CS772 Project Report

Enugula Vishnucharan
Department of Computer Science
Indian Institute of Technology, Kanpur
enugulav23@iitk.ac.in

Maurya Jadav
Department of Computer Science
Indian Institute of Technology, Kanpur
mauryaj20@iitk.ac.in

Pradeep Chahal
Department of Computer Science
Indian Institute of Technology, Kanpur
pradeepc21@iitk.ac.in

Abstract

This work is done in extention to the work Accurate and scalable Gaussian Processes for Fine-Grained Air Quality Inference[1]. This seed paper used Gaussian Processes for the air quality inference. We tried to improve the same model by carefully analyzing the kernel functions and enhancing their selection and usage process by using deep kernel learning, neural kernel network techniques and find feature specific kernels.

1 Problem Description

Given S Air Quality monitors, T Timestamps and features (latitude, longitude, temperature, humidity, weather, wind speed and wind direction) and corresponding PM2.5 observations. Aim is to find PM2.5 observation for a new set of locations S^*.

We wanted to improve the existing state-of-the-art inference model based on Gaussian Processes by looking into the gaussian process setting or to come up with a whole new model though relating to some kind of probabilistic setting as per the course context.

A few future improvements were mentioned in the paper, we wanted to understand them properly and implement and append them to the model. These included GP scalability, other non-stationary methods and missing data imputation. We wanted to look at and research methods for these and implement them into the model.

2 Literature Review

Estimating air quality (AQ) is paramount for safeguarding public health and managing environmental resources. Traditionally, interpolation methods such as Kriging, Spline, and Trend, alongside data-driven techniques like deep learning, have been employed to generate AQ maps (Cheng et al., 2018; Xu & Zhu, 2016; Zheng, Liu, and Hsieh, 2013). However, these methods need help in accurately capturing the spatio-temporal dynamics of AQ.

Recent advancements have introduced Gaussian processes (GPs) as a promising tool for AQ estimation (Sampson & Guttorp, 1992; Krige, 1951). GPs, known as Kriging in geostatistics, offer a non-parametric Bayesian approach to modeling spatial and temporal processes. Their significant advantage lies in providing probabilistic predictions and quantifying uncertainty. While GPs have demonstrated effectiveness in various environmental modeling tasks (Plagemann, Kersting, and Burgard, 2008; Paciorek & Schervish, 2003), their application to AQ estimation requires further exploration. Recent studies have focused on leveraging GPs to capture the intricate spatio-temporal patterns of AQ, especially in urban areas (Cheng et al., 2018; Luo et al., 2019).

However, despite the promising results of existing GP-based approaches, ample room still exits for improvement. Understanding the strengths and limitations of GPs in the context of AQ estimation is crucial for advancing research in this field. In addressing specific requirements pertinent to our problem, the utilization of kernel functions within GPs emerged as a superior approach. For instance, AQ data often exhibits variable covariance for locations equidistant apart due to geographical heterogeneity and complex chemical reactions. A Hamming distance-based kernel was employed for categorical features. Furthermore, given the periodic nature of most environmental phenomena in the temporal dimension, such as diurnal traffic patterns and daily and seasonal meteorological variations, a sinusoidal Periodic kernel was utilized to model periodicity within the dataset. To enhance kernel selection and usage, we delved into research exploring techniques such as deep learning kernels, neural kernel networks, and sparse GPs. Literature on these methodologies provided insights into their applicability to our problem, paving the way for further advancements in AQ estimation.

By integrating insights from these studies and techniques, we aim to enhance the effectiveness and accuracy of GP-based approaches for AQ estimation, contributing to advancing state-of-the-art research in this critical field.

2.1 Prior Works on the Problem

The seed paper by Cheng et al. (2018) significantly advances AQ estimation using GPs. By leveraging hourly AQ and meteorological data from Beijing and London, the authors demonstrate the superiority of their GP-based approach over traditional interpolation and data-driven methods. Their model, ADAIN, incorporates time-invariant and time-series data using linear and recurrent neural network layers. The attention mechanism dynamically computes the importance of train stations in predicting AQ at test locations, resulting in accurate and interpretable predictions.

Before the seed paper, existing methods relied on interpolation techniques or data-driven approaches, which often needed help to capture the complex spatio-temporal dynamics of AQ. The work by Xu and Zhu (2016) and Zheng, Liu, and Hsieh (2013) utilized deep learning techniques to generate AQ maps but faced challenges in accurately modeling temporal dependencies and uncertainty.

In contrast, the seed paper introduces GPs as a powerful tool for AQ estimation, leveraging their ability to model complex spatial and temporal processes. The authors demonstrate superior performance in predicting AQ at unmonitored locations by incorporating domain knowledge through carefully designed kernels. The probabilistic nature of GPs enables them to provide point estimates and uncertainty estimates, which are essential for informed decision-making.

