpretably controls the smoothness and generalisation properties of a GP. A well chosen kernel leads to impressive empirical performances [2].

However, it is extremely difficult to perform large scale multidimensional pattern extrapolation with kernel methods. In this context, the ability to learn a representation of the data entirely depends

---

[*]Authors contributed equally.

on *learning* a kernel, which is a priori unknown. Moreover, kernel learning methods [4] are not typically intended for automatic pattern extrapolation; these methods often involve hand crafting combinations of Gaussian kernels (for smoothing and interpolation), for specific applications such as modelling low dimensional structure in high dimensional data. Without human intervention, the vast majority of existing GP models are unable to perform pattern discovery and extrapolation. While recent approaches such as [5] enable extrapolation on small one dimensional datasets, it is difficult to generalise these approaches for larger multidimensional situations. These difficulties arise because Gaussian processes are computationally intractable on large scale data, and while scalable approximate GP methods have been developed [6, 7, 8, 9, 10, 11, 12, 13], it is uncertain how to best scale expressive kernel learning approaches. Furthermore, the need for flexible kernel learning on large datasets is especially great, since such datasets often provide more information to learn an appropriate statistical representation.

In this paper, we introduce GPatt, a flexible, non-parametric, and computationally tractable approach to kernel learning for multidimensional pattern extrapolation, with particular applicability to data with grid structure, such as images, video, and spatial-temporal statistics. Specifically, the contributions of this paper include:

- We extend fast and exact Kronecker-based GP inference (e.g., [14]) to account for non-grid data. Our experiments include data where more than 70% of the training data are not on a grid. By adapting expressive spectral mixture kernels to the setting of multidimensional inputs and Kronecker structure, we achieve exact inference and learning costs of $\mathcal{O}(PN^{\frac{P+1}{P}})$ computations and $\mathcal{O}(PN^{\frac{2}{P}})$ storage, for N datapoints and P input dimensions, compared to the standard $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ storage associated with a Cholesky decomposition.

- We show that this combination of i) spectral mixture kernels (adapted for Kronecker structure); ii) scalable inference based on Kronecker methods (adapted for incomplete grids); and, iii) truly non-parametric representations, when used *in combination* (to form GPatt) distinctly enable large-scale multidimensional pattern extrapolation with GPs. We demonstrate this through a comparison with various expressive models and inference techniques: i) spectral mixture kernels with arguably the most popular scalable GP inference method (FITC) [10]; ii) a fast and efficient recent spectral based kernel learning method (SSGP) [6]; and, iii) the most popular GP kernels with Kronecker based inference.

- The information capacity of non-parametric methods grows with the size of the data. A truly non-parametric GP must have a kernel that is derived from an infinite basis function expansion. We find that a truly non-parametric representation is necessary for pattern extrapolation on large datasets, and provide insights into this surprising result.

- GPatt is highly scalable and accurate. This is the first time, as far as we are aware, that highly expressive *non-parametric* kernels with in some cases hundreds of hyperparameters, on datasets exceeding $N = 10^5$ training instances, can be learned from the marginal likelihood of a GP, in only minutes. Such experiments drive home the point that one can, to some extent, solve kernel selection, and automatically extract useful features from the data, on large datasets, using a special combination of expressive kernels and scalable inference.

- We show that the proposed methodology is rather distinct, as a GP model, in its ability to approach texture extrapolation and inpainting. It was not previously known how to make GPs work for these fundamental applications.

- Moreover, unlike typical inpainting approaches, such as patch-based methods (which work by recursively copying pixels or patches into a gap in an image, preserving neighbourhood similarities), GPatt is not restricted to spatial inpainting. This is demonstrated on a video extrapolation example, for which standard inpainting methods would be inapplicable [15]. Similarly, we apply GPatt to perform large-scale long range forecasting of land surface temperatures, through learning a sophisticated correlation structure across space and time. This learned correlation structure also provides insights into the underlying statistical properties of these data.

- We demonstrate that GPatt can precisely recover sophisticated out-of-class kernels automatically.

2   Spectral Mixture Product Kernels for Pattern Discovery

The spectral mixture kernel has recently been introduced [5] to offer a flexible kernel that can learn any stationary kernel. By appealing to Bochner's theorem [16] and building a scale mixture of A

Gaussian pairs in the spectral domain, [5] produced the spectral mixture kernel:

$$k_{\text{SM}}(\tau) = \sum_{a=1}^{A} w_a^2 \exp\{-2\pi^2\tau^2\sigma_a^2\} \cos(2\pi\tau\mu_a), \tag{1}$$

which they applied to a small number of one-dimensional input data. For tractability with multidimensional inputs and large data, we propose a spectral mixture product (SMP) kernel:

$$k_{\text{SMP}}(\tau|\boldsymbol{\theta}) = \prod_{p=1}^{P} k_{\text{SM}}(\tau_p|\boldsymbol{\theta}_p), \tag{2}$$

where τ_p is the pth component of $\tau = x - x' \in \mathbb{R}^P$, $\boldsymbol{\theta}_p$ are the hyperparameters $\{\mu_a, \sigma_a^2, w_a^2\}_{a=1}^{A}$ of the pth spectral mixture kernel in the product of Eq. (2), and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_p\}_{p=1}^{P}$ are the hyperparameters of the SMP kernel. The SMP kernel of Eq. (2) has Kronecker structure which we exploit for scalable and exact inference in section 2.1. With enough components A, the SMP kernel of Eq. (2) can model any stationary product kernel to arbitrary precision, and is flexible even with a small number of components, since scale-location Gaussian mixture models can approximate many spectral densities. We use SMP-A as shorthand for an SMP kernel with A components in each dimension (for a total of $3PA$ kernel hyperparameters and 1 noise hyperparameter).

Critically, a GP with an SMP kernel is not a finite basis function method, but instead corresponds to a finite (A component) mixture of infinite basis function expansions. Therefore such a GP is a truly nonparametric method. This difference between a *truly nonparametric* representation – namely a mixture of infinite bases – and a parametric kernel method, a finite basis expansion corresponding to a *degenerate* GP, is critical both conceptually and practically, as our results will show.

2.1 Fast Exact Inference with Spectral Mixture Product Kernels

Gaussian process inference and learning requires evaluating $(K+\sigma^2 I)^{-1}\mathbf{y}$ and $\log|K+\sigma^2 I|$, for an $N \times N$ covariance matrix K, a vector of N datapoints $\mathbf{y}$, and noise variance σ^2, as described in the supplementary material. For this purpose, it is standard practice to take the Cholesky decomposition of $(K + \sigma^2 I)$ which requires $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ storage, for a dataset of size N. However, many real world applications are engineered for grid structure, including spatial statistics, sensor arrays, image analysis, and time sampling. [14] has shown that the Kronecker structure in product kernels can be exploited for exact inference and hyperparameter learning in $\mathcal{O}(PN^{\frac{2}{P}})$ storage and $\mathcal{O}(PN^{\frac{P+1}{P}})$ operations, so long as the inputs $x \in \mathcal{X}$ are on a multidimensional grid, meaning $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P \subset \mathbb{R}^P$. Details are in the supplement.

Here we relax this grid assumption. Assuming we have a dataset of M observations which are not necessarily on a grid, we propose to form a complete grid using W imaginary observations, $\mathbf{y}_W \sim \mathcal{N}(\mathbf{f}_W, \epsilon^{-1}I_W)$, $\epsilon \to 0$. The total observation vector $\mathbf{y} = [\mathbf{y}_M, \mathbf{y}_W]^\top$ has $N = M + W$ entries: $\mathbf{y} = \mathcal{N}(\mathbf{f}, D_N)$, where the noise covariance matrix $D_N = \text{diag}(D_M, \epsilon^{-1}I_W)$, $D_M = \sigma^2 I_M$. The imaginary observations $\mathbf{y}_W$ have *no corrupting effect* on inference: the moments of the resulting predictive distribution are exactly the same as for the standard predictive distribution, namely $\lim_{\epsilon \to 0}(K_N + D_N)^{-1}\mathbf{y} = (K_M + D_M)^{-1}\mathbf{y}_M$ (proof in the supplement).

