Kernel Combinations for Sparse Gaussian Processes in Correlated Web Traffic Forecasting

Arnav Chahal[†]
Department of Computer Science
Vanderbilt University
Nashville, TN 37235
arnav.chahal@vanderbilt.edu

Aditya Shrey[†]
Department of Computer Science
Vanderbilt University
Nashville, TN 37235
aditya.shrey@vanderbilt.edu

1 Introduction

This project aims to explore the impact of different kernels by combining multiple kernels within a Sparse Gaussian Process (GP) framework for correlated time series forecasting. In particular, we use a subset of the Wikipedia Traffic Data Exploration dataset. We create a linear combination of unique kernels and determine which kernels have a larger effect. By maximizing the evidence lower bound (ELBO) in the context of Titsias's Sparse GP framework [9], we efficiently approximate the log marginal likelihood while addressing the scalability challenges of GPs for large datasets. The ELBO optimization relies on inducing points, which summarize the data to reduce computational demands. We tune the weights of each kernel as hyperparameters to assess the individual contributions of each kernel type. This approach provides insights into kernel combinations in real-world forecasting contexts and highlights the intricacies of Sparse GP modeling.

We utilize a combination of the Squared Exponential Kernel, Spectral Mixture Kernel, Matérn Kernel, Linear Kernel, and Sinusoidal Kernel, each with a weighted hyperparameter to quantify its impact. This allows us to identify the most effective kernels for web traffic forecasting and explore whether specific kernel combinations enhance model accuracy. We hypothesize that the Spectral Mixture and Matérn Kernels will play a more significant role in accurately forecasting web traffic time series data. Additionally, we investigate how the number of inducing points in the Sparse GP framework and the step size influence kernel importance. We expect the kernel weights to remain relatively consistent across different numbers of inducing points. For step size, we anticipate that an optimal range exists—neither too small nor too large—that results in better performance.

2 Related Work

2.1 Kernels

Beyond more traditional kernels, such as the Linear Kernel (Eq. 1), Sinusoidal Kernel (Eq. 2), and Squared Exponential Kernel, more advanced kernels have been developed to address specific challenges in time series forecasting and other machine learning tasks. Notably, the Spectral Mixture Kernel and the Matérn Kernel were selected for their potential relevance to our study.

$$k_{Linear}(x, x\prime) = \sigma_b^2 + \sigma_v^2(x - c)(x\prime - c) \tag{1}$$

$$k_{Sinusoidal}(x, x\prime) = \sigma_f^2 \cos\left(\frac{2\pi|x - x\prime|}{p}\right) \tag{2}$$

[†]Both authors contributed equally to this work.

Squared Exponential Kernel As a basis, we define the Squared Exponential (SE) Kernel, also known as the Radial Basis Function (RBF) Kernel, which is expressed as

$$k_{SE}(x, x\prime) = \sigma_f^2 \exp\left(-\frac{(x - x\prime)^2}{2\ell^2}\right). \tag{3}$$

It takes in two data points and includes a hyperparameter ℓ, which corresponds to the length scale, determining how sensitive the model is to differences between input features when measuring similarity between points. The length scale governs the rate at which similarity decreases as the distance between data points increases. However, the SE Kernel assumes that the underlying function is infinitely differentiable, meaning that the modeled function is infinitely smooth, which can sometimes limit its reliability in cases where the data exhibits non-smooth or abrupt changes [7].

Spectral Mixture Kernel Before introducing the Spectral Mixture Kernel, it is important to establish some foundational concepts. A *spectral density* is the Fourier transform of a kernel, describing how the energy of a signal is distributed across different frequencies. This property provides insight into the periodic components and underlying patterns in the data. A *Gaussian Mixture* refers to a weighted combination of Gaussian distributions, where each component is characterized by its own mean, variance, and weight. Gaussian mixtures are often used to model complex data distributions by capturing diverse patterns within the data.

The Spectral Mixture Kernel is particularly useful for time series. The Kernel is derived by modeling a spectral density with a Gaussian Mixture [10]. Essentially this kernel enables us to learn our stationary kernel by learning our spectral density. This is because we say that any stationary kernel is the Fourier transformation of our Spectral density and vice versa. This will allow us to know the stationary kernel if we know the spectral density. To model the spectral density, we use Gaussian mixtures, which help capture complex frequency patterns. The idea is that the data has multiple dominant frequencies, each represented by a Gaussian component. By fitting a Gaussian mixture to the spectral density, the Spectral Mixture Kernel can flexibly handle signals with a variety of behaviors: smooth trends, period trends, or a mix of both. The kernel is given by

$$k_{SM}(x, x\prime) = \sum_{q=1}^{Q} w_q \exp\left(-2\pi^2 \|x - x\prime\|^2 v_q\right) \cos\left(2\pi \|x - x\prime\| \mu_q\right). \tag{4}$$

Matérn Kernel The Matérn Kernel is a class of kernels that generalizes the SE kernel. Like the SE kernel, it has a hyperparameter corresponding to the length scale, ℓ [7]. However, it introduces an additional parameter ℓ that controls the smoothness of the function being modeled. The Matérn kernel provides flexibility by allowing for different levels of smoothness depending on the value of v. This flexibility makes the Matérn kernel particularly useful in situations where the smoothness of the underlying function is uncertain, offering more control over the prior assumptions regarding the modeled data—ideal for time-series data where the smoothness may vary between samples [2]. The kernel with $\nu = 3/2$ is expressed as

$$k_{Matern}(x, x\prime) = \sigma_f^2 \left(1 + \frac{\sqrt{3}|x - x\prime|}{\ell}\right) \exp\left(-\frac{\sqrt{3}|x - x\prime|}{\ell}\right). \tag{5}$$