3 Novelty of Our Work as Compared to Prior Work

The important elements which makes Gaussian process efficient in estimating is the kernel selection. So we focused more on finding the best kernel compositions for the data available.

3.1 Deep learning kernel

The deep learning kernel concept stands out as a significant departure from conventional approaches to kernel learning in machine learning models. Unlike traditional kernels, often pre-defined mathematical functions, the deep learning kernel offers a dynamic and data-driven approach to kernel representation. At its core, the deep learning kernel is a neural network module intricately woven into the fabric of PyTorch model. Its architecture is a testament to the power of deep learning, featuring a series of interconnected components meticulously designed to extract and learn complex representations from raw data.

At the heart of the deep learning kernel lies a stack of fully connected layers, each equipped with parameters to adapt the input data. These transformations are not limited to simple linear operations but are imbued with non-linearities introduced by activation functions like Rectified Linear Units (ReLU) and Leaky ReLU. Through these non-linear activations, the deep learning kernel gains the capability to capture intricate patterns and dependencies present in the data, paving the way for more nuanced and expressive representations.

The deep learning kernel is fortified with dropout layers and batch normalization to enhance its learning capacity and robustness further. Dropout layers act as a form of regularization by randomly deactivating a fraction of the neurons during training, thereby preventing overfitting and promoting generalization. Meanwhile, batch normalization ensures that the activations of each layer are standardized, leading to more stable and accelerated training dynamics. This combination of regularization techniques safeguards against the pitfalls of overfitting and fosters smoother convergence and improved model performance.

The output layer of the deep learning kernel serves as the culmination of its learning journey, synthesizing the acquired representations into a cohesive kernel output. This output, in turn, becomes an integral part of the broader kernel composition process within the neural kernel network. The deep learning kernel collaborates with other kernel representations through weighted compositions and iterative evaluations to deliver optimal performance in downstream tasks such as regression or classification.

Training the deep learning kernel is a collaborative endeavor orchestrated alongside the rest of the model's components. Leveraging backpropagation and optimization algorithms like the Adam optimizer, the parameters of the deep learning kernel are fine-tuned iteratively to minimize a chosen loss function. This training process not only hones the deep learning kernel's ability to extract meaningful representations from the data but also contributes to the overall efficacy of the model.

3.2 Neural kernel network

Gaussian Process (GP) regression is a powerful tool in various machine learning tasks, particularly in scenarios where uncertainty estimation and flexibility in modeling complex data relationships are crucial. Central to the efficacy of GP regression is the choice of kernel function, which determines the assumptions about the underlying data structure. However, manually selecting an appropriate kernel composition often proves challenging due to the data's vast parameter space and intricate relationships. An automatic search methodology for identifying the optimal kernel composition has been devised to alleviate this burden and enhance the efficiency of model development.

The proposed methodology integrates a diverse set of base kernel functions, each capturing different aspects of data relationships. These base kernels encompass a spectrum of functions, including but not limited to Radial Basis Function (RBF), Matérn, Cosine, and Rational Quadratic kernels.

The search procedure systematically explores various compositions of these base kernels by employing operations such as addition and multiplication. This exploration results in an expansive space of potential kernel compositions, reflecting different combinations of underlying data structures and relationships. For instance, combinations such as (kernel1 + kernel2) * kernel3, (kernel1 * kernel2) + kernel3, etc, are evaluated to unveil the diverse range of interactions between kernels.

A vital component of the automatic search methodology is the Neural Kernel Network (NKN), which facilitates the exploration and evaluation of kernel compositions. The NKN comprises a neural network architecture that dynamically combines and evaluates kernel compositions based on learned weights. It operates by ingesting input data and iteratively adjusting the weights associated with each base kernel, thereby modulating their contribution to the overall composition.

The NKN's architecture includes multiple layers that are responsible for processing input data and computing the weights associated with different kernel compositions. The final layer of the NKN outputs a set of weights corresponding to the relative importance of each base kernel in the composition. These weights are then used to construct the final kernel composition, which is evaluated using a validation dataset.

Upon determining the optimal kernel composition using the NKN, a GP regression model is instantiated using this composition as the kernel function. This automated approach reduces the manual effort required for kernel selection, enhances scalability, and enables exploring a wide range of kernel compositions, ultimately improving model performance and interpretability.

3.3 Finding feature specific kernel

Our methodology integrates advanced neural network architecture with diverse kernel functions to construct a robust framework for Gaussian process regression. At the core of our approach lies the

Kernel Composition Neural Network, designed to select and weigh kernels based on the input data dynamically.

In the Kernel Composition Neural Network, we employ a Kernel Selector module alongside a collection of kernel functions. The Kernel Selector utilizes a fully connected layer to predict the weights associated with each kernel. These weights are learned during the training process, allowing the model to adaptively assign importance to different kernels based on the characteristics of the input data.