We use preconditioned conjugate gradients (PCG) [17] to compute $(K_N + D_N)^{-1}\mathbf{y}$. We use the preconditioning matrix $C = D_N^{-1/2}$ to solve $C^\top(K_N + D_N)C\mathbf{z} = C^\top\mathbf{y}$. The preconditioning matrix C speeds up convergence by ignoring the imaginary observations $\mathbf{y}_W$. Exploiting the fast multiplication of Kronecker matrices, PCG takes $\mathcal{O}(JPN^{\frac{P+1}{P}})$ total operations (where the number of iterations $J \ll N$) to compute $(K_N + D_N)^{-1}\mathbf{y}$ to convergence within machine precision.

For learning (hyperparameter training) we must evaluate the marginal likelihood (supplement). We cannot efficiently decompose $K_M + D_M$ to compute the $\log|K_M + D_M|$ complexity penalty in the marginal likelihood, because K_M is not a Kronecker matrix and D_M is not a scaled identity (as is the usual case for Kronecker decompositions). We approximate the complexity penalty as

$$\log|K_M + D_M| = \sum_{i=1}^{M} \log(\lambda_i^M + \sigma^2) \approx \sum_{i=1}^{M} \log(\tilde{\lambda}_i^M + \sigma^2), \tag{3}$$

3

where σ is the noise standard deviation of the data. We approximate the eigenvalues λ_i^M of K_M using the eigenvalues of K_N such that $\tilde{\lambda}_i^M = \frac{M}{N}\lambda_i^N$ for $i = 1, \ldots, M$, which is particularly effective for large M (e.g. $M > 1000$) [7]. Notably, only the log determinant (complexity penalty) term in the marginal likelihood undergoes a small approximation, and inference remains exact.

All remaining terms in the marginal likelihood can be computed exactlyly and efficiently using PCG. The total runtime cost of hyperparameter learning and exact inference with an incomplete grid is thus $\mathcal{O}(PN^{\frac{P+1}{P}})$. In image problems, for example, $P = 2$, and so the runtime complexity reduces to $\mathcal{O}(N^{1.5})$. Although the proposed inference can handle non-grid data, this inference is most suited to inputs where there is some grid structure – images, video, spatial statistics, etc. If there is no such grid structure (e.g., none of the training data fall onto a grid), then the computational expense necessary to augment the data with imaginary grid observations can be prohibitive.

3 Experiments

In our experiments we combine the SMP kernel of Eq. (2) with the fast exact inference and learning procedures of section 2.1, in a GP method we henceforth call GPatt[1,2].

We contrast GPatt with many alternative Gaussian process kernel methods. We are particularly interested in kernel methods, since they are considered to be general purpose regression methods, but conventionally have difficulty with large scale multidimensional pattern extrapolation. Specifically, we compare to the recent sparse spectrum Gaussian process regression (SSGP) [6] method, which provides fast and flexible kernel learning. SSGP models the kernel spectrum (spectral density) as a sum of point masses, such that SSGP is a finite basis function (*parametric*) model, with as many basis functions as there are spectral point masses. SSGP is similar to the recent models of Le et al. [8] and Rahimi and Recht [9], except it learns the locations of the point masses through marginal likelihood optimization. We use the SSGP implementation provided by the authors at `http://www.tsc.uc3m.es/˜miguel/downloads.php`.

To further test the importance of the fast inference (section 2.1) used in GPatt, we compare to a GP which uses the SMP kernel of section 2 but with the popular fast FITC [10, 18] inference, implemented in GPML (`http://www.gaussianprocess.org/gpml`). We also compare to GPs with the popular squared exponential (SE), rational quadratic (RQ) and Matérn (MA) (with 3 degrees of freedom) kernels, catalogued in Rasmussen and Williams [1], respectively for smooth, multi-scale, and finitely differentiable functions. Since GPs with these kernels cannot scale to the large datasets we consider, we combine these kernels with the same fast inference techniques that we use with GPatt, to enable a comparison.[3] Moreover, we stress test each of these methods in terms of speed and accuracy, as a function of available data and extrapolation range, and number of components. *All* of our experiments contain a large percentage of missing (non-grid) data, and we test accuracy and efficiency as a function of the percentage of missing data.

In all experiments we assume Gaussian noise, to express the marginal likelihood of the data $p(\mathbf{y}|\boldsymbol{\theta})$ solely as a function of kernel hyperparameters $\boldsymbol{\theta}$. To learn $\boldsymbol{\theta}$ we optimize the marginal likelihood using BFGS. We use a simple initialisation scheme: any frequencies $\{\mu_a\}$ are drawn from a uniform distribution from 0 to the Nyquist frequency (1/2 the sampling rate), length-scales $\{1/\sigma_a\}$ from a truncated Gaussian distribution, with mean proportional to the range of the data, and weights $\{w_a\}$ are initialised as the empirical standard deviation of the data divided by the number of components used in the model. In general, we find GPatt is robust to initialisation, particularly for $N > 10^4$ datapoints. We show a representative initialisation in the experiments.

This range of tests allows us to separately understand the effects of the SMP kernel, a non-parametric representation, and the proposed inference methods of section 2.1; we will show that all are required for good extrapolation performance.

---

[1]We write *GPatt-A* when GPatt uses an SMP-A kernel.

[2]Experiments were run on a 64bit PC, with 8GB RAM and a 2.8 GHz Intel i7 processor.

[3]We also considered the model of [19], but this model is intractable for the datasets we considered and is not structured for the fast inference of section 2.1.

3.1 Extrapolating Metal Tread Plate and Pores Patterns

We extrapolate the missing region, shown in Figure 1a, on a real metal tread plate texture. There are 12675 training instances (Figure 1a), and 4225 test instances (Figure 1b). The inputs are pixel locations $x \in \mathbb{R}^2$ ($P = 2$), and the outputs are pixel intensities. The full pattern is shown in Figure 1c. This texture contains shadows and subtle irregularities, no two identical diagonal markings, and patterns that have correlations across both input dimensions.

(a) Train (b) Test (c) Full (d) GPatt (e) SSGP (f) FITC

(g) GP-SE (h) GP-MA (i) GP-RQ (j) GPatt Initialisation

(k) Train (l) GPatt (m) GP-MA (n) Train (o) GPatt (p) GP-MA
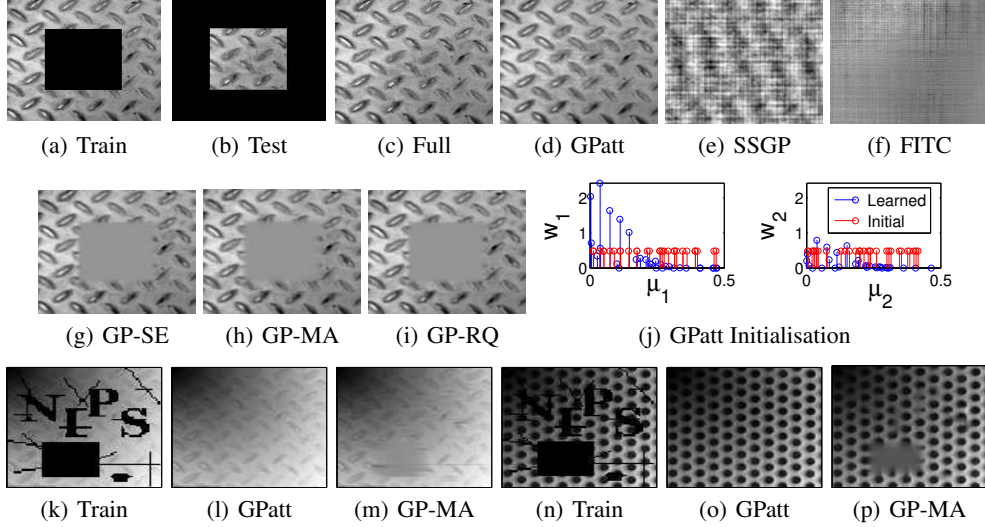
Figure 1: (a)-(j): Extrapolation on a Metal Tread Plate Pattern. Missing data are shown in black. a) Training region (12675 points), b) Testing region (4225 points), c) Full tread plate pattern, d) GPatt-30, e) SSGP with 500 basis functions, f) FITC with 500 inducing (pseudo) inputs, and the SMP-30 kernel, and GPs with the fast exact inference in section 2.1, and g) squared exponential (SE), h) Matérn (MA), and i) rational quadratic (RQ) kernels. j) Initial and learned hyperparameters using GPatt using simple initialisation. During training, weights of extraneous components automatically shrink to zero. (k)-(h) and (n)-(p): Extrapolation on tread plate and pore patterns, respectively, with added artifacts and non-stationary lighting changes.