Multiple Kernel Learning In addition to implementing the kernels above, a combination of multiple kernels is possible, creating a new single kernel [7]. Multiple kernel learning (MKL) has been proven as an effective way to improve model flexibility by tuning each kernel's weight as a hyper parameter. Gönen and Alpaydın discuss the benefits of MKL, emphasizing its flexibility. This adaptability is especially beneficial when data exhibit a range of behaviors: such as varying smoothness or periodicity—since combining kernels provides a richer, adaptive representation for complex datasets. However, there are drawbacks to this approach. While the paper primarily discusses this learning approach for Support Vector Machines (SVMs) and certain forms of regression, it can also be applied to Sparse GPs, where some of the same drawbacks still pose a problem.

The first drawback is computational complexity, as it requires tuning hyperparameters for the weighting of the kernels, especially if kernels are dynamically combined for specific sections of data.

This was a primary concern in the paper, and while we use Sparse GPs to mitigate computational burdens, these methods do not eliminate the complexity problem entirely.

The second drawback involves the choice between one-step and two-step training. In one-step training, the model attempts to optimize both the kernel weights and the base learner parameters simultaneously, which can be computationally intensive and may lead to convergence issues. In contrast, two-step training iteratively optimizes the kernel weights and the base learner parameters in alternating steps, which can stabilize the optimization process but might be slower to converge overall, especially in large datasets [3].

2.2 Sparse Gaussian Processes

GPs have poor scalability with the training of data taking $\mathcal{O}(N^3)$, where N is the number of data. In our context, since we are dealing with a somewhat large dataset, we will need to overcome this challenge. Various work has been done to make GPs tractable in the context of large datasets. Methods, such as Sparse Pseudo-input Gaussian processes (SPGPs), use inducing input points, which act as a compressed representation of the dataset [8]. In considering M inducing points, we obtain a training cost of $\mathcal{O}(M^2N)$, where $M \ll N$. This reduction is achieved because the inducing points summarize the essential structure of the data, allowing the GP to focus its computation on a smaller subset of points, making inference and learning more efficient. One example of this approach is seen in the work by Groot et al. [4], where pseudo-inputs are employed to summarize the data in multi-step time series forecasting, significantly reducing computational complexity while maintaining predictive accuracy in control applications.

However, a significant issue with SPGPs is that the selection and placement of these pseudo-inputs can interfere with the optimization of kernel hyperparameters during training. In particular, the pseudo-inputs can cause non-smooth variations in the marginal likelihood, which complicates hyperparameter learning and can lead to suboptimal results [9].

To address this, we adopt the approach of variational learning of inducing variables as proposed by Titsias [9]. This method jointly optimizes the inducing variables and kernel hyperparameters within a variational framework, introducing a new ELBO on the log marginal likelihood. The ELBO is maximized as part of the Empirical Bayes step and is structured to achieve computational efficiency equivalent to SPGPS, with a cost of $\mathcal{O}(M^2N)$. This approach provides improved consistency and accuracy in hyperparameter optimization by eliminating the non-smooth behavior associated with pseudo-input placement.

The ELBO in Titsias's variational framework is expressed as:

$$ELBO = \mathbb{E}_{q(f|\mathbf{u})}[\log p(\mathbf{y}|f)] - D_{KL}(q(\mathbf{u})\|p(\mathbf{u})) \tag{6}$$

The objective function consists of two components. The data fit term, $\mathbb{E}_{q(f|\mathbf{u})}[\log p(\mathbf{y}|f)]$, measures how well the model explains the observed data by evaluating the likelihood of the data under the model. It encourages the variational distribution over the function values f to match the true distribution, given the inducing points $\mathbf{u}$. By focusing on this data likelihood, we optimize the model's predictive accuracy for the training data. The KL divergence term, $D_{KL}(q(\mathbf{u})\|p(\mathbf{u}))$, serves as a regularizer, encouraging the variational distribution $q(\mathbf{u})$ to stay close to the prior distribution $p(\mathbf{u})$. This regularization helps prevent over-fitting by controlling the complexity of the posterior distribution, ensuring the model generalizes well beyond the training data.

This approach addresses the shortcomings of SPGPs by jointly optimizing the inducing input variables and kernel hyperparameters within a variational framework. The ELBO is maximized during the Emperical Bayes step. This method maintains the same computational efficiency as SPGPs, with a training cost of $\mathcal{O}(M^2N)$, but with improved consistency and accuracy in hyperparameter learning. The variational approach has been successfully applied in time series settings, where it allows efficient and scalable modeling of temporal dependencies [5].

2.3 Time Series Data Synchronization

In time series forecasting, data synchronization is crucial when patterns across different time series respond to shared or interdependent phenomena, as observed in various fields from geosciences to neuroscience. For example, time series synchronization in landslide displacement forecasting has shown that incorporating temporal dependencies can enhance predictive performance and stability in dynamic, interconnected systems [12]. Similarly, the framework for phase synchronization in neuroscience demonstrates how aligned temporal patterns can reveal underlying dynamics in complex systems, such as brain wave synchronization in neurological studies [1, 6]. These approaches illustrate how synchronized patterns across time series can reveal influential shared events or triggers, making synchronization a valuable tool for understanding interactions in time-dependent data.

