# Random Fourier Series Features

Fabian Damken
fabian.damken@stud.tu-darmstadt.de

Joe Watson
joe.watson@tu-darmstadt.de

Jan Peters
jan.peters@tu-darmstadt.de

***Abstract*—Neural networks (NNs) are powerful prediction models used in various domains such as robotics control. However, they lack a principled approach for uncertainty quantification. While research has been conducted towards Bayesian neural networks (BNNs), Gaussian processes (GPs) remain the go-to approach for reliable uncertainty estimation, capturing both aleatoric and epistemic uncertainty. This weakness is due to the intractability of inference on BNNs and a lack of principled and scalable approximate inference methods. However, the success of a GP highly depends on the kernel *design*, i.e., the kernel is not learned from data compared to a NN. We propose random Fourier series features (RFSFs), an extension of the well-known random Fourier features (RFFs) to bridge the gap between the data-driven features of NNs and the powerful inductive biases and practical inference of GPs.**

## I. Introduction

(Deep) neural networks (NNs) are extremely powerful and expressive machine learning models dominating the current landscape of artificial intelligence [1]. While they have great predictive power, NNs usually lack uncertainty estimation. In Bayesian machine learning, we not only seek models that predict some value, but that also gauge their uncertainty about the prediction. While frequentistic models are capable of estimating aleatoric (i.e., noise-induced) uncertainty, Bayesian models include epistemic uncertainty that quantifies the model's trust "in itself." Quantification of the uncertainty is useful in a variety of domains such as model-predictive control (MPC) [2], [3].

A proposed class of extensions for NNs are Bayesian neural networks (BNNs) [4], [5] combining the expressiveness of NNs with uncertainty quantification. However, exact inference is intractable, so these methods have to resort to approximate inference approaches [5]–[10]. Also, BNNs suffer from various drawbacks such as expensive and complicated training, inaccurate posteriors, and unreliable uncertainty quantification [11]–[16]. Hence, they are not suitable for application in high-stakes domains such as medical diagnosis where accurate uncertainty quantification is necessary [17].

A well-known alternative approach for Bayesian machine learning are Gaussian processes (GPs). Gaussian processes allow exact inference leveraging linear regres-

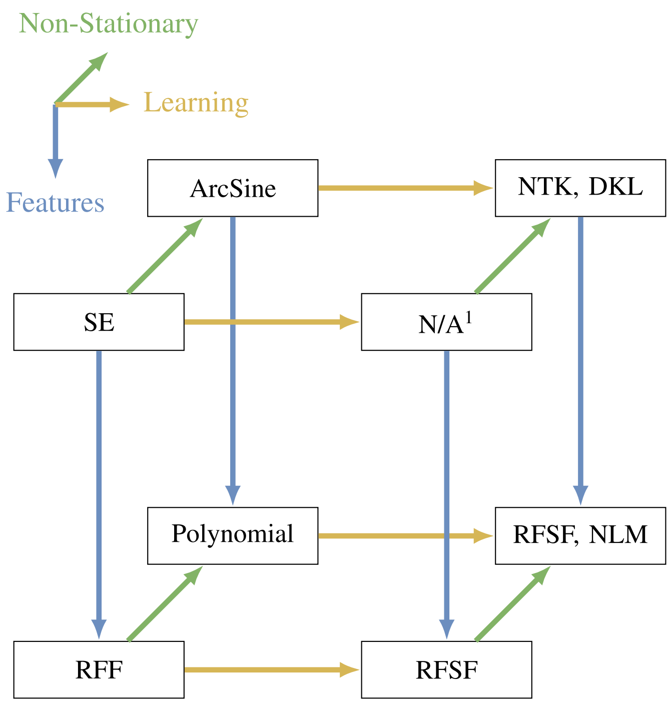


Figure 1: Connections between various (Bayesian) regression models. The three axis distinct them between using kernels vs. features (blue), relying on manually designing vs. learning (yellow), and stationary vs. non-stationary behavior (green). RFSFs (our method) are not clearly stationary or non-stationary and theoretical analysis is up to future research. Abbreviations used in the figure: SE (Squared Exponential), RFF (Random Fourier Features), RFSF (Random Fourier Series Features), NLM (Neural Linear Model), NTK (Neural Tangent Kernel), DKL (Deep Kernel Learning)

[1]There is, to the best of our knowledge, no method for learning kernels white simultaneously guaranteeing stationarity.

sion. Despite the great uncertainty estimation of GPs, their raw prediction power is limited by and highly dependent on the kernel. This dependence is reflected in the high amount of kernels that have been studied throughout the years [18, ch. 2]. Motivated by this dependence, methods for (deep) kernel learning (DKL) have been developed [19]–[21]. However, exact inference with kernels is a tedious task requiring inversion of an $N \times N$-matrix (the Gram matrix) where $N$ is the number of data points. The computational complexity of this is cubic, prohibiting online use of GPs for large data sets [22]. Approaches for reducing the computational complexity have been proposed such as *inducing points* [23] or the Nyström method [24], [25].

Alternatively, one can resort to Bayesian linear regression with features. To mimic GP regression, these features are chosen to approximate a kernel, e.g., the squared exponential (SE) kernel. A well-known choice are random Fourier features (RFFs) that can approximate arbitrary stationary kernels and are often configured for the SE kernel as this is possible in closed form [22]. However, the SE kernel usually produces results that are too smooth [26] and is, by design, not able to capture functions with non-stationary length-scale.

*Contribution:* We propose an extension of RFFs, random Fourier series features (RFSFs), that build a bridge between (a) RFFs, (b) DKL, and (c) BNNs by (a) reducing the computational complexity of GP regression due to working with features, (b) enrich the capacity of GPs by adding more parameters and reducing the need of (manually) designed kernels, and (c) using classical training methods known from NNs for optimizing the hyper-parameters. Figure 1 illustrates these connections on three axis: kernels vs. features, designing vs. learning, and stationary vs. non-stationary length-scales. Despite the lack of a proof and given only empirical evidence, we found that RFSFs can represent various kernels and features, bridging the gap between stationary and non-stationary kernels.