To reconstruct the missing region, as well as the training region, we use GPatt-30. The GPatt reconstruction shown in Figure 1d is as plausible as the true full pattern shown in Figure 1c, and largely automatic. Without hand crafting of kernel features to suit this image, exposure to similar images, or a sophisticated initialisation procedure, GPatt has automatically discovered the underlying structure of this image, and extrapolated that structure across a large missing region, even though the structure of this pattern is not independent across the two spatial input dimensions. Indeed the separability of the SMP kernel represents only a soft prior assumption, and does not rule out posterior correlations between input dimensions.

The reconstruction in Figure 1e was produced with SSGP, using 500 basis functions. In principle SSGP can model any spectral density (and thus any stationary kernel) with infinitely many components (basis functions). However, since these components are point masses (in frequency space), each component has highly limited expressive power. Moreover, with many components SSGP experiences practical difficulties regarding initialisation, over-fitting, and computation time (scaling quadratically with the number of basis functions). Although SSGP does discover some interesting structure (a diagonal pattern), and has equal training and test performance, it is unable to capture enough information for a convincing reconstruction, and we did not find that more basis functions improved performance. Likewise, FITC with an SMP-30 kernel and 500 inducing (pseudo) inputs cannot capture the necessary information to interpolate or extrapolate. On this example, FITC ran for 2 days, and SSGP-500 for 1 hour, compared to GPatt which took under 5 minutes.

GPs with SE, MA, and RQ kernels are all truly Bayesian nonparametric models – these kernels are derived from infinite basis function expansions. Therefore, as seen in Figure 1 g), h), i), these methods are completely able to capture the information in the training region; however, these kernels do not have the proper structure to reasonably extrapolate across the missing region – they simply

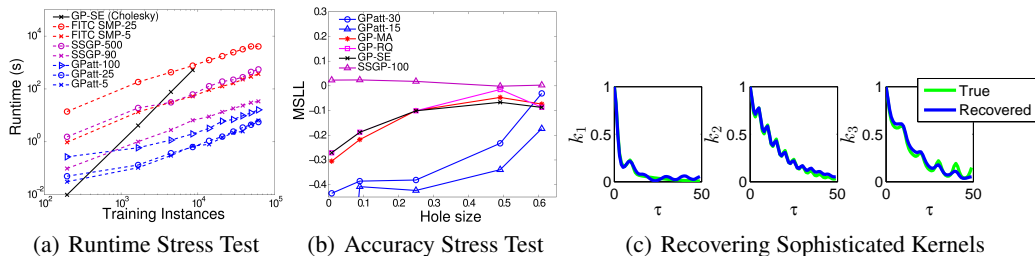(a) Runtime Stress Test   (b) Accuracy Stress Test   (c) Recovering Sophisticated Kernels

Figure 2: Stress Tests. a) **Runtime Stress Test**. We show the runtimes in seconds, as a function of training instances, for evaluating the log marginal likelihood, and any relevant derivatives, for a standard GP with SE kernel (as implemented in GPML), FITC with 500 inducing (pseudo) inputs and SMP-25 and SMP-5 kernels, SSGP with 90 and 500 basis functions, and GPatt-100, GPatt-25, and GPatt-5. Runtimes are for a 64bit PC, with 8GB RAM and a 2.8 GHz Intel i7 processor, on the cone pattern ($P = 2$), shown in the supplement. The ratio of training inputs to the sum of imaginary and training inputs for GPatt is 0.4 and 0.6 for the smallest two training sizes, and 0.7 for all other training sets. b) **Accuracy Stress Test**. MSLL as a function of holesize on the metal pattern of Figure 1. The values on the horizontal axis represent the fraction of missing (testing) data from the full pattern (for comparison Fig 1a has 25% missing data). We compare GPatt-30 and GPatt-15 with GPs with SE, MA, and RQ kernels (and the inference of section 2.1), and SSGP with 100 basis functions. The MSLL for GPatt-15 at a holesize of 0.01 is -1.5886. c) **Recovering Sophisticated Kernels**. A product of three kernels (shown in green) was used to generate a movie of 112,500 training points. From this data, GPatt-20 reconstructs these component kernels (the learned SMP-20 kernel is shown in blue). All kernels are a function of $\tau = x - x'$. has been scaled by $k(0)$.

act as smoothing filters. We note that this comparison is only possible because these GPs are using the fast exact inference techniques in section 2.1.

Overall, these results indicate that both expressive nonparametric kernels, such as the SMP kernel, and the specific fast inference in section 2.1, are needed to extrapolate patterns in these images.

We note that the SMP-30 kernel used with GPatt has more components than needed for this problem. However, as shown in Fig. 1j, if the model is overspecified, the complexity penalty in the marginal likelihood shrinks the weights ($\{w_a\}$ in Eq. (1)) of extraneous components, as a proxy for model selection – an effect similar to *automatic relevance determination* [20]. Components which do not significantly contribute to model fit will therefore be automatically pruned, as shrinking the weights decreases the eigenvalues of K and thus minimizes the complexity penalty (a sum of log eigenvalues). This simple GPatt initialisation procedure was used for the results in all experiments and is especially effective for $N > 10^4$ points.

In Figure 1 (k)-(h) and (n)-(p) we use GPatt to extrapolate on treadplate and pore patterns with added artifacts and lighting changes. GPatt still provides a convincing extrapolation – able to uncover both local and global structure. Alternative GPs with the inference of section 2.1 can interpolate small artifacts quite accurately, but have trouble with larger missing regions.

3.2 Stress Tests and Recovering Complex 3D Kernels from Video

We stress test GPatt and alternative methods in terms of speed and accuracy, with varying datasizes, extrapolation ranges, basis functions, inducing (pseudo) inputs, and components. We assess accuracy using standardised mean square error (SMSE) and mean standardized log loss (MSLL) (a scaled negative log likelihood), as defined in Rasmussen and Williams [1] on page 23. Using the empirical mean and variance to fit the data would give an SMSE and MSLL of 1 and 0 respectively. Smaller SMSE and more negative MSLL values correspond to better fits of the data.

The runtime stress test in Figure 2a shows that the number of components used in GPatt does not significantly affect runtime, and that GPatt is much faster than FITC (using 500 inducing inputs) and SSGP (using 90 or 500 basis functions), even with 100 components (601 kernel hyperparameters). The slope of each curve roughly indicates the asymptotic scaling of each method. In this experiment, the standard GP (with SE kernel) has a slope of 2.9, which is close to the cubic scaling we expect. All

Table 1: We compare the test performance of GPatt-30 with SSGP (using 100 basis functions), and GPs using squared exponential (SE), Matérn (MA), and rational quadratic (RQ) kernels, combined with the inference of section 3.2, on patterns with a train test split as in the metal treadplate pattern of Figure 1. We show the results as SMSE (MSLL).

train, test	Rubber mat 12675, 4225	Tread plate 12675, 4225	Pores 12675, 4225	Wood 14259, 4941	Chain mail 14101, 4779
GPatt	**0.31 (-0.57)**	**0.45 (-0.38)**	**0.0038 (-2.8)**	**0.015 (-1.4)**	**0.79 (-0.052)**
SSGP	0.65 (-0.21)	1.06 (0.018)	1.04 (-0.024)	0.19 (-0.80)	1.1 (0.036)
SE	0.97 (0.14)	0.90 (-0.10)	0.89 (-0.21)	0.64 (1.6)	1.1 (1.6)
MA	0.86 (-0.069)	0.88 (-0.10)	0.88 (-0.24)	0.43 (1.6)	0.99 (0.26)
RQ	0.89 (0.039)	0.90 (-0.10)	0.88 (-0.048)	0.077 (0.77)	0.97 (-0.0025)

other curves have a slope of 1 ± 0.1, indicating linear scaling with the number of training instances. However, FITC and SSGP are used here with a *fixed* number of inducing inputs and basis functions. More inducing inputs and basis functions should be used when there are more training instances – and these methods scale quadratically with inducing inputs and basis functions for a fixed number of training instances. GPatt, on the other hand, can scale linearly in runtime as a function of training size, without any deterioration in performance. Furthermore, the fixed 2-3 orders of magnitude GPatt outperforms alternatives are as practically important as asymptotic scaling.