In our context, web traffic page popularity data is influenced by web search trends, often driven by collective responses to external events, such as news stories, cultural events, or seasonal interests. By leveraging a synchronized GP model, we can capture these temporal dependencies in Wikipedia traffic data more effectively, as synchronized responses to shared events can enhance the model's predictive power. Incorporating synchronization can also help model the cascading effect of a popular event that spreads across related topics, as seen in the study of large-scale networked data [11]. Such synchronized modeling approaches can significantly enhance our understanding of web traffic dynamics by capturing the interdependent and often simultaneous patterns that drive user interest.

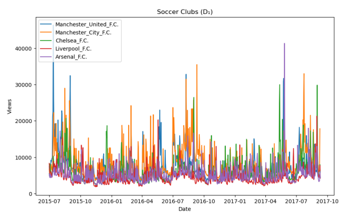
3 Methods

3.1 Data

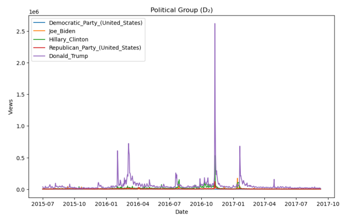
We use the Wikipedia Traffic Data Exploration dataset, originally from a 2017 Kaggle competition hosted by Google. The competition included several smaller datasets alongside a larger test dataset containing the most comprehensive data. We chose to work with this larger test dataset as it aligns best with our objectives.

The dataset tracks daily search volumes for most, if not all, Wikipedia pages from 07/01/2015 to 09/10/2017, providing over 883 columns per page. Due to the large amount of Wikipedia pages (rows), we selected a subset of five specific categories for our analysis. These categories include English Premier League soccer clubs (e.g., Manchester United, Chelsea F.C.), the Democratic and Republican Parties as well as key figures from each, and major technology companies such as Apple and Facebook (now Meta).

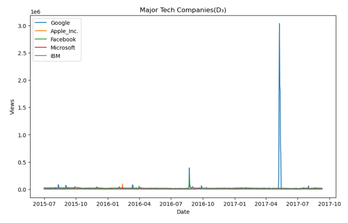
For the three groups Soccer (D_1), Political (D_2), and Technology (D_3), we plotted their respective time series plots. Figures 1a, 1b, and 1c display these plots, showing the daily search volumes over the given period. Each group D_i consists of five representative time series denoted as D_i^j , for $j = \{1, 2, 3, 4, 5\}$. Each time series in D_i represents daily search volumes for a specific entity.



(a) Search volume for Soccer group (D_1)



(b) Search volume for Political group (D_2)

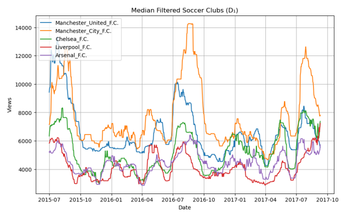


(c) Search volume for Technology group (D_3)

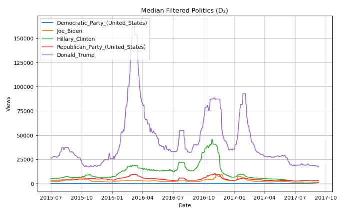
Figure 1: Daily search volumes for each group over the given period.

Both D_2 and D_3 have exhibited extreme behavior that could potentially lead to issues in our predictions. To address this, we utilized two separate approaches. The first approach involved applying a median filtering algorithm. This method uses a 30-day sliding window around each day (15 days before and 15 days after) and replaces the current day's value with the median of that window. At the start of the dataset, the window included the next 30 days, or for the first index, the previous point and the next 29 days. Similarly, at the end of the dataset, the window included the previous 30 days, with adjustments for fewer available points. The graphs illustrating this approach are provided below.

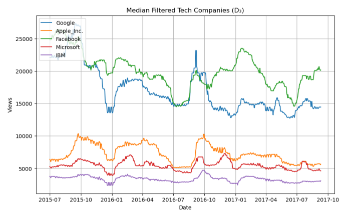
The second approach combined the aforementioned median filtering algorithm with a min-max normalization between 0 and 1. After applying the median filtering, normalization was applied



(a) Median Filtered Search volume for Soccer group (D_1)



(b) Median Filtered Search volume for Political group (D_2)



(c) Median Filtered Search volume for Technology group (D_3)

Figure 2: Median Filtered Daily search volumes for each group over the given period.

independently to each time series in the dataset. However, this approach did not perform well and failed to significantly improve the results. As such, we opted to continue using the first method, applying the median filtering window alone.

The input vector for time t is structured as $X_t = [t, D_i^1(t), D_i^2(t), \dots, D_i^4(t)]$, where $D_i^j(t)$, where $D_i^j(t)$ represents the value of the j-th time series at time t. During training, we use the first 800 entries of each series to learn the patterns, while the remaining data points are reserved for testing.