To provide a rich representation of the data, we incorporate a variety of kernel functions into our framework. These include the Radial Basis Function (RBF), Matérn, Periodic, and Rational Quadratic (RQ) kernels. Each kernel captures distinct patterns and structures in the data, offering flexibility in modeling diverse datasets.

During the training phase, our model minimizes the mean squared error (MSE) loss function, optimizing the prediction accuracy of the Gaussian process regression. By iteratively adjusting the model parameters, such as kernel weights, the model learns to accurately capture the underlying relationships between input features and target outputs.

To evaluate the performance of our model, we monitor its training and testing performance using the root mean square error (RMSE) metric. This allows us to assess the model's ability to generalize to unseen data and track its learning progress over time.

Upon completion of training, we analyze the learned kernel weights to gain insights into the relevance of different kernels for modeling specific features. Higher weights assigned to certain kernels indicate their effectiveness in capturing the characteristics of corresponding features, enabling us to perform feature-specific kernel analysis.

4 Tools/Software Used

4.1 IDE

1. Visual Studio Code

4.2 Framework

1. PyTorch: PyTorch is a deep learning framework that provides Tensors and dynamic neural networks in Python with strong GPU acceleration. PyTorch is used extensively to define and train neural network models, including modules such as nn.Module, nn.Linear, etc.

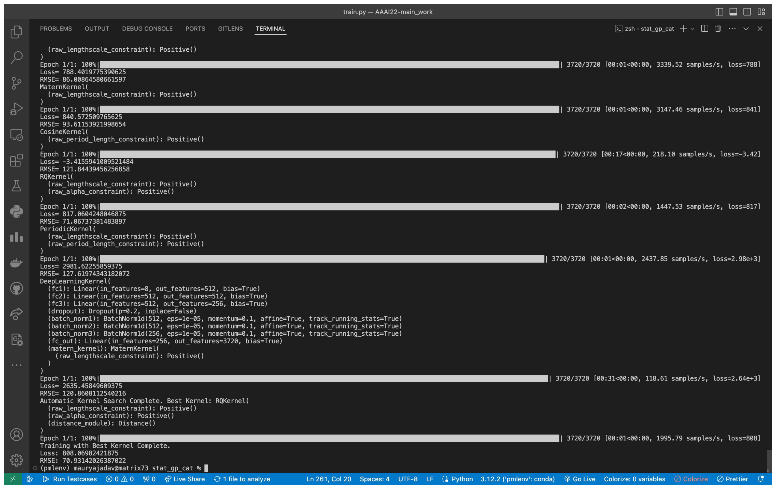
4.3 Libraries

1. Numpy -Used for manipulating multidimentional arrays and matrices.
2. Pandas -Used data structures such as DataFrames to handle data .
3. Torch - Used for machine learning algorithms and optimization routines.
4. GPyTorch: GPyTorch is a Gaussian processes library implemented using PyTorch. It provides scalable, efficient implementations of Gaussian processes and is tightly integrated with PyTorch. GPyTorch is used for Gaussian process regression, including defining kernels (gpytorch.kernels) such as RBFKernel, MaternKernel, CosineKernel, etc., likelihoods (gpytorch.likelihoods) such as GaussianLikelihood, and Gaussian process models.
5. BoTorch: BoTorch is a library for Bayesian optimization built on PyTorch. It provides scalable Bayesian optimization and is designed to work seamlessly with GPyTorch. Although not explicitly used in the provided code, BoTorch could be used for Bayesian optimization tasks in other project parts.
6. scikit-learn (sklearn): scikit-learn is a machine learning library in Python that provides simple and efficient tools for data mining and analysis. scikit-learn is used for evaluation metrics such as mean squared error (mean_squared_error).
7. tqdm: tqdm is a Python library that adds progress bars to code. It is used to create progress bars during model training loops (train_model function) to track the progress of epochs.

8. EarlyStopping: EarlyStopping is a technique used to prevent overfitting in machine learning models by monitoring a metric (e.g., validation loss) and stopping training when the metric stops improving. It is implemented as a custom class (EarlyStopping) and used during model training to stop training early if there is no improvement in validation loss.