The accuracy stress test in Figure 2b shows extrapolation (MSLL) performance on the metal tread plate pattern of Figure 1c with varying holesizes, running from 0% to 60% missing data for testing (for comparison the hole in Fig 1a has 25% missing data). GPs with SE, RQ, and MA kernels (and the fast inference of section 2.1) all steadily increase in error as a function of holesize. Conversely, SSGP does not increase in error as a function of holesize – with finite basis functions SSGP cannot extract as much information from larger datasets as the alternatives. GPatt performs well relative to the other methods, even with a small number of components. GPatt is particularly able to exploit the extra information in additional training instances: only when the holesize is so large that over 60% of the data are missing does GPatt's performance degrade to the same level as alternative methods.

In Table 1 we compare the test performance of GPatt with SSGP, and GPs using SE, MA, and RQ kernels, for extrapolating five different patterns, with the same train test split as for the tread plate pattern in Figure 1. All patterns are shown in the supplement. GPatt consistently has the lowest SMSE and MSLL. Note that many of these datasets are sophisticated patterns, containing intricate details which are not strictly periodic, such as lighting irregularities, metal impurities, etc. Indeed SSGP has a periodic kernel (unlike the SMP kernel which is not strictly periodic), and is capable of modelling multiple periodic components, but does not perform as well as GPatt on these examples.

We also consider a particularly large example, where we use GPatt-10 to perform learning and exact inference on the *Pores* pattern, with 383,400 training points, to extrapolate a large missing region with 96,600 test points. The SMSE is 0.077, and the total runtime was 2800 seconds. Images of the successful extrapolation are shown in the supplement.

We end this section by showing that GPatt can accurately recover a wide range of kernels, even using a small number of components. To test GPatt's ability to recover ground truth kernels, we simulate a $50 \times 50 \times 50$ movie of data (e.g. two spatial input dimensions, one temporal) using a GP with kernel $k = k_1 k_2 k_3$ (each component kernel in this product operates on a different input dimension), where $k_1 = k_{\text{SE}} + k_{\text{SE}} \times k_{\text{PER}}$, $k_2 = k_{\text{MA}} \times k_{\text{PER}} + k_{\text{MA}} \times k_{\text{PER}}$, and $k_3 = (k_{\text{RQ}} + k_{\text{PER}}) \times k_{\text{PER}} + k_{\text{SE}}$. $(k_{\text{PER}}(\tau) = \exp[-2\sin^2(\pi\,\tau\,\omega)/\ell^2], \tau = x - x')$. We use 5 consecutive 50×50 slices for testing, leaving a large number $N = 112500$ of training points, providing much information to learn the true generating kernels. Moreover, GPatt-20 reconstructs these complex out of class kernels in under 10 minutes, as shown in Fig 2c. In the supplement, we show true and predicted frames from the movie.

3.3 Wallpaper and Scene Reconstruction and Long Range Temperature Forecasting

Although GPatt is a general purpose regression method, it can also be used for inpainting: image restoration, object removal, etc. We first consider a wallpaper image stained by a black apple mark, shown in Figure 3. To remove the stain, we apply a mask and then separate the image into its three channels (red, green, and blue), resulting in 15047 pixels in each channel for training. In each
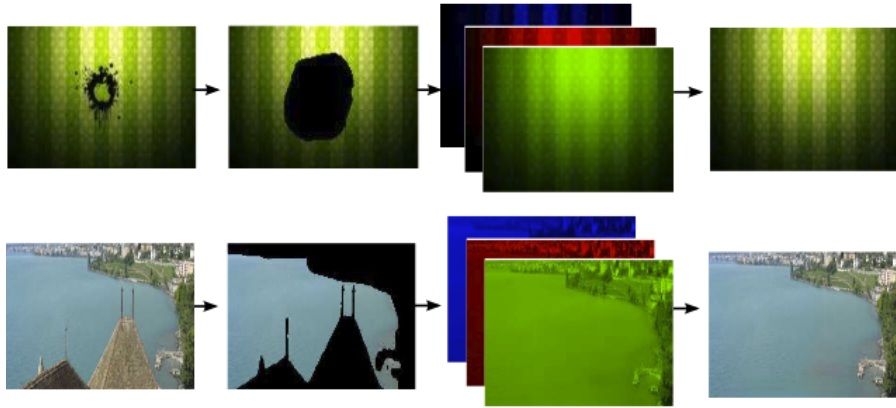
(a) Inpainting
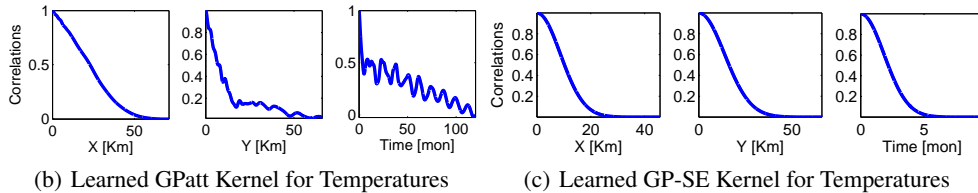
(b) Learned GPatt Kernel for Temperatures   (c) Learned GP-SE Kernel for Temperatures

Figure 3: a) Image inpainting with GPatt. From left to right: A mask is applied to the original image, GPatt extrapolates the mask region in each of the three (red, blue, green) image channels, and the results are joined to produce the restored image. Top row: Removing a stain (train: 15047×3). Bottom row: Removing a rooftop to restore a natural scene (train: 32269×3). We do not extrapolate the coast. (b)-(c): Kernels learned for land surface temperatures using GPatt and GP-SE.

channel we ran GPatt using SMP-30. We then combined the results from each channel to restore the image without any stain, which is impressive given the subtleties in the pattern and lighting.

In our next example, we wish to reconstruct a natural scene obscured by a prominent rooftop, shown in the second row of Figure 3a). By applying a mask, and following the same procedure as for the stain, this time with 32269 pixels in each channel for training, GPatt reconstructs the scene without the rooftop. This reconstruction captures subtle details, such as waves, with only a single training image. In fact this example has been used with inpainting algorithms which were given access to a repository of thousands of similar images [21]. The results emphasized that conventional inpainting algorithms and GPatt have profoundly different objectives, which are sometimes even at cross purposes: inpainting attempts to make the image look good to a human (e.g., the example in [21] placed boats in the water), while GPatt is a general purpose regression algorithm, which simply aims to make accurate predictions at test input locations, from training data alone. For example, GPatt can naturally learn temporal correlations to make predictions in the video example of section 3.2, for which standard patch based inpainting methods would be inapplicable [15].

Similarly, we use GPatt to perform long range forecasting of land surface temperatures. After training on 108 months (9 years) of temperature data across North America (299,268 training points; a $71 \times 66 \times 108$ grid, with missing data for water), we forecast 12 months (1 year) ahead (33,252 testing points). The runtime was under 30 minutes. The learned kernels using GPatt and GP-SE are shown in Figure 3 b) and c). The learned kernels for GPatt are highly non-standard – both quasi periodic and heavy tailed. These learned correlation patterns provide insights into features (such as seasonal influences) which affect how temperatures are varying in space and time. Indeed learning the kernel allows us to discover fundamental properties of the data. The temperature forecasts using GPatt and GP-SE, superimposed on maps of North America, are shown in the supplement.