3.2 Model Formulation

In our approach, we combine a set of kernels $\mathcal{K} = \{k_{sinusoidal}, k_{SE}, k_{SM}, k_{Matern}, k_{linear}\}$ to capture complex temporal patterns present in the Wikipedia traffic data. The combined kernel function, $k_{combined}$, is expressed as a weighted sum of the individual kernels in $\mathcal{K}$, where each kernel $k \in \mathcal{K}$ contributes according to its optimized weight α_k. Formally, this can be written as

$$k_{combined}(x, x\prime) = \sum_{k \in \mathcal{K}} \alpha_k \cdot k(x, x\prime). \tag{7}$$

To optimize this model, we maximize the ELBO (Eq. 6). We efficiently tune both the kernel weights α_k and other kernel hyperparameters. To reduce computational cost, we use a subset of the training data known as inducing points $\mathbf{Z} = \{z_i\}$, which effectively represent the data. The inducing points are selected as a small representative sample of the input data, allowing the Gaussian Process to scale efficiently. These inducing points form the basis of our variational approximation, ensuring the model remains tractable on a large dataset

For prediction, we compute the posterior predictive distribution using the optimized parameters. Given a test input $\mathbf{x}$, the predictive mean $\mu_{pred}(\mathbf{x}^*)$ and predictive variance $\sigma^2_{pred}(\mathbf{x}^*)$ are given by:

$$\mu_{pred}(\mathbf{x}^*) = \mathbf{K}_{*Z}^\top \mathbf{K}_{ZZ}^{-1} \mathbf{m}, \tag{8}$$

$$\sigma^2_{pred}(\mathbf{x}^*) = k_{combined}(\mathbf{x}, \mathbf{x}^*) - \mathbf{K}_{*Z}^\top \mathbf{K}_{ZZ}^{-1} \mathbf{K}_{*Z}, \tag{9}$$

where $\mathbf{K}_{*Z}$ is the covariance vector between the test point and the inducing points, $\mathbf{K}_{ZZ}$ is the covariance matrix of the inducing points, and $\mathbf{m}$represents the mean of the latent function.

For evaluating model performance, we use two metrics. First, we calculate the Mean Squared Error (MSE) between the observed data points $\mathbf{y}$ and the predictive mean μ_{pred}. We evaluate the model's uncertainty handling by computing the Negative Log Predictive Density (NLPD), which quantifies the fit of the model's predictive distribution to the observed data. For a set of test observations $\mathbf{y}_{test}$. This metric captures both the accuracy of the predictive mean and the reliability of the predictive variance, providing a comprehensive assessment of the model's performance. Together, these evaluation metrics allow us to determine the efficacy of our kernel combination approach and validate our hypotheses about kernel selection in forecasting tasks.

4 Experiments

4.1 Impact of Step Size

In this experiment, we investigate the effects of using different step sizes during ELBO maximization in our Sparse GP model. The goal is to evaluate how the step size impacts model convergence, kernel weights, predictive accuracy, and computational stability. The experiment uses the Soccer dataset (D_1) with 100 inducing points selected from the training data. The model incorporates a combination of kernels, including the SE Kernel, Linear Kernel, Matérn Kernel, Sinusoidal Kernel, and Spectral Mixture Kernel with 10 mixture components. The model was trained on the first 602 days of the dataset and tested on predictions for the last 200 days.

The inducing points were initialized as a subset of the training data. A fixed hyperparameter initialization was set for all experiments. The spectral mixture kernel was the only kernel where each of the mixture parameters were chosen from a Gaussian distribution. These parameters were concatenated into a single vector to represent the initial hyperparameters in the unconstrained space, enabling efficient optimization.

ELBO maximization was performed with step sizes ranging from 10^{-7} to 10^{-10}. The results, summarized in Table 1, show the number of hyperparameter updates observed for each step size. The kernel weight distributions, shown in Table 2, indicate how different step sizes affect the contribution of each kernel. Smaller step sizes were associated with lower total noise variance and improved predictive accuracy, as measured by MSE and NLPD. However, smaller step sizes also resulted in fewer hyperparameter updates, reflecting a trade-off between computational stability and optimization flexibility.

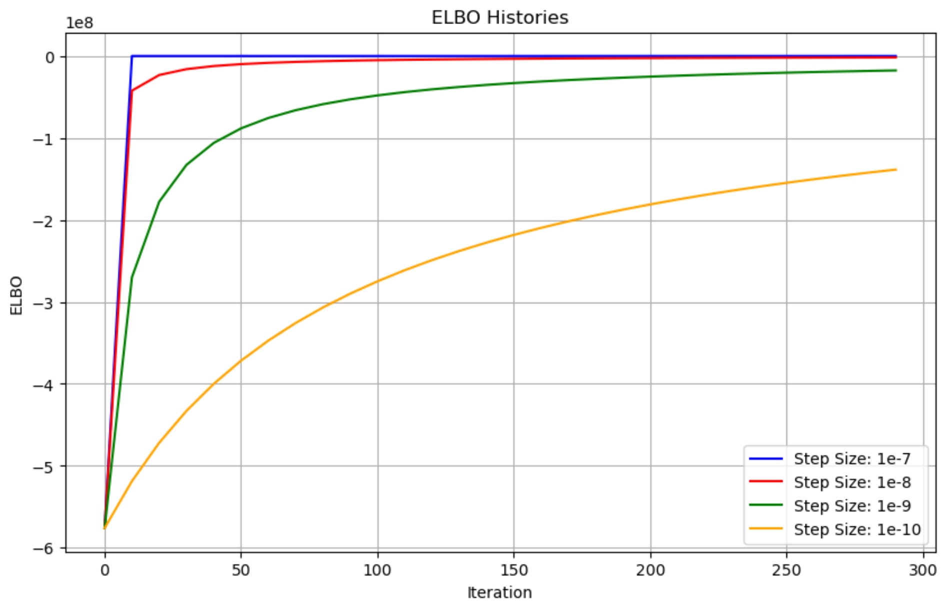


Figure 3: Impact of step size on ELBO maximization during Sparse GP optimization. The figure illustrates the convergence behavior of the ELBO for various step sizes ranging from 10^{-7} to 10^{-10}.