We will now continue by highlighting connections to related work (section II). Subsequently, we will lay out some preliminaries (section III), present the methodology of RFSFs in section IV, and finally empirically evaluate and summarize our findings as well as provide direction for future research (sections V and VI).

## II. Related Work

We closely follow the work of Watson et al. [17] here. We structure related work into the origins of motivation for this work: Bayesian neural networks and Gaussian processes.

*a) Bayesian Neural Networks:* Bayesian neural network have a long history directly of using NNs as statistical models [4] and for regularization [27]. Many approaches are based on Markov-chain Monte Carlo (MCMC) [28]–[30], however, these approaches are computationally expensive and scale poorly with larger models. This motivated the development of variational inference (VI) methods for efficient approximation [27], [31], [32]. Some alternative approximation techniques are, for instance, Laplace approximation [4], [7], [33], ensembles [9], [13], [34], [35], and expectation propagation [6]. Another alternative are Bayesian last layers (BLLs) networks where the last layer-weights are treated in a Bayesian fashion [36]. They have successfully been applied in a variety of tasks [37]–[42].

*b) Gaussian Processes:* It was shown that NNs with infinite width are equivalent to GPs under mild conditions [5]. But also beyond this equivalence a lot of focus is put on the intersection of GPs and BNNs. DKL [19] is concerned with learning closed-form deep kernels using NNs to find more expressive covariance functions. A similar idea are manifold GPs [20] which extract intermediate features from data on which a designed covariance function performs better. Despite the great success of GPs, exact inference is computationally expensive. This motivated the development of approximate GPs in many fashions such as inducing points and direct approximations of the kernel [22], [24].

## III. Preliminaries

### A. Fourier Series

*Fourier series* are a principled way to represent any periodic function (satisfying the Dirichlet conditions [43]) as a linear combination of sine and cosine waves. The resulting approximation $\widehat{f}_K$ of the function $f$ can be represented in three different fashions: in sine-cosine form, amplitude-phase form, and exponential form. In the sine-cosine form, the series is represented by the sum of separate sine and cosine waves with separate amplitudes,

$$\widehat{f}_K(x) = \frac{a_0}{2} + \sum_{k=1}^{K} a_k \cos(\omega k x) + b_k \sin(\omega k x), \quad (1)$$

where the Fourier coefficients $a_k$ and $b_k$ are given by

$$a_k = \frac{1}{\widetilde{T}} \int_{x_0}^{x_0+2\widetilde{T}} f(x) \cos(\omega k x) \, \mathrm{d}x$$

$$b_k = \frac{1}{\widetilde{T}} \int_{x_0}^{x_0+2\widetilde{T}} f(x) \sin(\omega k x) \, \mathrm{d}x \, .$$

with $\omega := \pi/\widetilde{T}$ Here, $\widetilde{T}$ is half the period of $f$ (for instance, $f(x) = \sin(x)$ has the half-period $\widetilde{T} = \pi$). In amplitude-phase form,

$$\widehat{f}_K(x) = \frac{A_0}{2} + \sum_{k=1}^{K} A_k \cos(\omega k x - \varphi_k),$$

the series is represented by individual cosine waves with amplitudes and phases. These can be computed from the Fourier coefficients as follows:

$$A_k = \sqrt{a_k^2 + b_k^2} \qquad \varphi_k = \operatorname{arctan2}(b_k, a_k).$$

For the following, we stick to the former version (sine-cosine) for practical reasons: the amplitude-phase formulation introduces ambiguities as $\varphi \equiv \varphi + 2\pi$. When optimizing numerically, this can cause problems due to either having ambiguous optima or having to include constraints.

### B. *Gaussian Process Regression*

GP regression can be viewed in two fashions: firstly, as an extension of linear regression to an infinite number of features involving computation of the limit $n \to \infty$ where $n$ is the number of features. Secondly, as an infinite Gaussian distribution over a function space where every finite subset of the random variables is jointly Gaussian. However, both definitions (or views) yield the exact same result. For the rest of this paper we will stick to the former definition.

We closely follow Rasmussen and Williams [44] in terms of notation and will give a brief overview over it now. Let $\boldsymbol{x}_*$ and $y_*$ be a test input and target[1], respectively, then the GP regression posterior is a Gaussian distribution with the following mean and variance:

$$\mathbb{E}[y_*] = \boldsymbol{k}_*^\top \mathbf{K}^{-1} \boldsymbol{y} \quad \mathbb{V}[y_*] = k_* - \boldsymbol{k}_*^\top \mathbf{K}^{-1} \boldsymbol{k}_*. \tag{2}$$

Here, $\mathbf{K}$ denotes the Gram matrix on the training inputs, $\boldsymbol{k}_*$ denotes the evaluation of the kernel between the test and train inputs organized into a column vector, and $k_*$ is the kernel evaluation of the test input $\boldsymbol{x}_*$; train targets are organized into $\boldsymbol{y}$.

[1]When predicting, the target is not known and the GP defines a Gaussian distribution over it.

An exemplary kernel is the SE kernel $k_{\mathrm{SE}}(\boldsymbol{x} - \boldsymbol{y}) = \exp\{-\|\boldsymbol{x} - \boldsymbol{y}\|_2^2/(2\ell^2)\}$ with the length-scale[2] $\ell^2$. However, it has been shown that this kernel often produces extremely smooth functions (as it is infinitely differentiable) often unrealistic for real-world data [26]. Despite its drawbacks, the SE kernel remains one of the most widely used kernels due to its simplicity [44, p. 83].

While employing the "kernel trick" allows great flexibility with (potentially) infinite-dimensional feature spaces, it introduces major computational challenges due to the inversion of the Gram matrix in eq. (2). That is, the inversion has time complexity $\mathcal{O}(N^3)$, where $N$ is the number of data points in the training data set. One approach for tackling this problem are random Fourier features.

## IV. Methods

In this section we cover the methodology and core contributions of our work, staring with a discussion of RFFs and introducing the novel RFSFs.