4 Discussion

Large scale multidimensional pattern extrapolation problems are of fundamental importance in machine learning, where we wish to develop scalable models which can make impressive generalisa-

tions. However, there are many obstacles towards applying popular kernel methods, such as Gaussian processes, to these fundamental problems. We have shown that a combination of expressive kernels, truly Bayesian nonparametric representations, and inference which exploits model structure, can distinctly enable a kernel approach to these problems. Moreover, there is much promise in further exploring Bayesian nonparametric kernel methods for large scale pattern extrapolation. Such methods can be extremely expressive, and expressive methods are most needed for large scale problems, which provide relatively more information for automatically learning a rich statistical representation of the data.

References

[1] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for Machine Learning*. The MIT Press, 2006.

[2] C.E. Rasmussen. *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*. PhD thesis, University of Toronto, 1996.

[3] Anthony O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society*, B(40):1–42, 1978.

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

[5] Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process kernels for pattern discovery and extrapolation. *International Conference on Machine Learning*, 2013.

[6] M. Lázaro-Gredilla, J. Quiñonero-Candela, C.E. Rasmussen, and A.R. Figueiras-Vidal. Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[7] C Williams and M Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688. MIT Press, 2001.

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

[9] A Rahimi and B Recht. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, 2007.

[10] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, volume 18, page 1257. MIT Press, 2006.

[11] J Hensman, N Fusi, and N.D. Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 2013.

[12] M. Seeger, C.K.I. Williams, and N.D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Workshop on AI and Statistics*, volume 9, page 2003, 2003.

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

[14] Y Saatchi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011.

[15] Christine Guillemot and Olivier Le Meur. Image inpainting: Overview and recent advances. *Signal Processing Magazine, IEEE*, 31(1):127–144, 2014.

[16] S Bochner. *Lectures on Fourier Integrals.(AM-42)*, volume 42. Princeton University Press, 1959.

[17] Kendall E Atkinson. *An introduction to numerical analysis*. John Wiley & Sons, 2008.

[18] A Naish-Guzman and S Holden. The generalized fitc approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1064, 2007.

[19] D. Duvenaud, J.R. Lloyd, R. Grosse, J.B. Tenenbaum, and Z. Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, 2013.

[20] D.J.C MacKay. Bayesian nonlinear modeling for the prediction competition. *Ashrae Transactions*, 100(2):1053–1062, 1994.

[21] J Hays and A Efros. Scene completion using millions of photographs. *Communications of the ACM*, 51(10):87–94, 2008.

Supplementary Material: Fast Kernel Learning for Multidimensional Pattern Extrapolation

Andrew Gordon Wilson*, Elad Gilboa*, Arye Nehorai, John P. Cunningham

1 Introduction

We begin with background on Gaussian processes. We provide further detail about the eigendecomposition of kronecker matrices, and the runtime complexity of kronecker matrix vector products. We then provide images of the temperature forecasts We also provide spectral images of the learned kernels in the metal tread plate experiment, larger versions of the images in Table 1, images of the extrapolation results on the large pore example, and images of the GPatt reconstruction for several consecutive movie frames. We also enlarge some of the results in the main text.

2 Gaussian Processes

A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution. Using a Gaussian process, we can define a distribution over functions $f(x)$,

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')). \tag{1}$$

The mean function $m(x)$ and covariance kernel $k(x, x')$ are defined as

$$m(x) = \mathbb{E}[f(x)], \tag{2}$$

$$k(x, x') = \mathrm{cov}(f(x), f(x')), \tag{3}$$

where x and x' are any pair of inputs in $\mathbb{R}^P$. Any collection of function values has a joint Gaussian distribution,

$$[f(x_1), \ldots, f(x_N)] \sim \mathcal{N}(\boldsymbol{\mu}, K), \tag{4}$$

with mean vector $\boldsymbol{\mu}_i = m(x_i)$ and $N \times N$ covariance matrix $K_{ij} = k(x_i, x_j)$.

Assuming Gaussian noise, e.g. observations $y(x) = f(x) + \epsilon(x)$, $\epsilon(x) = \mathcal{N}(0, \sigma^2)$, the predictive distribution for $f(x_*)$ at a test input x_*, conditioned on $\mathbf{y} = (y(x_1), \ldots, y(x_N))^\top$ at training inputs $X = (x_1, \ldots, x_n)^\top$, is analytic and given by:

$$f(x_*)|x_*, X, \mathbf{y} \sim \mathcal{N}(\bar{f}_*, \mathbf{V}[f_*]) \tag{5}$$

$$\bar{f}_* = \mathbf{k}_*^\top (K + \sigma^2 I)^{-1} \mathbf{y} \tag{6}$$

$$\mathbf{V}[f_*] = k(x_*, x_*) - \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}_*, \tag{7}$$

where the $N \times 1$ vector $\mathbf{k}_*$ has entries $(\mathbf{k}_*)_i = k(x_*, x_i)$.

The Gaussian process $f(x)$ can also be analytically marginalised to obtain the likelihood of the data, conditioned only on the hyperparameters $\boldsymbol{\theta}$ of the kernel:

$$\log p(\mathbf{y}|\boldsymbol{\theta}) \propto -[\overbrace{\mathbf{y}^\top (K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1} \mathbf{y}}^{\text{model fit}} + \overbrace{\log |K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}}]. \tag{8}$$

This *marginal likelihood* in Eq. (8) pleasingly compartmentalises into automatically calibrated model fit and complexity terms [1], and can be optimized to learn the kernel hyperparameters $\boldsymbol{\theta}$,

or used to integrate out $\boldsymbol{\theta}$ using MCMC [2]. The problem of model selection and learning in Gaussian processes is "exactly the problem of finding suitable properties for the covariance function. Note that this gives us a model of the data, and characteristics (such as smoothness, length-scale, etc.) which we can interpret." [3].

The popular *squared exponential* (SE) kernel has the form

$$k_{\mathrm{SE}}(x, x') = \exp(-0.5||x - x'||^2/\ell^2).\tag{9}$$

GPs with SE kernels are smoothing devices, only able to learn how quickly sample functions vary with inputs x, through the length-scale parameter ℓ.

3 Eigendecomposition of Kronecker Matrices

Assuming a product kernel,

$$k(x_i, x_j) = \prod_{p=1}^{P} k^p(x_i^p, x_j^p),\tag{10}$$

and inputs $x \in \mathcal{X}$ on a multidimensional grid, $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P \subset \mathbb{R}^P$, then the covariance matrix K decomposes into a Kronecker product of matrices over each input dimension $K = K^1 \otimes \cdots \otimes K^P$ [4]. The eigendecomposition of K into $QVQ^\top$ similarly decomposes: $Q = Q^1 \otimes \cdots \otimes Q^P$ and $V = V^1 \otimes \cdots \otimes V^P$. Each covariance matrix K^p in the Kronecker product has entries $K_{ij}^p = k^p(x_i^p, x_j^p)$ and decomposes as $K^p = Q^p V^p Q^{p\top}$. Thus the $N \times N$ covariance matrix K can be stored in $\mathcal{O}(PN^{\frac{2}{P}})$ and decomposed into $QVQ^\top$ in $\mathcal{O}(PN^{\frac{3}{P}})$ operations, for N datapoints and P input dimensions. [1] Moreover, the product of Kronecker matrices such as K, Q, or their inverses, with a vector $\mathbf{u}$, can be performed in $\mathcal{O}(PN^{\frac{P+1}{P}})$ operations (section 4).

Given the eigendecomposition of K as $QVQ^\top$, we can re-write $(K + \sigma^2 I)^{-1}\mathbf{y}$ and $\log|K + \sigma^2 I|$ (required for GP inference and learning as described in the supplement)

$$(K + \sigma^2 I)^{-1}\mathbf{y} = (QVQ^\top + \sigma^2 I)^{-1}\mathbf{y}\tag{11}$$

$$= Q(V + \sigma^2 I)^{-1}Q^\top \mathbf{y},\tag{12}$$

and

$$\log|K + \sigma^2 I| = \log|QVQ^\top + \sigma^2 I| = \sum_{i=1}^{N} \log(\lambda_i + \sigma^2),\tag{13}$$

where λ_i are the eigenvalues of K, which can be computed in $\mathcal{O}(PN^{\frac{3}{P}})$.

Thus we can evaluate the predictive distribution and marginal likelihood to perform *exact* inference and hyperparameter learning, with $\mathcal{O}(PN^{\frac{2}{P}})$ storage and $\mathcal{O}(PN^{\frac{P+1}{P}})$ operations.