Step Size	**Hyperparameter Changes**
1×10^{-7}	15
1×10^{-8}	31
1×10^{-9}	**32**
1×10^{-10}	31

Table 1: Effect of Step Size on Hyperparameter Changes. A total of 128 hyperparameters are tunable.

Step Size	SE (%)	Linear (%)	Matérn (%)	Sinusoidal (%)	Spectral Mixture (%)
10^{-7}	17.70	14.45	19.76	15.29	32.80
10^{-8}	17.56	16.68	25.47	19.54	20.76
10^{-9}	18.76	17.78	21.83	19.19	22.44
10^{-10}	19.07	18.23	20.69	19.50	22.51

Table 2: Kernel Weight Percentages Across Different Step Sizes.

Step Size	MSE	Train NLPD	Test NLPD
1×10^{-7}	**17,400,221.13**	**14.42**	**17.61**
1×10^{-8}	16,296,801.60	1,314.80	1,888.86
1×10^{-9}	14,980,071.83	12,333.57	17,239.30
1×10^{-10}	13,539,993.98	135,399.94	179,399.94

Table 3: Comparison of MSE and NLPD values for different step sizes.

The analysis of step size tuning (Table 3) reveals that smaller step sizes result in lower MSE values, indicating more accurate predictions. For example, at a step size of 1×10^{-10}, the MSE is significantly reduced compared to larger step sizes. Similarly, NLPD values suggest that smaller step sizes improve model calibration and predictive performance, particularly in the training phase. However, only 15 hyperparameters were tuned with the larger step size of 1×10^{-7} compared to 31 and 32 with all other step sizes, indicating the model is not tuning as granularly as possible.

4.2 Impact of Inducing Points

In a similar setup when we studied the impact of step size, we investigated the effects of using a different number of inducing points on our Sparse GP model. We held a fixed step size of 1×10^{-8}. The performance metrics for different configurations of inducing points (25, 50, 100, and 200) are summarized in Tables 7 and 4. As shown, varying the number of inducing points did not significantly impact the model's predictive performance. While small differences were observed in MSE and NLPD, these variations are not substantial enough to indicate a strong dependence on the choice of inducing points in this experiment. The kernel weights and noise parameters also showed minor variations.

Inducing Points	SE (%)	Linear (%)	Matérn (%)	Sinusoidal (%)	Spectral Mixture (%)
25	16.17	16.94	31.42	21.32	14.16
50	16.96	17.12	28.15	21.27	16.49
100	17.21	16.37	26.59	19.55	20.28
200	16.24	13.83	22.39	16.08	31.45

Table 4: Kernel Weights and Noise for Different Numbers of Inducing Points.

The results indicate that increasing the number of inducing points provides minimal improvements in both training and testing metrics. This suggests that the Sparse GP model's performance is relatively robust to the number of inducing points, at least within the range tested. Therefore, computational efficiency may be prioritized without significant loss of accuracy.

Inducing Points	MSE	Train NLPD	Test NLPD
25	**16,892,358.74**	**399.92**	580.54
50	16,961,921.35	400.04	**582.22**
100	16,940,356.90	401.00	583.66
200	16,915,976.89	403.28	587.39

Table 5: MSE and NLPD for Different Numbers of Inducing Points.

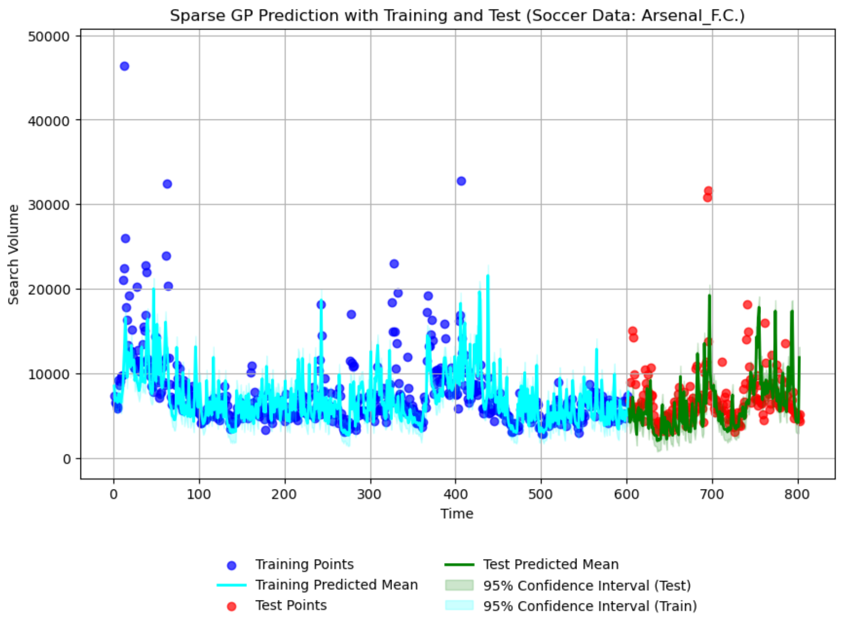


Figure 4: Unfiltered Soccer Data Forecast.