5 Experimental Results

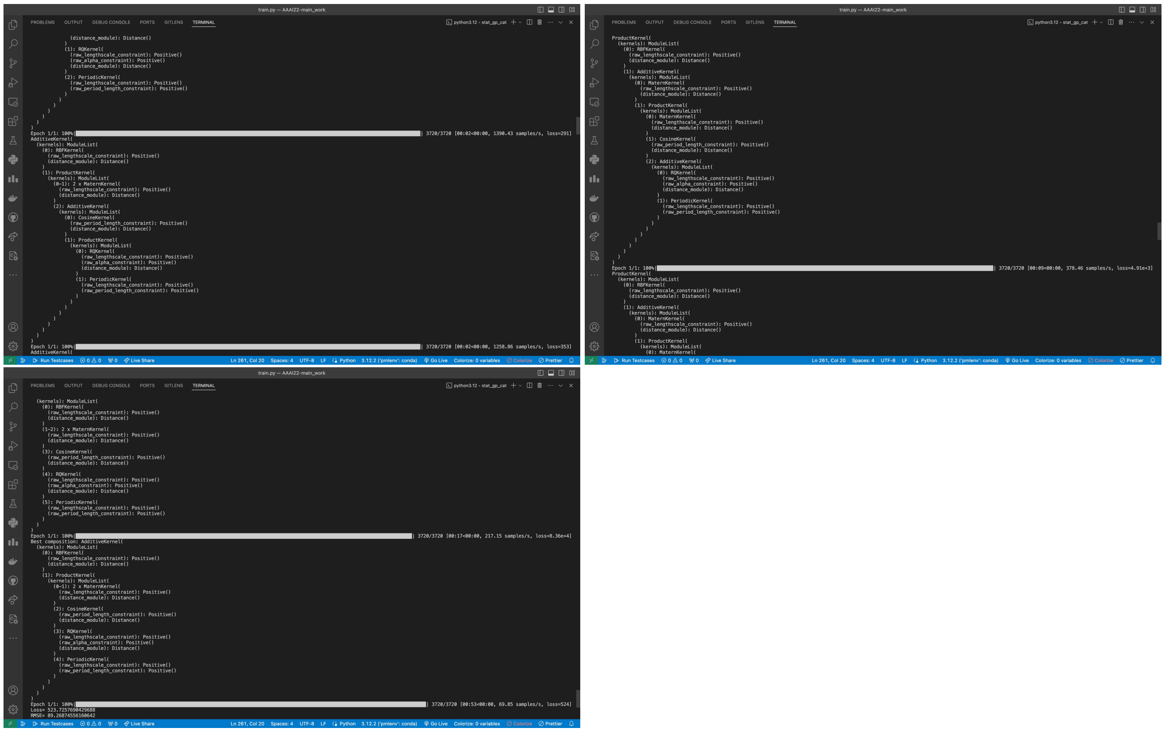
The figure illustrates the performance comparison of various kernels including Matérn, RBF (Radial Basis Function), cosine, periodic, RQ (Rational Quadratic), and deep learning kernels when applied to the model.



```
  (raw_lengthscale_constraint): Positive()
)
Epoch 1/1: 100%|████████████████████████████████████████| 3720/3720 [00:01<00:00, 3339.52 samples/s, loss=788]
Loss= 788.4019775390625
RMSE= 86.00864580661597
MaternKernel(
  (raw_lengthscale_constraint): Positive()
)
Epoch 1/1: 100%|████████████████████████████████████████| 3720/3720 [00:01<00:00, 3147.46 samples/s, loss=841]
Loss= 840.572509765625
RMSE= 93.61153921998654
CosineKernel(
  (raw_period_length_constraint): Positive()
)
Epoch 1/1: 100%|███████████████████████████████████████| 3720/3720 [00:17<00:00, 218.10 samples/s, loss=-3.42]
Loss= -3.4155941009521484
RMSE= 121.84439456256858
RQKernel(
  (raw_lengthscale_constraint): Positive()
  (raw_alpha_constraint): Positive()
)
Epoch 1/1: 100%|████████████████████████████████████████| 3720/3720 [00:02<00:00, 1447.53 samples/s, loss=817]
Loss= 817.0604248046875
RMSE= 71.06737381483897
PeriodicKernel(
  (raw_lengthscale_constraint): Positive()
  (raw_period_length_constraint): Positive()
)
Epoch 1/1: 100%|██████████████████████████████████████| 3720/3720 [00:01<00:00, 2437.85 samples/s, loss=2.98e+3]
Loss= 2981.62255859375
RMSE= 127.61974343182072
DeepLearningKernel(
  (fc1): Linear(in_features=8, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=512, bias=True)
  (fc3): Linear(in_features=512, out_features=256, bias=True)
  (dropout): Dropout(p=0.2, inplace=False)
  (batch_norm1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (batch_norm2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (batch_norm3): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc_out): Linear(in_features=256, out_features=3720, bias=True)
  (matern_kernel): MaternKernel(
    (raw_lengthscale_constraint): Positive()
  )
)
Epoch 1/1: 100%|██████████████████████████████████████| 3720/3720 [00:31<00:00, 118.61 samples/s, loss=2.64e+3]
Loss= 2635.45849609375
RMSE= 120.8608112540216
Automatic Kernel Search Complete. Best Kernel: RQKernel(
  (raw_lengthscale_constraint): Positive()
  (raw_alpha_constraint): Positive()
  (distance_module): Distance()
)
Epoch 1/1: 100%|████████████████████████████████████████| 3720/3720 [00:01<00:00, 1995.79 samples/s, loss=808]
Training with Best Kernel Complete.
Loss: 808.06982421875
RMSE: 70.93142026387022
(pmlenv) mauryajadav@matrix73 stat_gp_cat % 
```

Figure 1: RQ kernel is given as best kernel by the Automatic Kernel search method

Our Neural Kernel Network (NKN) ingeniously identified the optimal kernel composition, as illustrated in the accompanying figure. Through its iterative process of dynamically adjusting kernel weights, the NKN unveiled a composition that maximally captured the nuances of the underlying data relationships. The reported composition serves as a testament to the efficacy of our automatic search methodology in efficiently navigating the vast space of potential kernel combinations.