4 Matrix-vector Product for Kronecker Matrices

We first define a few operators from standard Kronecker literature. Let $\mathbf{B}$ be a matrix of size $p \times q$. The reshape$(\mathbf{B}, r, c)$ operator returns a r-by-c matrix ($rc = pq$) whose elements are taken column-wise from $\mathbf{B}$. The vec$(\cdot)$ operator stacks the matrix columns onto a single vector, vec$(\mathbf{B}) = $ reshape$(\mathbf{B}, pq, 1)$, and the vec$^{-1}(\cdot)$ operator is defined as vec$^{-1}(\text{vec}(\mathbf{B})) = \mathbf{B}$. Finally, using the standard Kronecker property $(\mathbf{B} \otimes \mathbf{C})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{C}\mathbf{X}\mathbf{B}^\top)$, we note that for any N argument vector $\mathbf{u} \in \mathbb{R}^N$ we have

$$\mathbf{K}_N \mathbf{u} = \left(\bigotimes_{p=1}^{P} \mathbf{K}_{N^{1/P}}^p\right) \mathbf{u} = \text{vec}\left(\mathbf{K}_{N^{1/P}}^P \mathbf{U} \left(\bigotimes_{p=1}^{P-1} \mathbf{K}_{N^{1/P}}^p\right)^\top\right),\tag{14}$$

---

[1]The total number of datapoints $N = \prod_p |\mathcal{X}_p|$, where $|\mathcal{X}_p|$ is the cardinality of $\mathcal{X}_p$. For clarity of presentation, we assume each $|\mathcal{X}_p|$ has equal cardinality $N^{1/P}$.

where $\mathbf{U} = \text{reshape}(\mathbf{u}, N^{1/P}, N^{\frac{P-1}{P}})$, and $\mathbf{K}_N$ is an $N \times N$ Kronecker matrix. With no change to Eq. (14) we can introduce the $\text{vec}^{-1}(\text{vec}(\cdot))$ operators to get

$$\mathbf{K}_N \mathbf{u} = \text{vec}\left(\left(\ \text{vec}^{-1}\left(\text{vec}\left(\ \left(\bigotimes_{p=1}^{P-1} \mathbf{K}_{N^{1/P}}^p\right)\left(\mathbf{K}_{N^{1/P}}^P \mathbf{U}\right)^\top\ \right)\right)\ \right)^\top\right). \quad (15)$$

The inner component of Eq. (15) can be written as

$$\text{vec}\left(\left(\bigotimes_{p=1}^{P-1} \mathbf{K}_{N^{1/P}}^p\right)\left(\mathbf{K}_{N^{1/P}}^P \mathbf{U}\right)^\top \mathbf{I}_{N^{1/P}}\right) = \mathbf{I}_{N^{1/P}} \otimes \left(\bigotimes_{p=1}^{P-1} \mathbf{K}_{N^{1/P}}^p\right)\text{vec}\left(\left(\mathbf{K}_{N^{1/P}}^P \mathbf{U}\right)^\top\right). \quad (16)$$

Notice that Eq. (16) is in the same form as Eq. (14) (Kronecker matrix-vector product). By repeating Eqs. (15-16) over all P dimensions, and noting that $\left(\bigotimes_{p=1}^P \mathbf{I}_{N^{1/P}}\right)\mathbf{u} = \mathbf{u}$, we see that the original matrix-vector product can be written as

$$\left(\bigotimes_{p=1}^P \mathbf{K}_{N^{1/P}}^p\right)\mathbf{u} = \text{vec}\left(\left[\mathbf{K}_{N^{1/P}}^1, \ldots \left[\mathbf{K}_{N^{1/P}}^{P-1}, \left[\mathbf{K}_{N^{1/P}}^P, \mathbf{U}\right]\right]\right]\right) \quad (17)$$

$$\overset{\text{def}}{=} \text{kron\_mvprod}\left(\mathbf{K}_{N^{1/P}}^1, \mathbf{K}_{N^{1/P}}^2, \ldots, \mathbf{K}_{N^{1/P}}^P, \mathbf{u}\right) \quad (18)$$

where the bracket notation denotes matrix product, transpose then reshape, i.e.,

$$\left[\mathbf{K}_{N^{1/P}}^p, \mathbf{U}\right] = \text{reshape}\left(\left(\mathbf{K}_{N^{1/P}}^p \mathbf{U}\right)^\top, N^{1/P}, N^{\frac{P-1}{P}}\right). \quad (19)$$

Iteratively solving the kron_mvprod operator in Eq. (18) requires $(PN^{\frac{P+1}{P}})$, because each of the P bracket operations requires $\mathcal{O}(N^{\frac{P+1}{P}})$.

4.1 Inference with Missing Observations

The predictive mean of a Gaussian process at L test points, given N training points, is given by

$$\boldsymbol{\mu}_L = \mathbf{K}_{LN}\left(\mathbf{K}_N + \sigma^2 I_N\right)^{-1})\mathbf{y}, \quad (20)$$

where $\mathbf{K}_{LN}$ is an $L \times N$ matrix of cross covariances between the test and training points. We wish to show that when we have M observations which are not on a grid that the desired predictive mean

$$\boldsymbol{\mu}_L = \mathbf{K}_{LM}\left(\mathbf{K}_M + \sigma^2 I_M\right)^{-1}\mathbf{y}_M = \mathbf{K}_{LN}\left(\mathbf{K}_N + \mathbf{D}_N\right)^{-1}\mathbf{y}, \quad (21)$$

where $\mathbf{y} = [\mathbf{y}_M, \mathbf{y}_W]^\top$ includes imaginary observations $\mathbf{y}_W$, and D_N is as defined in Section 4. as

$$D_N = \begin{bmatrix} D_M & 0 \\ 0 & \epsilon^{-1} I_W \end{bmatrix}, \quad (22)$$

where we set $D_M = \sigma^2 I_M$.

Starting with the right hand side of Eq. (21),

$$\boldsymbol{\mu}_L = \begin{bmatrix} \mathbf{K}_{LM} \\ \mathbf{K}_{LW} \end{bmatrix}\begin{bmatrix} \mathbf{K}_M + \mathbf{D}_M & \mathbf{K}_{MW} \\ \mathbf{K}_{MW}^\top & \mathbf{K}_W + \epsilon^{-1}\mathbf{I}_W \end{bmatrix}^{-1}\begin{bmatrix} \mathbf{y}_M \\ \mathbf{y}_W \end{bmatrix}. \quad (23)$$

Using the block matrix inversion theorem, we get

$$\begin{bmatrix} A & B \\ C & E \end{bmatrix}^{-1} = \begin{bmatrix} (A - BE^{-1}C)^{-1} & -A^{-1}B(I - E^{-1}CA^{-1}B)^{-1}E^{-1} \\ -E^{-1}C(A - BE^{-1}C)^{-1} & (I - E^{-1}CA^{-1}B)^{-1}E^{-1} \end{bmatrix}, \quad (24)$$

where $A = \mathbf{K}_M + \mathbf{D}_M$, $B = \mathbf{K}_{MW}$, $C = \mathbf{K}_{MW}^\top$, and $E = \mathbf{K}_W + \epsilon^{-1}\mathbf{I}_W$. If we take the limit of $E^{-1} = \epsilon(\epsilon\mathbf{K}_W + \mathbf{I}_W)^{-1} \overset{\epsilon \to 0}{\longrightarrow} \mathbf{0}$, and solve for the other components, Eq. (23) becomes

$$\boldsymbol{\mu}_L = \begin{bmatrix} \mathbf{K}_{LM} \\ \mathbf{K}_{LW} \end{bmatrix}\begin{bmatrix} (\mathbf{K}_M + \mathbf{D}_M)^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \mathbf{y}_M \\ \mathbf{y}_W \end{bmatrix} = \mathbf{K}_{LM}(\mathbf{K}_M + \mathbf{D}_M)^{-1}\mathbf{y}_M \quad (25)$$

which is the exact GP result. In other words, performing inference given observations $\mathbf{y}$ will give the same result as directly using observations $\mathbf{y}_M$. The proof that the predictive covariances remain unchanged proceeds similarly.

