CSE 5835 - Bayesian Optimization for Material Science Paper Recreation Final Report

Sarah Tucker, Kip Hunt

December 2025

1 Introduction

Bayesian optimization (BO) is the process of optimizing an objective function by starting with a small number of measurements to form a distribution from which an acquisition function can determine the next best location in feature space to perform a new measurement which both explores the unknown feature space and exploits the known feature space. Then the prior distribution can be updated with the new measurement as determined by the acquisition function in order to iterate this process and determine the optimal regions in feature space for the given objective function. The research article, Benchmarking the Performance of Bayesian Optimization across Multiple Experimental Materials Science Domains, analyzes the use of BO across 5 different materials science experiment datasets and makes comparisons of performance using different surrogate models for fitting the prior distribution and acquisition functions for analyzing the distribution and determining the direction of following measurements. The project outlined here attempts to perform the same analyses on specifically the P3HT dataset which aims to maximize the conductivity of a material given varying compositions. This is done by using the Gaussian Process with Automatic Relevance Determination (GP-ARD) surrogate model which consists of a Gaussian Process regressor with an anisotropic kernel allowing for independent lengthscales of the five feature dimensions over which measurements were varied. Comparisons are made to the use of a Random Forest surrogate model along with variations of the acquisition function using both the expected improvement (EI) and lower confidence bound (LCB) metrics.

2 Data

2.1 P3HT Experimental Dataset

The P3HT dataset is one of the five studied in the original research article which this project attempts to replicate. It contains 178 measurements of varying composite blends and carbon nanotube types. The amount of each of the five varied

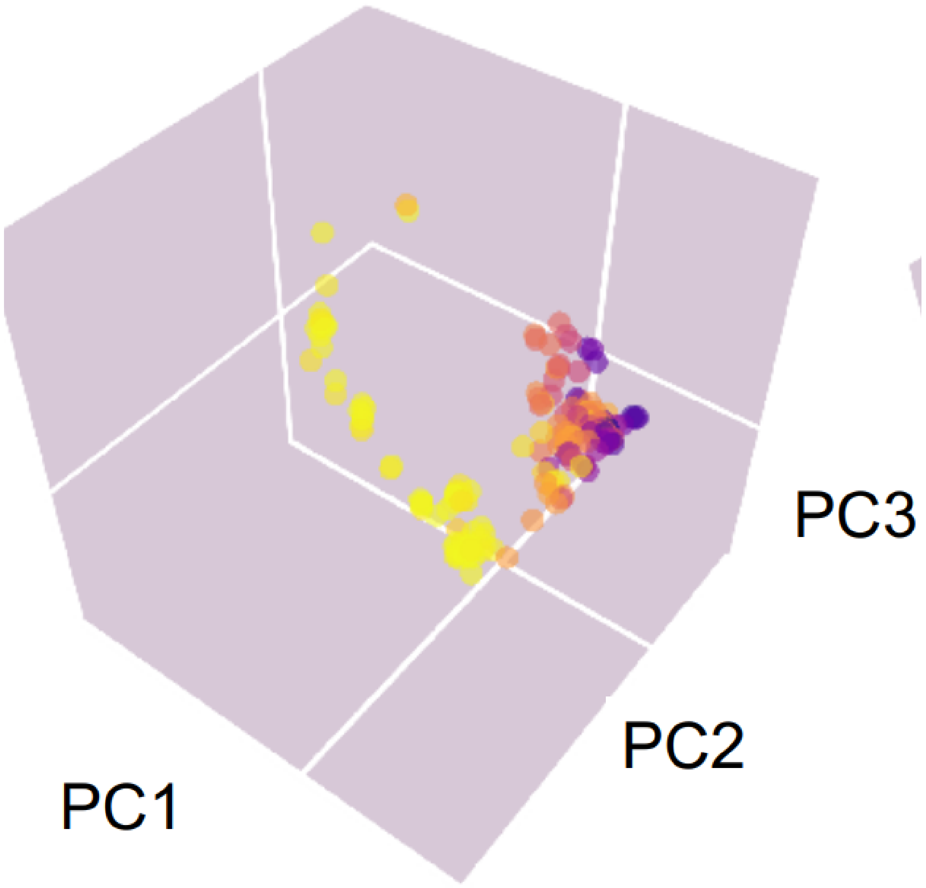

Figure 1: PCA reduction performed by original paper for visualization

material measurements of the composition makes up the five independent input features. The objective of this research was to maximize electrical conductivity of the composite material. The features are distributed continuously across five-dimensional feature space. In a PCA reduction to three dimension for visualization performed by the original research article shown in figure 1, moderate complexity was seen with potentially some clustering being visible, but nothing dramatic.

The labels (conductivity) of the dataset span a large region of label space, allowing for clear discrimination of a model's ability to find maximized objective values. Considered in their original units as shown in figure 2, the D2, D6, and D8 features (these are just different types of carbon nanotubes) are largely concentrated towards low values and have a few outlying groups of higher valued variations. In contrast, the D1 and P3HT contents have much wider variation, seemingly indicating that the original experiment required significantly more variance in these dimensions in order to maximize conductivity.