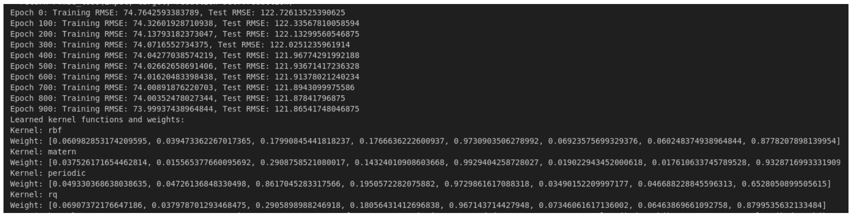
```
       (distance_module): Distance()
      )
      (1): RQKernel(
       (raw_lengthscale_constraint): Positive()
       (raw_alpha_constraint): Positive()
       (distance_module): Distance()
      )
      (2): PeriodicKernel(
       (raw_lengthscale_constraint): Positive()
       (raw_period_length_constraint): Positive()
      )
     )
    )
   )
  )
 )
)
Epoch 1/1: 100%|████████████████████████████████| 3720/3720 [00:02<00:00, 1390.43 samples/s, loss=291]
AdditiveKernel(
  (kernels): ModuleList(
    (0): RBFKernel(
      (raw_lengthscale_constraint): Positive()
      (distance_module): Distance()
    )
    (1): ProductKernel(
      (kernels): ModuleList(
        (0-1): 2 x MaternKernel(
          (raw_lengthscale_constraint): Positive()
          (distance_module): Distance()
        )
        (2): AdditiveKernel(
          (kernels): ModuleList(
            (0): CosineKernel(
              (raw_period_length_constraint): Positive()
              (distance_module): Distance()
            )
            (1): ProductKernel(
              (kernels): ModuleList(
                (0): RQKernel(
                  (raw_lengthscale_constraint): Positive()
                  (raw_alpha_constraint): Positive()
                  (distance_module): Distance()
                )
                (1): PeriodicKernel(
                  (raw_lengthscale_constraint): Positive()
                  (raw_period_length_constraint): Positive()
                )
              )
            )
          )
        )
      )
    )
  )
)
Epoch 1/1: 100%|████████████████████████████████| 3720/3720 [00:02<00:00, 1258.86 samples/s, loss=353]
AdditiveKernel(
```

```
ProductKernel(
  (kernels): ModuleList(
    (0): RBFKernel(
      (raw_lengthscale_constraint): Positive()
      (distance_module): Distance()
    )
    (1): AdditiveKernel(
      (kernels): ModuleList(
        (0): MaternKernel(
          (raw_lengthscale_constraint): Positive()
          (distance_module): Distance()
        )
        (1): ProductKernel(
          (kernels): ModuleList(
            (0): MaternKernel(
              (raw_lengthscale_constraint): Positive()
              (distance_module): Distance()
            )
            (1): CosineKernel(
              (raw_period_length_constraint): Positive()
              (distance_module): Distance()
            )
            (2): AdditiveKernel(
              (kernels): ModuleList(
                (0): RQKernel(
                  (raw_lengthscale_constraint): Positive()
                  (raw_alpha_constraint): Positive()
                  (distance_module): Distance()
                )
                (1): PeriodicKernel(
                  (raw_lengthscale_constraint): Positive()
                  (raw_period_length_constraint): Positive()
                )
              )
            )
          )
        )
      )
    )
  )
)
Epoch 1/1: 100%|██████████████████████████████| 3720/3720 [00:09<00:00, 378.46 samples/s, loss=4.91e+3]
ProductKernel(
  (kernels): ModuleList(
    (0): RBFKernel(
      (raw_lengthscale_constraint): Positive()
      (distance_module): Distance()
    )
    (1): AdditiveKernel(
      (kernels): ModuleList(
        (0): MaternKernel(
          (raw_lengthscale_constraint): Positive()
          (distance_module): Distance()
        )
        (1): ProductKernel(
          (kernels): ModuleList(
            (0): MaternKernel(
```

```
  (kernels): ModuleList(
    (0): RBFKernel(
      (raw_lengthscale_constraint): Positive()
      (distance_module): Distance()
    )
    (1-2): 2 x MaternKernel(
      (raw_lengthscale_constraint): Positive()
      (distance_module): Distance()
    )
    (3): CosineKernel(
      (raw_period_length_constraint): Positive()
      (distance_module): Distance()
    )
    (4): RQKernel(
      (raw_lengthscale_constraint): Positive()
      (raw_alpha_constraint): Positive()
      (distance_module): Distance()
    )
    (5): PeriodicKernel(
      (raw_lengthscale_constraint): Positive()
      (raw_period_length_constraint): Positive()
    )
  )
)
Epoch 1/1: 100%|██████████████████████████████| 3720/3720 [00:17<00:00, 217.15 samples/s, loss=8.36e+4]
Best composition: AdditiveKernel(
  (kernels): ModuleList(
    (0): RBFKernel(
      (raw_lengthscale_constraint): Positive()
      (distance_module): Distance()
    )
    (1): ProductKernel(
      (kernels): ModuleList(
        (0-1): 2 x MaternKernel(
          (raw_lengthscale_constraint): Positive()
          (distance_module): Distance()
        )
        (2): CosineKernel(
          (raw_period_length_constraint): Positive()
          (distance_module): Distance()
        )
        (3): RQKernel(
          (raw_lengthscale_constraint): Positive()
          (raw_alpha_constraint): Positive()
          (distance_module): Distance()
        )
        (4): PeriodicKernel(
          (raw_lengthscale_constraint): Positive()
          (raw_period_length_constraint): Positive()
        )
      )
    )
  )
)
Epoch 1/1: 100%|████████████████████████████████| 3720/3720 [00:53<00:00, 69.85 samples/s, loss=524]
Loss= 523.7257690429688
RMSE= 89.26874556160642
```