### A. *Random Fourier Features*

By explicitly modeling the features, the inversion of the Gram matrix can be avoided by resorting to computing the distribution over the weights explicitly, i.e., "switching back" to Bayesian linear regression. A well-known approach for this are RFFs [22] which approximate the SE kernel. A single RFF is given by

$$\boldsymbol{z}_{\boldsymbol{\omega}}(\boldsymbol{x}) = \begin{bmatrix} \cos(\langle \boldsymbol{\omega} | \boldsymbol{x} \rangle) \\ \sin(\langle \boldsymbol{\omega} | \boldsymbol{x} \rangle) \end{bmatrix}. \tag{3}$$

This definition is based on the observation that

$$k(\boldsymbol{x} - \boldsymbol{y}) = \mathbb{E}_{\boldsymbol{\omega} \sim p(\cdot)} \big[ \langle \boldsymbol{z}_{\boldsymbol{\omega}}(\boldsymbol{x}) | \boldsymbol{z}_{\boldsymbol{\omega}}(\boldsymbol{y}) \rangle \big] \tag{4}$$

where $k(\boldsymbol{x} - \boldsymbol{y})$ is a stationary kernel and $p(\boldsymbol{\omega})$ is its Fourier transform (which is a proper probability distribution due to Bochner's theorem [26]). For the SE kernel, $p(\boldsymbol{\omega})$ is tractable and equal to a standard normal distribution [44]. As the integral in eq. (4) is intractable, it is usually approximated using Monte-Carlo estimation over $\boldsymbol{\omega}$. Let $\{\boldsymbol{w}_j\}_{j=1}^D$ be the particles sampled from $p(\boldsymbol{\omega})$. The corresponding feature particles $\boldsymbol{z}_{\boldsymbol{\omega}_j}(\cdot)$ are then concatenated, forming the complete feature $\boldsymbol{z}(\cdot)$ which is scaled by $1/\sqrt{D}$ such that the inner product yields the usual Monte Carlo approximation of an expectation.

[2]The length-scale determines the "roughness" of the function samples [44, p. 15].

### B. Random Fourier Series Features

We extend RFFs to random Fourier *series* features (RFSF) by splitting up eq. (3) further into sub-features with separate amplitudes for the sine/cosine component and adding the respective scaling factors for $\boldsymbol{x}$:

$$\tilde{\boldsymbol{z}}_{\boldsymbol{\omega}}^{(k)}(\boldsymbol{x}) = \begin{bmatrix} a_k \cos\bigl(\omega k \left\langle \boldsymbol{\omega} \middle| \boldsymbol{\Lambda}^{-1} \middle| \boldsymbol{x} \right\rangle\bigr) \\ b_k \sin\bigl(\omega k \left\langle \boldsymbol{\omega} \middle| \boldsymbol{\Lambda}^{-1} \middle| \boldsymbol{x} \right\rangle\bigr) \end{bmatrix}. \tag{5}$$

which are then summed over $k = 1, 2, \ldots, K$. The matrix $\boldsymbol{\Lambda}$ represents the length-scales which can be either isotropic for a single length-scale but also different for the input dimensions, allowing automatic relevance determination (ARD). This formulation is inspired by the sine-cosine formulation of a Fourier series (eq. (1)), motivating the name. Note that the half-period $\widetilde{T}$ has the function of a "secondary" length-scale applied equally to all input dimensions.

To find the optimal hyper-parameters $\boldsymbol{a}_{0:M}$, $\boldsymbol{b}_{0:M}$, and $\boldsymbol{\Lambda}$, we use the empirical Bayes approximation [45, p. 165], i.e., maximize the marginal log-likelihood over the training data. Beyond the kernel's parameters, the data is assumed to have Gaussian aleatoric noise with zero mean and variance $\sigma_n^2$ which is an additional hyper-parameter.

As said in the introduction, theoretical analysis is up to future work. We will therefore directly move to empirical evaluation in the next section.

## V. Evaluation

In this section, we will empirically evaluate the proposed RFSFs. We aim to assess the following central hypotheses:

**Hypothesis 1.** *"Do RFSFs outperform RFFs (due to their increased parameter capacity)?"*

Additionally, we try to answer:

**Hypothesis 2.** *How do different initial values of the feature's amplitudes affect the performance?*

In the following, we will first present some implementation details (section V-A) and subsequently present the experiment setup (section V-B) and results (section V-C).

### A. Implementation

We implemented RFSFs using GPyTorch [21] and used Adam [46] for maximizing the marginal log-likelihood. Despite the computational advantage of not having to compute (and, most importantly, invert) the Gram matrix, we implemented RFSFs directly as a GPyTorch kernel. This allows a unified view and evaluation when comparing to well-known kernels like the SE kernel. Also, it allowed to just modifying the existing implementation of RFFs, reducing the risk of mistakes. We published our implementation on GitHub[3].

### B. Experiment Setup

To assess our hypothesis, we evaluate our approach on three classes of data sets: synthetic (table I), UCI [47] (table II), and robotics (real data from a cartpole with the states as inputs and control signal as target). The model quality is measured by the RMSE and log-likelihood on the test data (using the test/train splits provided along [8] for a fair comparison). While the RMSE on the mean measures raw predictive performance and how closely the mean follows the true data, the log-likelihood also assess the uncertainty quantification. For RFSF, we assess three different initializations of the amplitudes:

- *Random:* Sampled from a uniform distribution, i.e., $\boldsymbol{a}_{0:M}, \boldsymbol{b}_{0:M} \sim \mathcal{U}(0, 1)$.
- *ReLU:* Taken from the Fourier series for a periodic ReLU, i.e., $a_m = (\widetilde{T}(-1)^m - \widetilde{T})/(m^2\pi^2)$ and $b_m = -(\widetilde{T}(-1)^m)/(m\pi)$ for $m > 0$ and $a_0 = \widetilde{T}/2$ and $b_0 = 0$ (see appendix A for further details and the derivation).
- *Single Harmonic (SH):* All set to zero except for the 0-th which are set to one, i.e., $a_0 = b_0 = 1$, $\boldsymbol{a}_{1:M} = \boldsymbol{b}_{1:M} = \boldsymbol{0}$.