5   Land Temperature Forecasts

Figure 1 shows 12 month ahead forecasts for land surface temperatures using GPatt. We can see that GPatt has learned a representation of the training data and has made sensible long range extrapolations. The forecasts of GP-SE, with the popular squared exponential covariance function, quickly lose any relation with the training data.

6   Spectrum Analysis

We can gain further insight into the behavior of GPatt by looking at the spectral density learned by the spectral mixture kernel. Figure 2 shows the log spectrum representations of the learned kernels from Section 5.1. Smoothers, such as the popular kernels SE, RQ, and MA, concentrate their spectral energy around the origin, differing only by their support for higher frequencies. Methods which used the SMP kernel, such as the GPatt and FITC (with an SMP kernel), are able to learn meaningful features in the spectrum space.

7   Enlarged Inpainting Image

## 8   Tread Plate, Stress Test, and Video Images

Figure 4 illustrates the images used for the stress tests. In Figure 5, we provide the results for the large pore example. Finally, Figure 6 shows the true and predicted movie frames.

References

[1] C.E. Rasmussen and Z. Ghahramani. Occam's razor. In *Neural Information Process Systems*, 2001.

[2] I. Murray and R.P. Adams. Slice sampling covariance hyperparameters in latent Gaussian models. In *Advances in Neural Information Processing Systems*, 2010.

[3] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for Machine Learning*. The MIT Press, 2006.

[4] Y Saatchi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011.
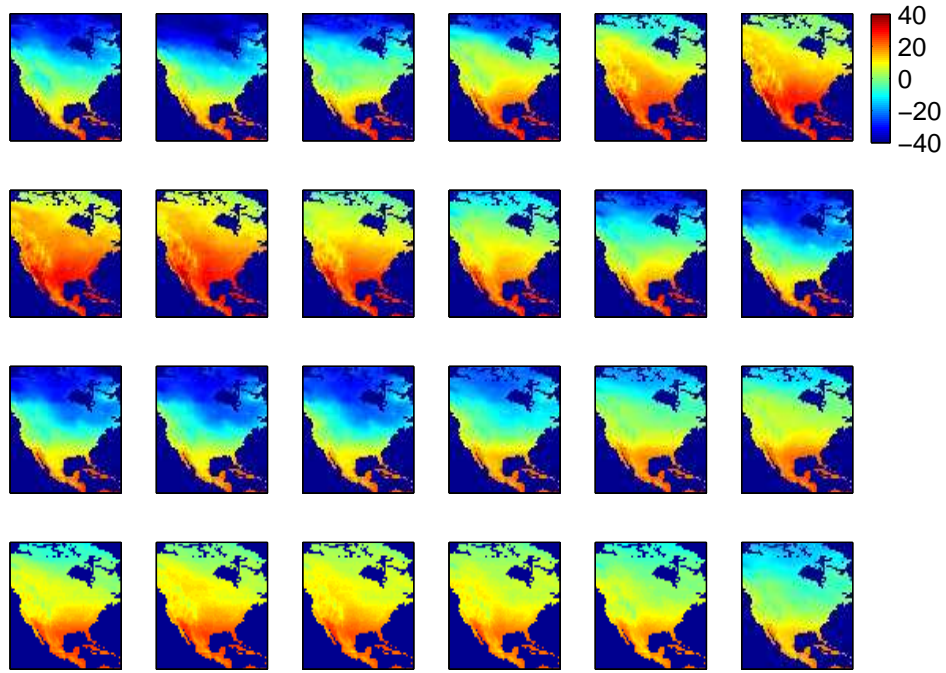
(a) GPatt
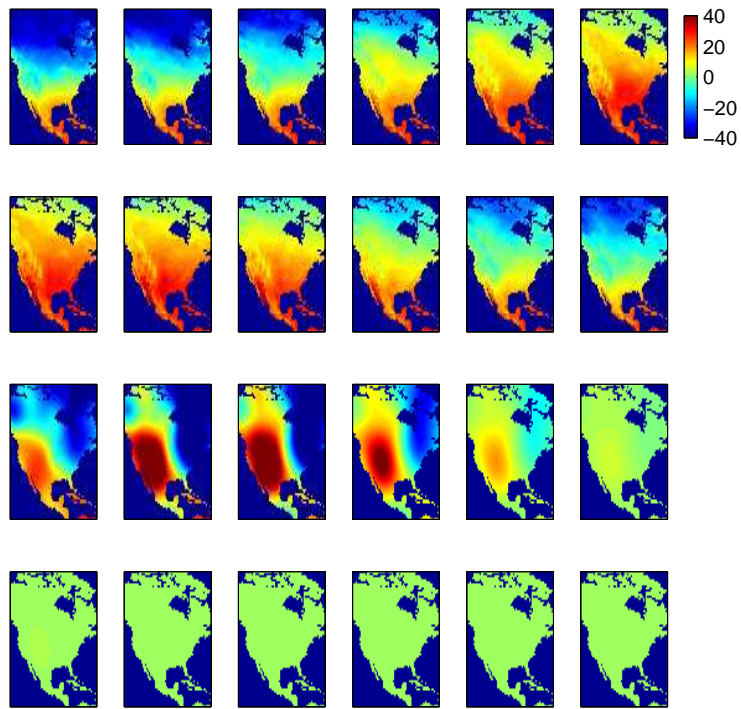
(b) GP-SE

Figure 1: In each image, the first two rows are the last 12 months of training data, and the last two rows are 12 month forecasts. Note that this is a true extrapolation problem: all 12 months are forecast at once (this is not a rolling forecast). a) GPatt, b) GP-SE.
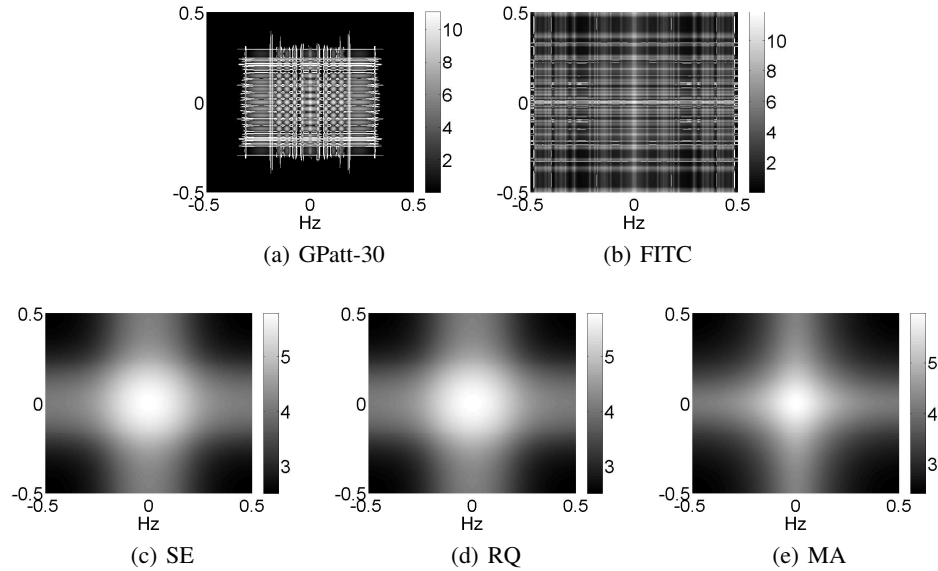
5