4.3 Combined Kernel Weight

As we use all three of our data groups for the final experiment, we tested this experiment using the median-filtered dataset to ensure cleaner and more reliable input. By optimizing the weights of each kernel in the combination, we aim to identify which kernels contribute most meaningfully to the prediction task for each of the three groups— Soccer (D_1), Political (D_2), and Tech (D_3).

As done in prior experiments, the model is trained on the first 602 days of data and then used to predict the last 200 days. Our input will be a specific day and the corresponding search volume for all group members excluding our output group (e.g., if Donald Trump is our output, then for our input will be $[t, D_i^1(t), D_i^2(t), D_i^3(t), D_i^4(t)]$ where D_i^5 is Donald Trump). By synchronizing time data across group members, we allow the model to pick up on correlated patterns and spikes in searches (e.g., increased interest in soccer clubs at the start of the Premier League season in August), which may enhance predictive accuracy. We train the model by maximizing the ELBO for 500 steps in this experiment. Table 6 displays the breakdowns of the Kernel Weights for each of the datasets.

Datasets (filtered)	**SE (%)**	**Linear (%)**	**Matérn (%)**	**Sinusoidal (%)**	**Spectral Mixture (%)**
Soccer	25.30	16.45	17.29	16.88	24.08
Politics	20.33	19.56	20.33	20.33	19.46
Technology	22.17	16.79	20.56	17.69	22.80

Table 6: Kernel Weights for the different filtered datasets.

The dataset we found the most success on was the Soccer dataset. For our kernel initialization values the hyperparameter initializations we found the most success with were a step size of 1×10^{-7}, and a noise of 5 and when applicable length scales of 10 for each of the kernels. One important thing to note about these parameter initializations is that this problem is a non-convex optimization problem and as such different parameters will lead to vastly different results.

Additionally, using the unfiltered soccer dataset from Figure 1a, we were also able to obtain good forecasting results despite the data being in its most complex, raw form. This forecast (Figure 4) was arguably better than our median-filtered dataset which we think might be because the data did not have major outliers our model was able to better find patterns compared to the median-filtered dataset which might have lost some information.

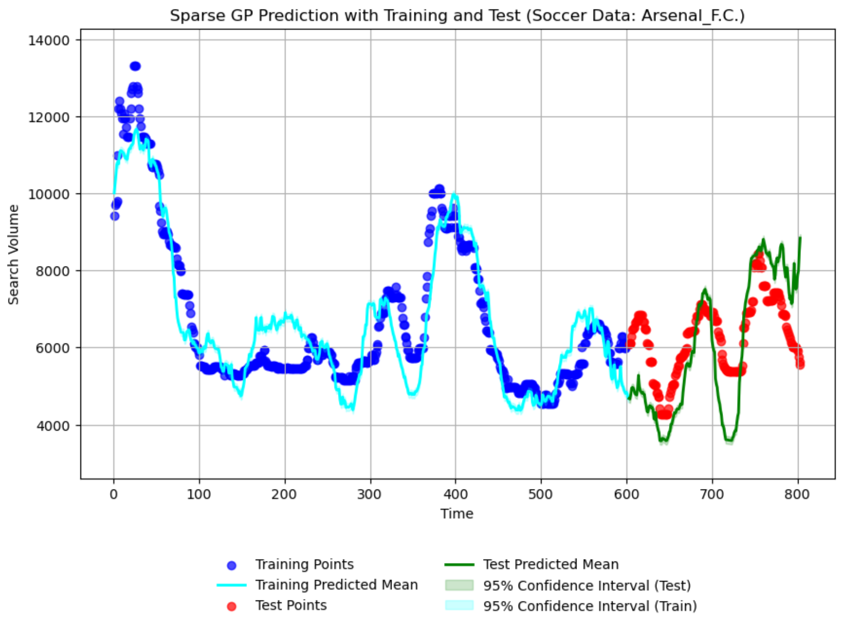


Figure 5: Soccer Forecast with Median Filter.

However from Figure 5 we can see that we forecasted generally well in the near term with some slight deviations toward the end of our dataset. However, it is interesting to see the difference between the filtered vs non-filtered forecasts. This trend did not continue in the other data sets where the median-filtered forecasts vastly outperformed the original datasets and as such the original dataset forecasts are omitted.

For the Political Dataset, we did not find as much success despite heavily varying parameters and initializations, as it seemed our model was unable to best identify different trends in the data. It is important to note that the time series we left out was the Trump time series, which varied the most compared to other time series in the dataset. For this experiment, our step size was 1×10^{-6}, with a noise for each kernel between 2 and 3, and length scales that varied from 1 to 30. From the graph 6, we can see the lack of discernible trends and the model's difficulty in capturing the variability.

Finally, on the Technology sector dataset, the model's performance was satisfactory,though there is still room for improvement. Our model seemed to be able to identify trends in the data, but it still did not perform too impressively. For this model we used a step size of 1×10^{-7} and maintained a noise of 0.1 and length scales of 1 to 10. However, given that the search volume has a downward trend we believe our model still performed well on the data.