The idea behind each is as follows. The random initialization is most simplistic and a baseline for the others. Initializing the amplitudes as (periodic) ReLUs sets the connection to kernel learning and BNNs with the ReLU being one of the most common activation functions. Starting with a single harmonic similar to RFFs connects RFSFs closer to the idea behind RFFs and serves as an assessment of our hypothesis that RFSFs should have higher capacity than RFFs (i.e., if the marginal log-likelihood would not get better by touching the other amplitudes, they shall not change).

We compare our method to the SE and RFF kernels. For some data sets (UCI Boston, Concrete, Power, and Yacht), we also compare to popular approaches for BNNs:

- *Gaussian Bayesian Last Layer (GBLL) [44]:* A BLL NN with Gaussian conjugate prior on the weight.
- *Ensemble [9]:* An ensemble of NNs all of which provide mean and variance of the target. Using

[3]https://github.com/fdamken/random-fourier-series-features

| **Name** | **Function** | **Domain** |
|---|---|---|
| Cosine | $\cos(2\pi x)$ | $[-0.5, 0.5]$ |
| Heaviside[†] | $\Theta(x)$ | $[-0.5, 0.5]$ |
| Heavi-Cosine | $\Theta(x)\cos(2\pi x)$ | $[-0.5, 0.5]$ |
| Gap-Cosine | $\cos(2\pi x)$ | $[-0.75, -0.25) \cup (0.5, 0.75]$ |

Table I: Synthetic Data Sets; All data sets are augmented with additive zero-mean Gaussian noise with $\sigma^2 = 10^{-4}$. [†]The heaviside function is defined as $\Theta(x) = \max\{0, \operatorname{sign}(x)\}$

| **Name** | **Short Name** | **Source** [47] |
|---|---|---|
| Boston Housing | Boston | |
| Concrete Compression Strength | Concrete | [48] |
| Combined Cycle Power Plant | Power | [49], [50] |
| Yacht | Yacht | |
| Energy Efficiency | Energy | [51] |
| Kinematics of 8-Link Robot | Kin8nm | |
| Naval Propulsion Plants | Naval | [52] |
| Protein Tertiary Structure | Protein | |
| Wine Quality (Red) | Wine | [53] |

Table II: UCI Data Sets

some neat tricks during training, Ensemble provides predictions along with uncertainty quantification.

- *Maximum A-Posteriori (MAP):* Regularized neural network trained to output mean and covariance of the target.

We include results for this models for a broader comparison, however, we did not implement them ourselves and took the results from [17] (with permission).

### *C. Results*

We first compare visually the quality of RFSFs to the SE kernel (and RFFs) on the synthetic data. As this data is one-dimensional, it is straightforward to visualize. Figure 2 (on page 6) shows the results of the aforementioned kernels on the synthetic data. It can be seen that the RFFs are hardly distinguishable from SE (which makes sense as RFFs can be proven to approximate the SE kernel). Therefore, we only discuss differences between RFSFs and the SE kernel which can be transferred to comparing RFSFs to RFFs. For every data set except the Gap-Cosine, both RFSFs and the SE kernel approximate the true function reasonably, although SE fails to capture discontinuities (Heaviside and Heavi-Cosine) and makes them extremely smooth. However, both kernels exhibit a small length-scale not appropriate for the functions. This is visible in the roughness of the GP samples and is especially prevailing for the SE on the Heaviside and Heavi-Cosine data sets (figs. 2b and 2e). This shows that RFSFs can capture non-stationary trends (cf. fig. 1) opposed to the SE kernel. For the Gap-Cosine data set, the SE kernel (fig. 2h) clearly outperforms RFSFs (fig. 2g) within the gap by actually capturing the overall trend (with the true function lying withing the confidence interval).

Quantitative results for all data sets are summarized in table III. Table IIIa contains results for the synthetic data (and Cartpole) and table IIIb contains UCI-results and, for a broader comparison, some results from [17]. The results for the remaining UCI data sets not covered in table IIIb are given in table IV in appendix appendix C for brevity (the results on the reduced data sets is sufficient for the discussion).

*1) Synthetic Data Sets:* On synthetic data, RFSFs perform roughly equal to the SE kernel and RFFs on the Cosine data set in terms of the log-likelihood and outperform both in terms of the RMSE (again on the Cosine data set). Also, they perform better on the Heaviside data set in terms of the log-likelihood. However, for all other metrics (log-likelihood and RMSE of Heavi-Cosine and Gap-Cosine as well as RMSE on Heaviside), either the SE kernel or RFFs outperform all variants of RFSFs.

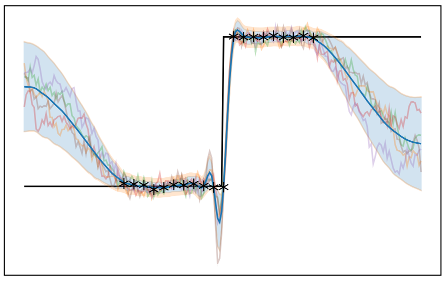
Focusing just on the RFSFs variants, no clear trend can be determined in terms of the log-likelihood. For the RMSE, however, the ReLU initialization is (with the exception of Cosine) outperformed by one of the other variants (Random or SH).

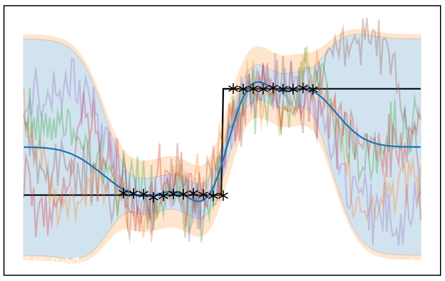
*2) UCI Data Sets:* In terms of the log-likelihood, RFSFs are either outperformed by or equal to the SE kernel and Ensemble. For the remaining data sets in table IV for which [17] do not report results, the results are similar. Within the class of RFSFs variants, the Random initialization is often superior w.r.t. to log-likelihood and RMSE.