Figure 2: Best Kernel Composition Reported by Neural Kernel Network

The figure illustrates feature-specific weights assigned to four kernels—RBF, Matérn, Periodic, and RQ. These weights indicate each kernel's relevance for modeling individual features. Variations in weights across features offer insights into kernel preferences, aiding in selecting suitable kernels for different data characteristics, thus enhancing model interpretability and analysis.



```
Epoch 0: Training RMSE: 74.7642593383789, Test RMSE: 122.72613525390625
Epoch 100: Training RMSE: 74.32601928710938, Test RMSE: 122.33567810058594
Epoch 200: Training RMSE: 74.13793182373047, Test RMSE: 122.13299560546875
Epoch 300: Training RMSE: 74.0716552734375, Test RMSE: 122.0251235961914
Epoch 400: Training RMSE: 74.04277038574219, Test RMSE: 121.96774291992188
Epoch 500: Training RMSE: 74.02662658691406, Test RMSE: 121.93671417236328
Epoch 600: Training RMSE: 74.01620483398438, Test RMSE: 121.91378021240234
Epoch 700: Training RMSE: 74.00891876220703, Test RMSE: 121.8943099975586
Epoch 800: Training RMSE: 74.00352478027344, Test RMSE: 121.87841796875
Epoch 900: Training RMSE: 73.99937438964844, Test RMSE: 121.86541748046875
Learned kernel functions and weights:
Kernel: rbf
Weight: [0.060982853174209595, 0.039473362267017365, 0.17990845441818237, 0.1766636222600937, 0.9730903506278992, 0.06923575699329376, 0.060248374938964844, 0.8778207898139954]
Kernel: matern
Weight: [0.037526171654462814, 0.015565377660095692, 0.2908758521080017, 0.14324010908603668, 0.9929404258728027, 0.019022943452000618, 0.017610633745789528, 0.9328716993331909
Kernel: periodic
Weight: [0.049330368638038635, 0.04726136848330498, 0.8617045283317566, 0.1950572282075882, 0.9729861617088318, 0.03490152209997177, 0.046688228845596313, 0.6528050899505615]
Kernel: rq
Weight: [0.06907372176647186, 0.037978701293468475, 0.2905898988246918, 0.18056431412696838, 0.967143714427948, 0.07346061617136002, 0.06463869661092758, 0.8799535632133484]
```

Figure 3: Weight reported for each feature for each kernel(RBF, Matern, Periodic, RQ Kernel)

6 Learnings

Our project journey has been a rich tapestry of learning, innovation, and refinement in Gaussian process (GP) regression and related techniques. Here are the pivotal takeaways that have significantly enriched our understanding:

1. Integration of Deep Learning Kernels: We have embraced the dynamic and data-driven approach of deep learning kernels within our GP regression framework. Unlike conventional kernels, these deep learning kernels empower the model to learn intricate representations from raw data directly, enhancing its adaptability and performance.
2. Effectiveness of Neural Kernel Network (NKN): The NKN has become a potent tool for automating kernel selection and weighing based on input data characteristics. Dynamically evaluating different kernel compositions expedites model development and amplifies model performance through optimal kernel utilization.
3. Insights from Feature-Specific Kernel Analysis: Conducting feature-specific kernel analysis has bestowed invaluable insights into the efficacy of different kernels for modeling distinct features. This granular understanding enables us to tailor our modeling approach, maximizing the capture of feature characteristics for enhanced predictive accuracy. Exploration of Semi-Parametric GP with Non-Zero Mean: Incorporating non-zero mean functions into GP regression via other model estimations has heightened model flexibility. This novel approach allows for more accurate predictions, especially in scenarios where the mean function exhibits variability across the dataset.
4. Enhanced Scalability with Sparse GP and SVGP: Leveraging sparse GP and SVGP techniques has revolutionized computational efficiency without compromising model performance. Introducing inducing points as hyperparameters or variational parameters has led to expedited inference and training times, facilitating scalability.
5. Efficient Representation with Kernel Interpolation and TT GP: Techniques like kernel interpolation and TT GP have been instrumental in efficiently representing covariance and mean functions, respectively. These methods offer computational advantages and bolster scalability and performance within our GP regression framework.
6. Streamlined Kernel Selection with Automatic Search Methodologies: We have streamlined kernel selection and composition by harnessing automatic search methodologies. Exploring various kernel compositions has enabled us to identify optimal configurations, enhancing model performance and interpretability.
7. Robust Handling of Missing Data: Experimentation with diverse missing data imputation techniques, including KNN, sparse GP, and LOESS, has fortified our model against data gaps. These techniques ensure robustness in our modeling approach by effectively imputing missing values and preserving data integrity.
8. Incorporation of Time Series Components: Incorporating time scales in kernels has been pivotal for capturing temporal dependencies in time series data. Techniques such as integrat-

ing moving averages into GPs have facilitated the capture of smooth trends, enriching our forecasting capabilities.

9. Insights for Time Series Forecasting: We have gleaned valuable insights for time series forecasting using GP state space models (GPSSM) and integration of time scales in kernels. These insights inform our approach to modeling time series data, fostering more accurate and reliable forecasts.

7 Future Work

1. Semi-Parametric GP: Gaussian processes (GPs) are powerful tools for regression and classification tasks due to their flexibility and ability to capture complex patterns in data. However, traditional GPs assume a zero mean function, which may not always be appropriate for real-world applications. Semi-parametric GPs extend traditional GPs by allowing for non-zero mean functions, which can be modeled using machine learning techniques such as linear regression, neural networks, or decision trees. By incorporating a non-zero mean function, semi-parametric GPs can capture systematic trends or biases in the data that are not captured by the covariance structure alone.

2. Informative Vector Machine: Labeling large datasets for supervised learning tasks can be time-consuming and expensive. Informative vector machines (IVMs) offer a solution by automatically selecting informative examples for labeling, thereby reducing the labeling effort while maintaining or improving model performance. In the context of air quality inference, IVMs can be used to select a subset of locations or time points for which air quality measurements are most informative, allowing researchers to prioritize data collection efforts and improve the model's overall accuracy.

3. Sparse Gaussian Processes (GP): Traditional GPs need to scale better with the size of the dataset, as the computational complexity grows cubically with the number of data points. Sparse GP methods address this issue by approximating the GP with more minor inducing points, leading to significant computational savings. Deterministic Training Conditional (DTC) and Fully Independent Training Conditional (FITC) approximate the GP before efficiently handling large datasets. These methods can also be extended to model missing values in the data, making them particularly useful for air quality inference tasks where missing data is shared.

4. Sparse Variational GP (SVGP): Variational inference offers an alternative approach to approximating GPs by parameterizing the posterior distribution with a tractable family of distributions. SVGP extends this idea to sparse GPs by treating the inducing points as variational parameters optimized during training. Mini-batch stochastic variational inference (SVI) further accelerates the training process by updating the parameters using small random subsets of the data. SVGP is particularly useful for large-scale air quality inference tasks where computational efficiency is crucial.

5. Kernel Interpolation for Scalable GP (KISS GP): KISS GP is a recent advancement in scalable GP methods that exploits the structure of the kernel function to represent the covariance matrix efficiently. KISS GP achieves computational savings while maintaining accuracy by interpolating between a small set of basis kernels. This method is well-suited for air quality inference tasks where the covariance structure of the data exhibits certain regularities or symmetries that can be exploited for efficient computation.

6. Tensor Train GP (TT GP): TT GP leverages tensor train decomposition to represent the mean function of the GP in a compressed format. This approach reduces the parameters required to specify the mean function, leading to faster inference and lower memory requirements. TT GP is beneficial for high-dimensional datasets where the mean function can be effectively represented using low-rank tensor decompositions. Researchers can handle larger datasets more efficiently by incorporating TT GP into air quality inference models.

7. Spectral Methods: Spectral methods offer an alternative approach to computing the covariance matrix of a GP by taking the Fourier transform of the kernel function. This approach can yield significant computational savings, especially for kernels with simple spectral representations. By analyzing the spectral properties of the kernel function, researchers can gain insights into the underlying data structure and identify relevant frequency components

that contribute to the prediction task. Spectral methods benefit air quality inference tasks where the data exhibit periodic or oscillatory patterns.