From the Table 7 we can see some interesting metrics. The models we thought did the best had the highest NLPD and MSE. The forecast for the Politics despite visually being the worst was actually the one with the lowest MSE and NLPD. By producing a forecast that hews closely to a steady, baseline level—essentially a flat line—any large deviations from the true underlying values are minimized. As a result, the MSE remains low, since squared errors do not become excessively large. This creates the illusion of a decent fit when assessed purely by MSE. Although this approach does not capture any nuanced structure, it avoids being severely penalized by large deviations. Consequently, the NLPD remains concentrated around values that, while not reflective of the data's true complexity, do not stray far enough to incur heavy log-loss penalties.

However, when considering the soccer and technology datasets, the models that attempted to more actively follow the true data patterns incurred higher MSE. When these models guessed incorrectly, their errors were larger, increasing the overall MSE. This might give the appearance of a worse fit from the MSE standpoint, but it also indicates that the model was at least striving to learn the underlying complexities. The higher NLPD in these cases may also reflect this earnest but imperfect attempt: by incorporating more variability and uncertainty in their predictions, these models acknowledge their

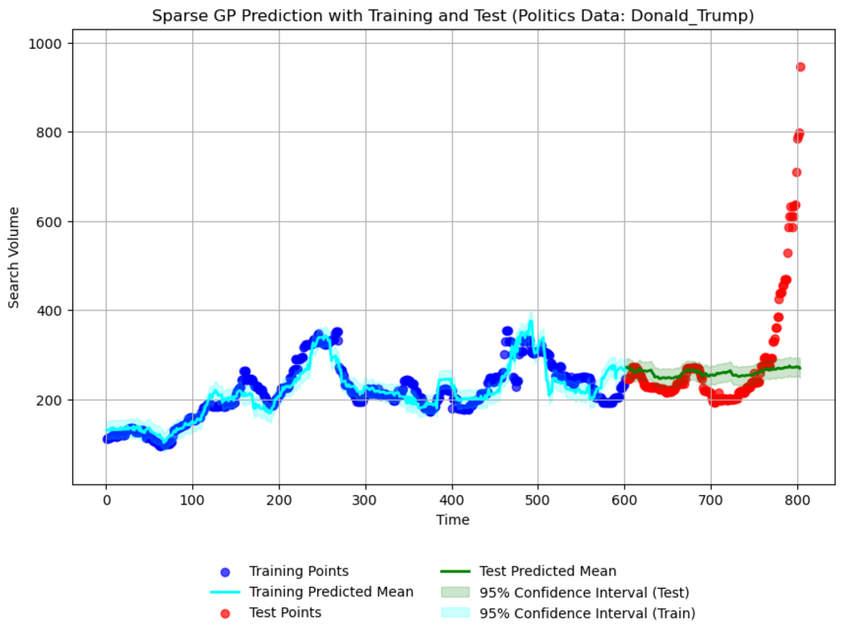


Figure 6: Politics Forecast with Median Filter. Results of the Political Dataset experiment. Despite varying parameters and initializations, the model struggled to identify trends in the data.

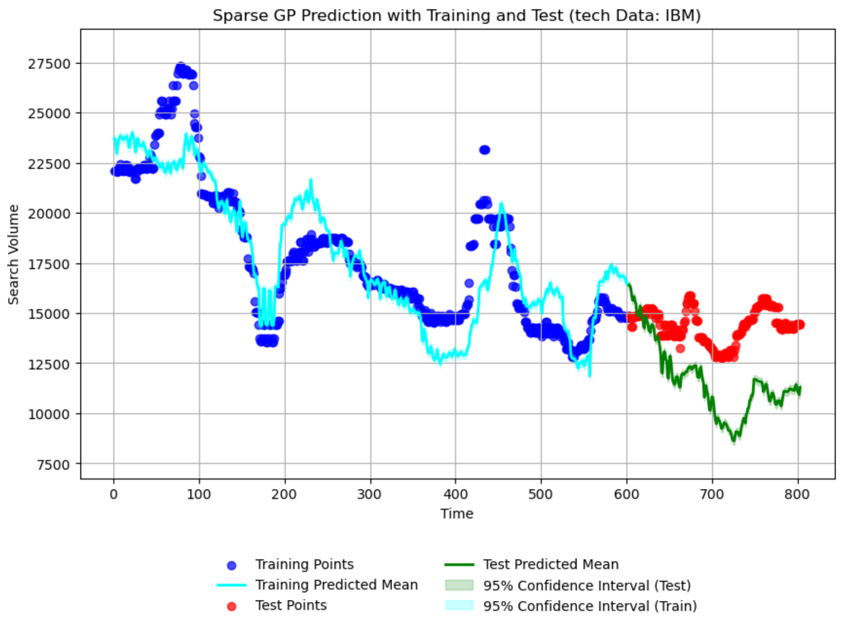


Figure 7: Technology Forecast with Median Filter. We were able to identify some trends in the data but the model tended to not perform as well as we would have liked.

limitations, resulting in less deceptively "good" metrics and a more realistic representation of their predictive confidence.

Datasets (Filtered)	**MSE**	**Train NLPD**	**Test NLPD**
Soccer	1,464,893.16	87.37	212.18
Politics	15,733.65	5.7	42.51
Technology	9,914,455.18	88.61	252.11

Table 7: MSE and NLPD for each model trained on different Datasets.