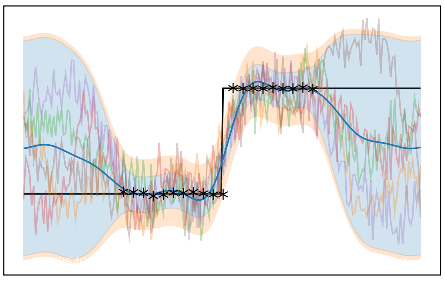
*3) Cartpole Data Set:* On the Cartpole data set, the SE kernel outperforms RFSFs and RFFs in both log-likelihood and RMSE by far. For the different RFSFs variants, no clear distinction (difference of more than the measurement uncertainty) is visible.

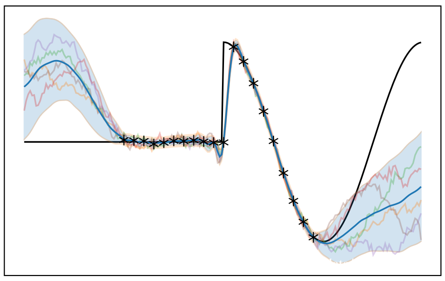
## VI. Conclusion

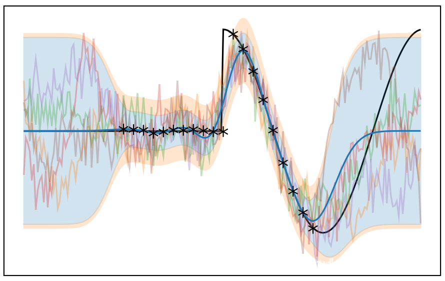
From the evaluation in section V, we conclude that there is no advantage of extending RFFs to RFSFs. In the following subsections we explore our experimental hypothesis in greater detail and lay out how we came to this conclusion. Finally, in section VI-C, we propose some directions for future research and highlight some peculiarities we found.

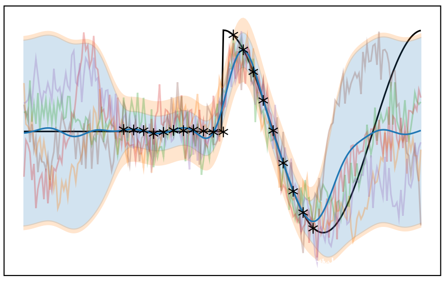

(a) RFSFs on Heaviside

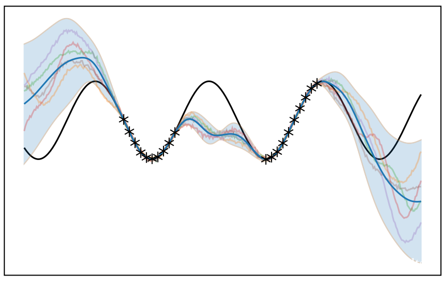

(b) SE on Heaviside

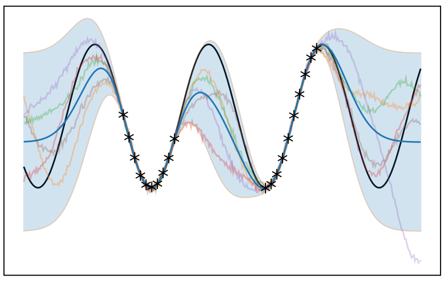

(c) RFFs on Heaviside


(d) RFSFs on Heavi-Cosine


(e) SE on Heavi-Cosine


(f) RFFs on Heavi-Cosine


(g) RFSFs on Gap-Cosine


(h) SE on Gap-Cosine

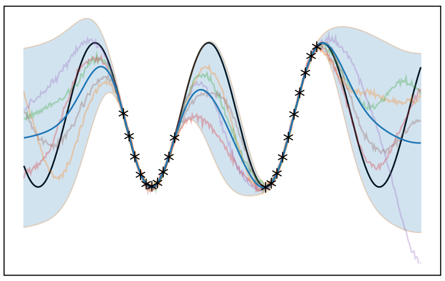

(i) RFFs on Gap-Cosine

Figure 2: Gaussian process regression results using RFSFs, SEs, and RFFs kernels for the synthetic data sets. The shaded areas depict the epistemic/aleatoric (posterior) uncertainty, the blue solid line depicts the (posterior) mean, the black stars are the training samples, and the black solid line depicts the true function. The colorful lines are samples from the GPs.

### A. *Hypothesis 1*

Hypothesis 1 was that "RFSFs outperform RFFs (due to their increased parameter capacity)." We tested this hypothesis by comparing RFSFs to RFFs and a SE kernel on a variety of data sets and train/test splits. As we saw in section V-C, RFSFs are competitive on some data sets, but their advantage is not consistent and they fall short in lots of cases compared to RFFs. Hence, we conclude that even though RFSFs have a greater capacity than RFFs, they hardly gain any performance from the added complexity. We want to highlight that this is also true w.r.t. to the SH initialization, i.e., initializing RFSFs such that they mimic RFFs in the initial phase.

### B. *Hypothesis 2*

In the second hypothesis 2 we asked how "different initial values of the feature's amplitudes affect the performance." We investigated this by testing three different initializations: Random, ReLU, and SH. The results presented in section V-C show that we did not saw a major trend in these initializations.

### C. *Future Work*

We investigated various reasons why RFSFs perform worse than RFFs. For future work and to tackle this question, theoretical analysis of what kernel is approximated by eq. (5) can be insightful. Additionally, a better understanding of the effect of the initial half-period value on the performance is desired (see appendix B).

| | Model | | Data Set | | | | |
|---|---|---|---|---|---|---|---|
| | | | Cosine | Heaviside | Heavi-Cosine | Gap-Cosine | Cartpole |
| **Log-Lik.** | RFSF | Random | **2.43** | 0.11 | −1.66 | 1.27 | −9.88 ± 1.86 |
| | | ReLU | 2.34 | **0.80** | −0.90 | 1.50 | −12.30 ± 2.31 |
| | | SH | 2.37 | **0.21** | −1.23 | 1.52 | −9.73 ± 2.10 |
| | GP | SE | **2.44** | 0.73 | **0.77** | **2.58** | **−3.21 ± 1.64** |
| | | RFF | **2.44** | 0.73 | **0.78** | **2.59** | −7.38 ± 1.94 |
| **RMSE** | RFSF | Random | 0.45 | 0.35 | 0.55 | 0.91 | 0.91 ± 0.08 |
| | | ReLU | **0.39** | 1.06 | 3.55 | 1.62 | 1.01 ± 0.10 |
| | | SH | 0.70 | 0.35 | 0.48 | 0.75 | 0.98 ± 0.11 |
| | GP | SE | 0.45 | **0.30** | **0.29** | **0.40** | **0.77 ± 0.07** |
| | | RFF | 0.45 | **0.31** | **0.30** | 0.42 | 1.26 ± 0.10 |