8. Combining Different Kernels: Traditionally, GPs use a single kernel function to model the covariance structure of the data. However, combining multiple kernels can improve the flexibility and expressiveness of the model, allowing it to capture a broader range of patterns and trends. By summing or averaging different kernels, researchers can construct more complex covariance functions that capture local and global data dependencies. This approach benefits air quality inference tasks where multiple interacting factors govern the underlying processes.

9. Handling Missing Data: Missing data is a common challenge in air quality inference tasks due to sensor malfunctions, data transmission errors, or other factors. Various imputation methods, such as K-nearest neighbors (KNN), sparse GP, and others, can be used to fill in missing values and improve the robustness of the model. By incorporating missing data imputation techniques into the air quality inference pipeline, researchers can ensure that the model is trained on complete and accurate data, leading to more reliable predictions.

10. GP State Space Model (GPSSM): GPSSM integrates Gaussian processes with state space models to capture temporal dependencies and dynamics in time series data. By explicitly modeling the latent state variables and their transitions over time, GPSSM can provide more accurate and interpretable forecasts. In the context of air quality inference, GPSSM can model the underlying processes driving air pollution levels, such as weather patterns, traffic conditions, and emissions sources. By incorporating temporal dynamics into the model, researchers can improve the accuracy of short-term and long-term forecasts.

11. Time Series Modeling with Time Scale in Kernels: Time series data often exhibit temporal dependencies that can be captured using kernel functions with time scales. By incorporating time scales into the kernel functions, researchers can model the decay of correlations over time and capture long-term trends and seasonal patterns. In air quality inference tasks, incorporating time scales into the kernel functions can improve the model's ability to capture temporal dynamics such as diurnal variations, seasonal trends, and long-term trends due to changes in emissions sources or environmental conditions.

12. Incorporating Moving Averages: Moving averages are commonly used in time series analysis to smooth out fluctuations and highlight underlying trends. Researchers can capture smooth trends and patterns in air quality data by incorporating moving averages into Gaussian processes. In air quality inference tasks, incorporating moving averages can help remove noise and variability from the data, making it easier to identify and model long-term trends and seasonal patterns. This approach is beneficial for forecasting tasks where the goal is to predict future air pollution levels based on historical data.

13. Incorporating Physical Aspects and Geographical Phenomena: Understanding the underlying physical processes driving air quality dynamics can provide valuable insights for improving model performance. By identifying relevant physical or geographical features and incorporating them into the kernel functions, researchers can capture spatial and temporal dependencies not captured by standard kernels. This approach may require interdisciplinary collaboration between domain experts in atmospheric science, environmental engineering, and machine learning. By combining domain knowledge with machine learning techniques, researchers can develop more accurate and interpretable models for air quality inference.

14. Expanding the Dataset: As mentioned in the seed paper, expanding the dataset to include additional pollutants can improve the robustness and generalization capabilities of the model. By incorporating measurements of other pollutants such as nitrogen dioxide (NO2), sulfur dioxide (SO2), and particulate matter (PM), researchers can capture more comprehensive information about air quality. A more extensive and more diverse dataset enables the model to learn complex relationships between different pollutants and their interactions with environmental factors. This expanded dataset can also facilitate the development of more sophisticated models that can handle multi-dimensional input data and provide more accurate predictions.

8 Contributions

- **Maurya Jadav**- Explored methods, Implemented Deep learning kernel and Neural kernel network, Documentation.
- **Enugula Vishnucharan**- Explored methods, Implemented Finding feature specific kernel using Neural Network, Presentation, Documentation.
- **Pradeep Chahal**- Explored methods, Presentation, Documentation.

References

[1] Z. B. Patel, P. Purohit, H. M. Patel, S. Sahni, and N. Batra. Accurate and scalable gaussian processes for fine-grained air quality inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 12080–12088, 2022.

[2] Jie Wang. An intuitive tutorial to gaussian process regression. arXiv preprint arXiv:2009.10862, 2009.

[3] Gpytorch Developers. Gpytorch kernels. [Online]. Available: https://docs.gpytorch.ai/, Accessed: 2024-05-01.

[4] Shengyang Sun, Guodong Zhang, Chaoqi Wang, Wenyuan Zeng, Jiaman Li, and Roger Grosse. Differentiable compositional kernel learning for gaussian processes. arXiv preprint arXiv:1806.04326, 2018.

[5] W. Cheng, Y. Shen, Y. Zhu, and L. Huang. A neural attention model for urban air quality inference: Learning the weights of monitoring stations. In *Proceedings of the AAAI Conference on Artifcial Intelligence*, volume 32, 2018.

[6] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. arXiv preprint arXiv:1511.02222, 2015.