(a) GPatt-30   (b) FITC

(c) SE   (d) RQ   (e) MA

Figure 2: Spectral representation of the learned kernels from Section 5.1. For methods which used the SMP kernel (namely, a) GPatt and b) FITC) we plot the analytical log spectrum using the learned hyperparameters. For c)Squared exponential, d) Rational quadratic, and e) Matérn-3 we plot instead the empirical log spectrum using the Fast Fourier transform of the kernel.
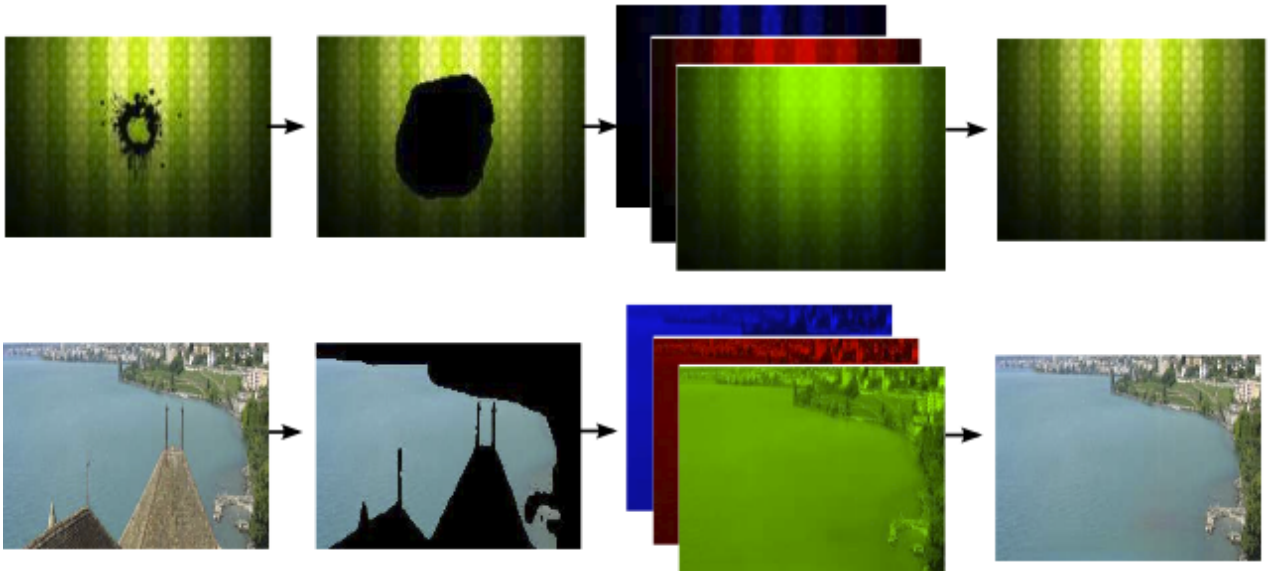

Figure 3: Image inpainting with GPatt. From left to right: A mask is applied to the original image, GPatt extrapolates the mask region in each of the three (red, blue, green) image channels, and the results are joined to produce the restored image. Top row: Removing a stain (train: 15047×3). Bottom row: Removing a rooftop to restore a natural scene (train: 32269×3). We do not attempt to extrapolate the coast, which is masked during training.

6

(a) Rubber mat   (b) Tread plate   (c) Pores
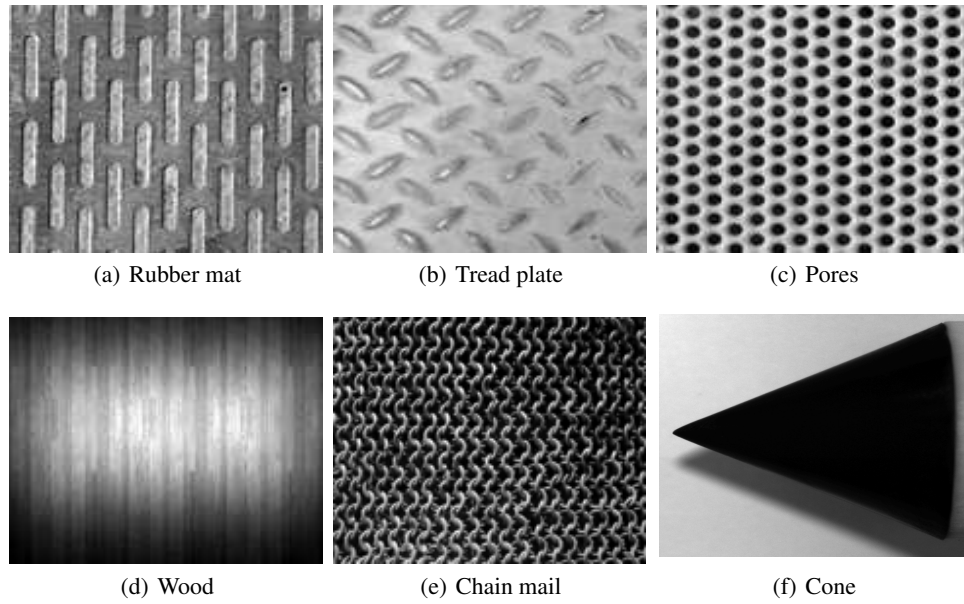
(d) Wood   (e) Chain mail   (f) Cone

Figure 4: Images used for stress tests in Section 5.2. Figures a) through e) show the textures used in the accuracy comparison of Table 1. Figure e) is the cone image which was used for the runtime analysis shown in Figure 3a.
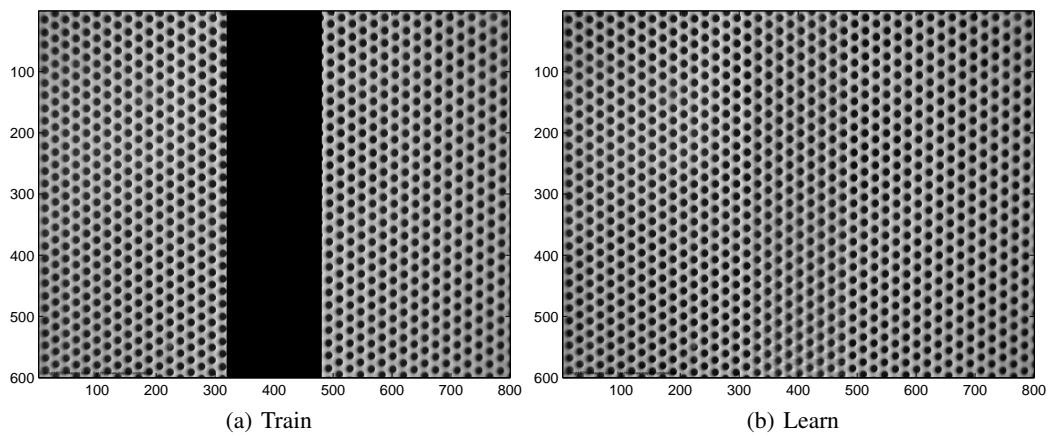

(a) Train   (b) Learn

Figure 5: GPatt on a particularly large multidimensional dataset. a) Training region (383400 points), b) GPatt-10 reconstruction of the missing region.
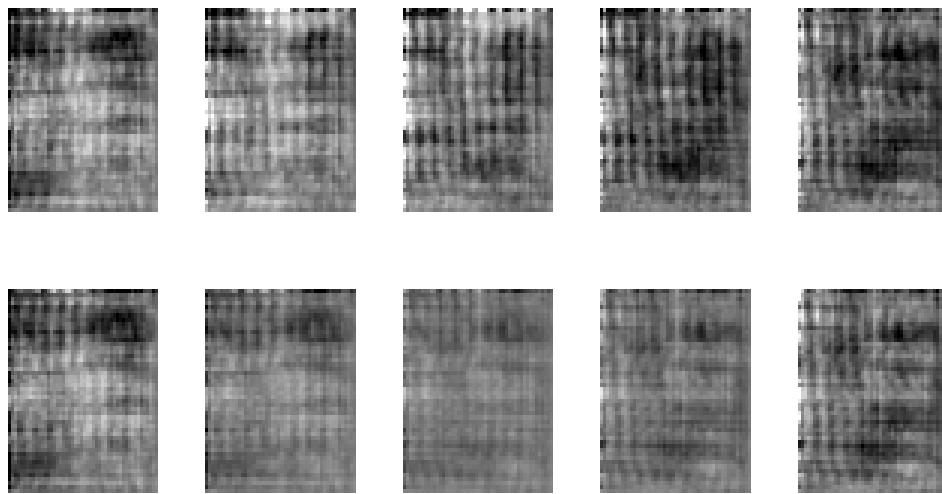
Figure 6: Recovering 5 consecutive slices from a movie. All 5 slices are missing during training: this is not one step ahead forecasting. (Top row: true slices take from the middle of the movie. Bottom row: predicted slices using GPatt-20.