(a) Results for the synthetic and robotics data sets.

| | Model | | Data Set | | | |
|---|---|---|---|---|---|---|
| | | | Boston | Concrete | Power | Yacht |
| **Log-Lik.** | RFSF | Random | −2.40 ± 0.05 | **−2.94 ± 0.05** | −2.78 ± 0.01 | −0.80 ± 0.02 |
| | | ReLU | −2.39 ± 0.05 | **−2.93 ± 0.04** | −2.80 ± 0.01 | −0.86 ± 0.02 |
| | | SH | −2.44 ± 0.06 | **−2.94 ± 0.05** | −2.78 ± 0.01 | −0.83 ± 0.02 |
| | GP | SE | **−2.38 ± 0.05** | **−2.98 ± 0.06** | −2.82 ± 0.01 | −0.80 ± 0.02 |
| | | RFF | −2.40 ± 0.06 | **−3.01 ± 0.05** | −2.84 ± 0.01 | −0.80 ± 0.02 |
| | GBLL[†] | Leaky ReLU | −2.90 ± 0.05 | −3.09 ± 0.03 | −2.77 ± 0.01 | −1.67 ± 0.11 |
| | | Tanh | −3.06 ± 0.03 | −3.21 ± 0.03 | −2.83 ± 0.01 | −0.70 ± 0.10 |
| | Ensemble[†] | Leaky ReLU | −2.48 ± 0.09 | **−3.04 ± 0.08** | **−2.70 ± 0.01** | −0.35 ± 0.07 |
| | | Tanh | −2.48 ± 0.08 | **−3.03 ± 0.07** | **−2.72 ± 0.01** | **−0.03 ± 0.05** |
| | MAP[†] | Leaky ReLU | −2.60 ± 0.07 | −3.04 ± 0.04 | −2.77 ± 0.01 | −5.14 ± 1.62 |
| | | Tanh | −2.59 ± 0.06 | −3.11 ± 0.04 | −2.76 ± 0.01 | −1.77 ± 0.53 |
| **RMSE** | RFSF | Random | **2.95 ± 0.15** | **4.70 ± 0.14** | 3.90 ± 0.03 | 0.51 ± 0.03 |
| | | ReLU | 3.51 ± 0.44 | 4.77 ± 0.15 | 3.95 ± 0.04 | 0.53 ± 0.03 |
| | | SH | 3.17 ± 0.17 | **4.66 ± 0.15** | 3.88 ± 0.03 | 0.52 ± 0.03 |
| | GP | SE | **2.81 ± 0.12** | 4.98 ± 0.15 | 4.03 ± 0.03 | 0.51 ± 0.04 |
| | | RFF | 2.98 ± 0.13 | 5.08 ± 0.15 | 4.11 ± 0.03 | 0.52 ± 0.04 |
| | GBLL[†] | Leaky ReLU | 4.19 ± 0.17 | 5.01 ± 0.18 | 3.85 ± 0.03 | 1.09 ± 0.09 |
| | | Tanh | 4.61 ± 0.23 | 5.50 ± 0.23 | 4.09 ± 0.04 | 0.43 ± 0.03 |
| | Ensemble[†] | Leaky ReLU | **2.79 ± 0.17** | **4.55 ± 0.12** | **3.59 ± 0.04** | 0.83 ± 0.08 |
| | | Tanh | **2.71 ± 0.13** | **4.51 ± 0.13** | **3.66 ± 0.04** | **0.38 ± 0.03** |
| | MAP[†] | Leaky ReLU | 3.02 ± 0.17 | **4.75 ± 0.12** | 3.81 ± 0.04 | 0.94 ± 0.09 |
| | | Tanh | **3.01 ± 0.17** | 5.15 ± 0.13 | 3.78 ± 0.04 | **0.39 ± 0.04** |

(b) Results for the UCI datasets. [†]Values taken from [17] with permission.

Table III: Evaluation results for the presented data sets. For UCI and Cartpole, the mean along with the measurement uncertainty are computed over the provided train/test splits. The synthetic data sets only have a single train/test split, hence no uncertainty can be provided. Both the log-likelihood and RMSE are reported where the higher and lower values are better, respectively. The best values for a data set are depicted boldface. Values are considered equal if their confidence regions overlap (w.r.t. the best mean value). The results for the remaining data sets are given in table IV in appendix C.

## References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012.

[2] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-Based Model Predictive Control: Toward Safe Learning in Control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.

[3] E. Bradford, L. Imsland, D. Zhang, and E. A. del Rio Chanona, "Stochastic data-driven model predictive control using gaussian processes," *Computers & Chemical Engineering*, vol. 139, p. 106844, Aug. 2020.

[4] D. J. C. MacKay, "A Practical Bayesian Framework for Backpropagation Networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, May 1992.

[5] R. M. Neal, *Bayesian Learning for Neural Networks*. Springer Science & Business Media, Dec. 2012.

[6] J. M. Hernandez-Lobato and R. Adams, "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks," in *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Jun. 2015, pp. 1861–1869.

[7] J. Denker and Y. LeCun, "Transforming Neural-Net Output Levels to Probability Distributions," in *Advances in Neural*

*Information Processing Systems*, vol. 3. Morgan-Kaufmann, 1990.

[8] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, Jun. 2016, pp. 1050–1059.

[9] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[10] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Network," in *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Jun. 2015, pp. 1613–1622.

[11] A. Foong, D. Burt, Y. Li, and R. Turner, "On the Expressiveness of Approximate Inference in Bayesian Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 15 897–15 908.