5 Discussion

In this work, we conducted three key experiments to evaluate the performance of kernel combinations in Sparse GPs for correlated time series forecasting: (1) the impact of step size during ELBO maximization, (2) the effect of the number of inducing points on model performance, and (3) the role of individual kernel contributions across different datasets. Below, we discuss the outcomes of these experiments, their alignment with our hypotheses, and the broader implications of our findings.

Our investigation into the effect of step size revealed a clear trade-off between optimization granularity and computational stability. Smaller step sizes resulted in lower MSE and NLPD values, indicating improved predictive performance and better calibration. However, smaller step sizes also led to fewer hyperparameter updates, suggesting that the optimization process converged less dynamically. While this aligns with our expectation that an optimal step size exists, our results indicate that extremely small step sizes may overfit to local features of the data, potentially neglecting broader trends. This highlights the importance of balancing computational efficiency with accuracy when selecting step sizes for Sparse GPs.

In the second experiment, we examined the impact of varying the number of inducing points. The results showed minimal differences in MSE and NLPD across the configurations tested, suggesting that the model is robust to the number of inducing points within the range tested (25 to 200). This supports our hypothesis that Sparse GPs can effectively leverage inducing points to capture the essential structure of the data without requiring large computational resources. However, we did observe slight variations in kernel weight distributions, indicating that inducing points may influence how individual kernels contribute to the overall model.

The third experiment focused on optimizing kernel weights for three datasets (Soccer, Politics, and Technology). Our findings revealed that kernel contributions varied across datasets, reflecting the diversity of patterns in the data. For instance, the Spectral Mixture Kernel contributed significantly to the Soccer and Technology datasets, likely capturing periodic trends in web traffic. In contrast, the Political dataset exhibited less discernible trends, resulting in more evenly distributed kernel weights. This aligns with our hypothesis that kernel effectiveness is data-dependent, with the Spectral Mixture and Matérn Kernels playing prominent roles in datasets with clear temporal patterns. Interestingly, none of the kernels were completely eliminated during optimization, as all received non-zero weights. This suggests that even kernels with lower contributions provide complementary information that enhances the overall model performance. However, the uniform distribution of weights in some cases raises questions about whether certain kernels are redundant in specific contexts.

Our initial hypothesis posited that the Spectral Mixture and Matérn Kernels would dominate the model for datasets with clear periodic or structured patterns. While this held true for the Soccer and Technology datasets, the Political dataset did not show a strong preference for these kernels, likely due to its higher variability and lack of consistent temporal structure. Additionally, we hypothesized that kernel weights would remain relatively stable across varying numbers of inducing points. The results largely supported this, with minor variations in weights observed. Finally, we expected step size to significantly influence kernel importance and overall model performance. While this was confirmed, the overconfidence of the models at smaller step sizes suggests the need for further exploration into noise variance and uncertainty calibration.

Our experiments highlight the potential of Sparse GPs for modeling correlated time series data, particularly in scenarios where computational efficiency is a priority. However, the overconfidence of our models, as evidenced by narrow confidence intervals even in regions of poor predictions, underscores a critical limitation. This may stem from insufficient noise optimization or the inherent assumptions of the GP framework.

Additionally, while Sparse GPs provide a scalable solution for large datasets, their reliance on inducing points and kernel hyperparameter tuning may make them less suitable for real-time applications. The trade-offs between flexibility, interpretability, and computational cost suggest that Sparse GPs may be more effective for offline analysis and forecasting rather than real-time decision-making.

Overall, our study demonstrates the effectiveness of Sparse GPs with combined kernels for time series forecasting, particularly when data exhibit structured temporal patterns. While challenges remain, such as model overconfidence and limitations in real-time applicability, our findings highlight the importance of kernel selection, step size tuning, and inducing point configuration in achieving optimal performance. Future work could explore alternative kernel combinations, dynamic inducing point selection, and improved uncertainty calibration to further enhance the utility of Sparse GPs in real-world applications.

References

[1] M. Chavez, M. Besserve, C. Adam, and J. Martinerie. Towards a proper estimation of phase synchronization from time series. In *Journal of Neuroscience Methods*, 2006.

[2] Marc G Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

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

[4] Perry Groot, Peter J. Lucas, and Paul Van den Bosch. Multiple-step time series forecasting with sparse gaussian processes. In *Proceedings of the 23rd Benelux Conference on Artificial Intelligence (BNAIC)*. Benelux Association for Artificial Intelligence (BAAI), 2011.

[5] Cesar Lincol Cavalcante Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme Barreto, and Lawrence Neil. Recurrent gaussian processes. In *Inernational Conference on Learning Reprsentations (ICLR)*, 2016.

[6] M. Palus and D. Hoyer. Detecting nonlinearity and phase synchronization with surrogate data. In *IEEE Engineering in Medicine and Biology*, 2002.

[7] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[8] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264. MIT Press, 2005.

[9] Michalis K. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5, pages 567–574. JMLR W&CP, 2009.

[10] Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1067–1075, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[11] Yue Xu, Feng Yin, Wenjun Xu, Jiaru Lin, and Shuguang Cui. Wireless traffic prediction with scalable gaussian process: Framework, algorithms, and verification. In *IEEE Journal on Selected Areas in Communications*, 2019.

[12] Weiqu Yang, Yuran Feng, Jian Wan, and Lingling Wang. Sparse gaussian process regression for landslide displacement time-series forecastin. In *Advanced Application of Deep Learning, Statistical Modelling, and Numerical Simulation on Geo-Environmental Hazards*, 2022.