[12] A. Y. K. Foong, Y. Li, J. M. Hernández-Lobato, and R. E. Turner, "'In-Between' Uncertainty in Bayesian Neural Networks," *arXiv:1906.11537 [cs, stat]*, Jun. 2019.

[13] I. Osband, J. Aslanides, and A. Cassirer, "Randomized Prior Functions for Deep Reinforcement Learning," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[14] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model' s uncertainty? Evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[15] F. Wenzel, K. Roth, B. S. Veeling, J. Świątkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin, "How Good is the Bayes Posterior in Deep Neural Networks Really?" *arXiv:2002.02405 [cs, stat]*, Jul. 2020.

[16] J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez, "Quality of Uncertainty Quantification for Bayesian Neural Network Inference," *arXiv:1906.09686 [cs, stat]*, Jun. 2019.

[17] J. Watson, J. A. Lin, P. Klink, J. Pajarinen, and J. Peters, "Latent Derivative Bayesian Last Layer Networks," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. PMLR, Mar. 2021, pp. 1198–1206.

[18] D. Duvenaud, "Automatic model construction with Gaussian processes," Thesis, University of Cambridge, Nov. 2014.

[19] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep Kernel Learning," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. PMLR, May 2016, pp. 370–378.

[20] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth, "Manifold Gaussian Processes for regression," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 3338–3345.

[21] A. Jacot, F. Gabriel, and C. Hongler, "Neural Tangent Kernel: Convergence and Generalization in Neural Networks," *arXiv:1806.07572 [cs, math, stat]*, Feb. 2020.

[22] A. Rahimi and B. Recht, "Random Features for Large-Scale Kernel Machines," in *Advances in Neural Information Processing Systems*, vol. 20. Curran Associates, Inc., 2007.

[23] E. Snelson and Z. Ghahramani, "Sparse Gaussian Processes using Pseudo-inputs," in *Advances in Neural Information Processing Systems*, vol. 18. MIT Press, 2005.

[24] E. J. Nyström, "über Die Praktische Auflösung von Integralgleichungen mit Anwendungen auf Randwertaufgaben," *Acta Mathematica*, vol. 54, no. none, pp. 185–204, Jan. 1930.

[25] S. Sun, J. Zhao, and J. Zhu, "A review of Nyström methods for large-scale machine learning," *Information Fusion*, vol. 26, pp. 36–48, Nov. 2015.

[26] M. L. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media, Jun. 1999.

[27] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, 1993, pp. 5–13.

[28] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An Introduction to MCMC for Machine Learning," *Machine Learning*, vol. 50, no. 1, pp. 5–43, Jan. 2003.

[29] M. D. Hoffman and A. Gelman, "The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[30] T. Chen, E. Fox, and C. Guestrin, "Stochastic Gradient Hamiltonian Monte Carlo," in *Proceedings of the 31st International Conference on Machine Learning*. PMLR, Jun. 2014, pp. 1683–1691.

[31] C. Peterson and E. Hartman, "Explorations of the mean field theory learning algorithm," *Neural Networks*, vol. 2, no. 6, pp. 475–494, Jan. 1989.

[32] A. Graves, "Practical Variational Inference for Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 24. Curran Associates, Inc., 2011.

[33] H. Ritter, A. Botev, and D. Barber, "A Scalable Laplace Approximation for Neural Networks," https://iclr.cc/Conferences/2018/Schedule?showEvent=224, Vancouver, Canada, Jan. 2018.

[34] D. Barber and C. M. Bishop, "Ensemble learning in Bayesian neural networks," *Nato ASI Series F Computer and Systems Sciences*, vol. 168, pp. 215–238, 1998.

[35] T. Pearce, F. Leibfried, and A. Brintrup, "Uncertainty in Neural Networks: Approximately Bayesian Ensembling," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. PMLR, Jun. 2020, pp. 234–244.

[36] M. Lazaro-Gredilla and A. R. Figueiras-Vidal, "Marginalized Neural Network Mixtures for Large-Scale Regression," *IEEE Transactions on Neural Networks*, vol. 21, no. 8, pp. 1345–1351, Aug. 2010.

[37] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams, "Scalable Bayesian Optimization Using Deep Neural Networks," in *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Jun. 2015, pp. 2171–2180.

[38] N. Weber, J. Starc, A. Mittal, R. Blanco, and L. Màrquez, "Optimizing over a bayesian last layer," in *NeurIPS Workshop on Bayesian Deep Learning*, 2018.

[39] C. Riquelme, G. Tucker, and J. Snoek, "Deep bayesian bandits showdown," in *International Conference on Learning Representations*, 2018.

[40] R. Pinsler, J. Gordon, E. Nalisnick, and J. M. Hernández-Lobato, "Bayesian Batch Active Learning as Sparse Subset Approximation," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[41] B. O'Donoghue, I. Osband, R. Munos, and V. Mnih, "The uncertainty bellman equation and exploration," in *International Conference on Machine Learning*, 2018, pp. 3836–3845.

[42] S. W. Ober and C. E. Rasmussen, "Benchmarking the Neural Linear Model for Regression," *arXiv:1912.08416 [cs, stat]*, Dec. 2019.

[43] A. V. Oppenheim, A. S. Willsky, S. H. Nawab, w. Hamid, and

G. M. Hernández, *Signals & Systems*. Pearson Educación, 1997.

[44] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, 2nd ed. MIT Press, 2006.

[45] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, Jan. 2006.

[46] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017.

[47] D. Dua and C. Graff, "UCI Machine Learning Repository," http://archive.ics.uci.edu/ml, 2017.

[48] I. C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete Research*, vol. 28, no. 12, pp. 1797–1808, Dec. 1998.

[49] H. Kaya, P. Tüfekci, and F. S. Gürgen, "Local and global learning methods for predicting power of a combined gas & steam turbine," in *Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE*, 2012, pp. 13–18.

[50] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," *International Journal of Electrical Power & Energy Systems*, vol. 60, pp. 126–140, Sep. 2014.

[51] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy and Buildings*, vol. 49, pp. 560–567, Jun. 2012.

[52] A. Coraddu, L. Oneto, A. Ghio, S. Savio, D. Anguita, and M. Figari, "Machine learning approaches for improving condition-based maintenance of naval propulsion plants," *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, vol. 230, no. 1, pp. 136–153, 2016.

[53] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, Nov. 2009.

[54] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 2623–2631.

## Appendix

### A. Fourier Series of the Periodic ReLU

Using the ReLU, we define the *Periodic* ReLU (PeReLU) $\mathrm{ReLU}_{\circlearrowleft}(x) := \mathrm{ReLU}\big([x+\widetilde{T}]_{2\widetilde{T}} - \widetilde{T}\big)$, where $[\cdot]_U$ is the modulus operator w.r.t. $U$ defined as $[\cdot]_U : \mathbb{R} \to \mathbb{R}^+ : u \mapsto \min\{u + zU \,:\, u + zU \geq 0,\, z \in \mathbb{Z}\}$. The $\pm\widetilde{T}$ in the PeReLU definition ensures that the PeReLU still has the usual ReLU properties in a region around zero, i.e., that $\mathrm{ReLU}_{\circlearrowleft}(x) = 0$ for $x < 0$ and $\mathrm{ReLU}_{\circlearrowleft}(x) = x$ for $x > 0$. The coefficients of the Fourier series $a_m$ and $b_m$ for $m > 0$ are then given

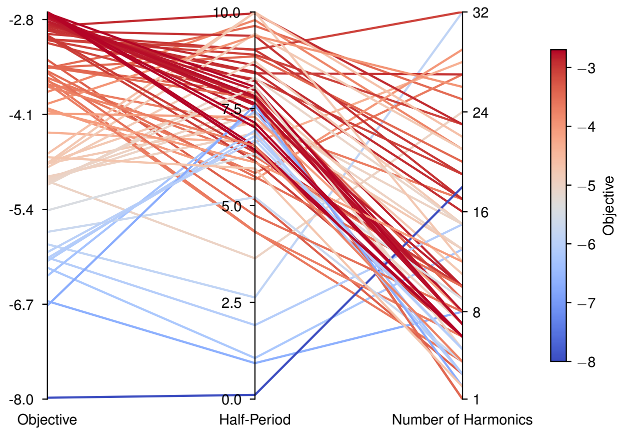


Figure 3: Parallel coordinates plot of the hyper-parameter's values connecting specific Optuna [54] trials. It can be seen that the half-period value has a major influence on the object (leftmost axis). Trial was run on the UCI [47] Boston Housing data set with random RFSF initialization. Objective is the negative log-likelihood on the test data.

by solving the following integrals:

$$\begin{aligned} a_m &= \frac{1}{\widetilde{T}} \int_{-\widetilde{T}}^{\widetilde{T}} \mathrm{ReLU}_{\circlearrowleft}(x) \cos\Big(m\pi x/\widetilde{T}\Big)\,\mathrm{d}x \\ &= \frac{\widetilde{T}(-1)^m - \widetilde{T}}{m^2\pi^2} \\ b_m &= \frac{1}{\widetilde{T}} \int_{-\widetilde{T}}^{\widetilde{T}} \mathrm{ReLU}_{\circlearrowleft}(x) \sin\Big(m\pi x/\widetilde{T}\Big)\,\mathrm{d}x \\ &= -\frac{\widetilde{T}(-1)^m}{m\pi} \end{aligned}$$

For $m = 0$, the coefficients are simply $a_0 = \widetilde{T}/2$ and $b_0 = 0$.

### B. Influence of the Half-Period

We noticed a major influence of the half-period during hyper-parameter optimization using Optuna [54]. That is, the higher the (initial) value of $\widetilde{T}$, the higher the objective (see fig. 3). As a consequence we included the half-period into the set of (hyper-) parameter that are optimized using empirical Bayes, however, this phenomenon is still in place. This is also reflected in Optuna's importance rating of $0.86$.

### C. Result for the Remaining UCI Data Sets

The results for the remaining UCI data sets not covered in table III are given in table IV.

| | **Model** | | **Data Set** Energy | Kin8nm | Naval | Protein | Wine |
|---|---|---|---|---|---|---|---|
| **Log-Lik.** | RFSF | Random | **−0.70 ± 0.02** | 0.68 ± 0.05 | −78.19 ± 69.72 | −2.94 ± 0.03 | −0.11 ± 0.07 |
| | | ReLU | −0.74 ± 0.02 | **0.97 ± 0.03** | −172.57 ± 104.83 | −629.05 ± 384.60 | −0.11 ± 0.06 |
| | | SH | −0.74 ± 0.02 | 0.52 ± 0.07 | −62.69 ± 55.40 | −2.96 ± 0.03 | **0.01 ± 0.06** |
| | GP | SE | **−0.68 ± 0.02** | −0.22 ± 0.24 | **6.91 ± 0.15** | **−2.89 ± 0.00** | −0.84 ± 0.05 |
| | | RFF | **−0.69 ± 0.02** | 0.75 ± 0.04 | −1941.56 ± 248.64 | −2.90 ± 0.00 | −0.89 ± 0.04 |
| **RMSE** | RFSF | Random | **0.48 ± 0.02** | **0.07 ± 0.00** | 0.01 ± 0.00 | **3.97 ± 0.02** | **0.64 ± 0.01** |
| | | ReLU | **0.49 ± 0.02** | **0.07 ± 0.00** | 0.01 ± 0.00 | **4.00 ± 0.03** | 0.66 ± 0.01 |
| | | SH | **0.49 ± 0.02** | 0.08 ± 0.00 | 0.01 ± 0.00 | **3.97 ± 0.02** | 0.65 ± 0.01 |
| | GP | SE | **0.48 ± 0.02** | 0.08 ± 0.00 | **0.00 ± 0.00** | 4.34 ± 0.01 | **0.63 ± 0.01** |
| | | RFF | **0.48 ± 0.01** | **0.07 ± 0.00** | 0.02 ± 0.00 | 4.42 ± 0.01 | **0.63 ± 0.01** |

Table IV: Refer to table III for a description of the provided values.