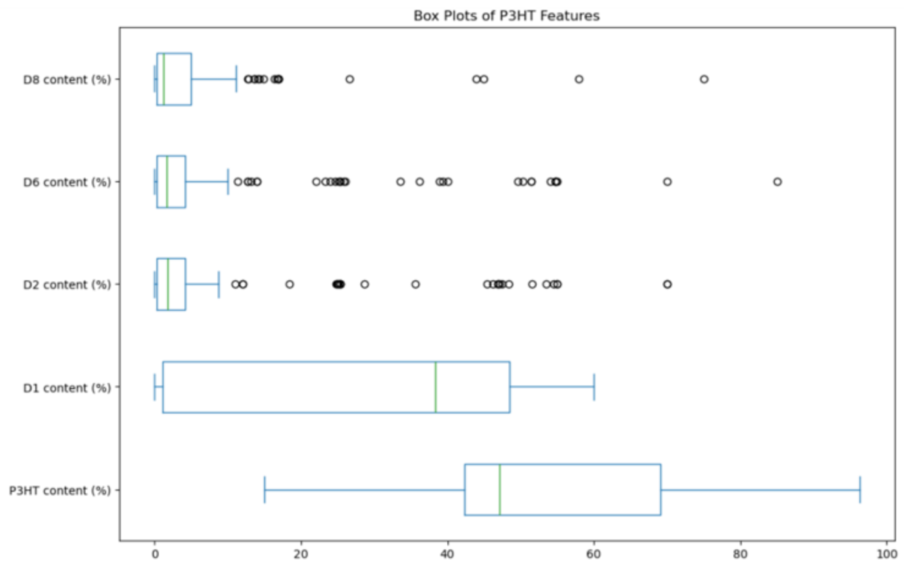

Figure 2: Box Plot of all five features

Upon finding the correlation matrix of the measurements as shown in figure 3, as expected given its wide distribution, the D1 content has a significant correlation with conductivity as shown in figure 4.

2.2 Data Preprocessing for Learning

The input features and labels were scaled to have a mean centered about zero and unit variance by using Scikit-Learn's standard scaler from the preprocessing library. As discussed further on in the learning algorithm section, the labels were all negated such that the measure of highest conductivity actually means the most negative label. This was done due to the optimizer used in this project's model performing optimization by default and it was more straightforward to simply negate the labels rather than trying to modify the optimizer.

3 Learning Algorithm

3.1 General Overview

To initialize the algorithm, a random sample of the dataset is taken to be a set of initial observations with varying features and known labels. This random set of observations comprise the initial prior distribution, thus the following

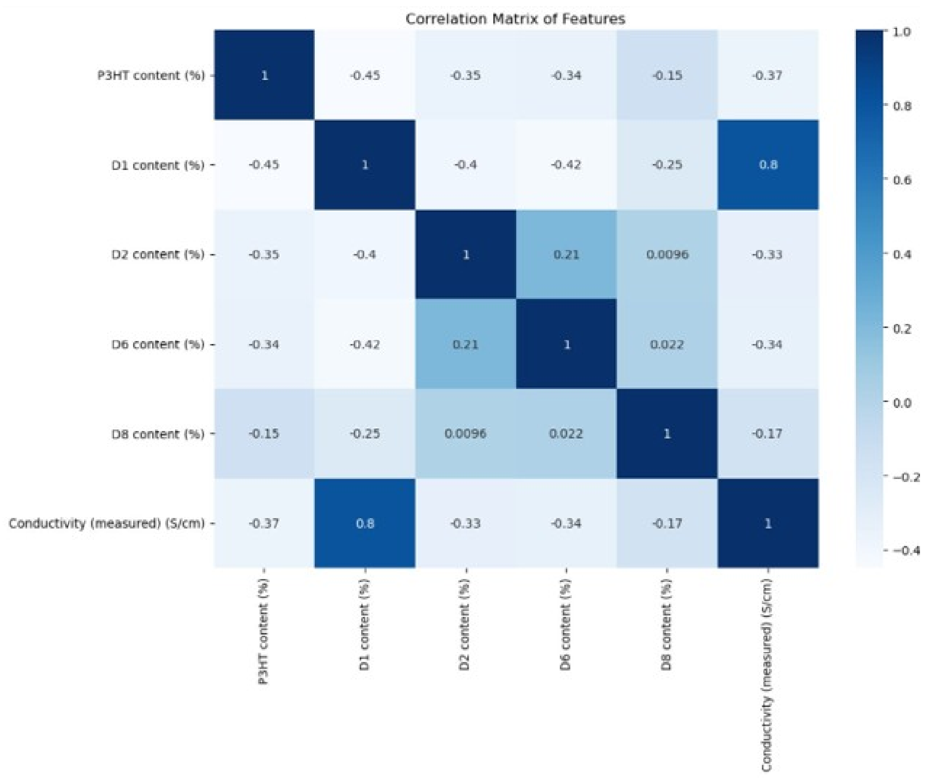

Figure 3: Correlation Matrix of all feature and the objective value, shows strong correlation between feature D1 content and Conductivity

steps are iterated over and make up the learning portion of the algorithm. The surrogate model is used to fit the prior distribution and predicted from this model are a mean and standard deviation. These characteristics are then fed into the acquisition function along with a ”best” observed label (in this case, highest conductivity) which is defined as the optimal objective value in the currently observed measurements. The acquisition functions used through this work include Expected Improvement(EI) which evaluates by how much a given point in feature space could improve the objective value compared to the best value of the prior distribution, and Lower Confidence Bound(LCB) which defines a lower limit on certain confidence intervals thus allowing for a reasonable range of exploration in a minimization problem given a certain prior distribution. Ideally, from here the acquisition function would determine the best position in feature space to perform the next measurement; however, this study is limited to only measurements which were already performed by an experiment. Therefore, the acquisition function uses these characteristics of the prior distribution to determine the next best measurement from the list of unobserved measurements available in the dataset. This is done by assigning an

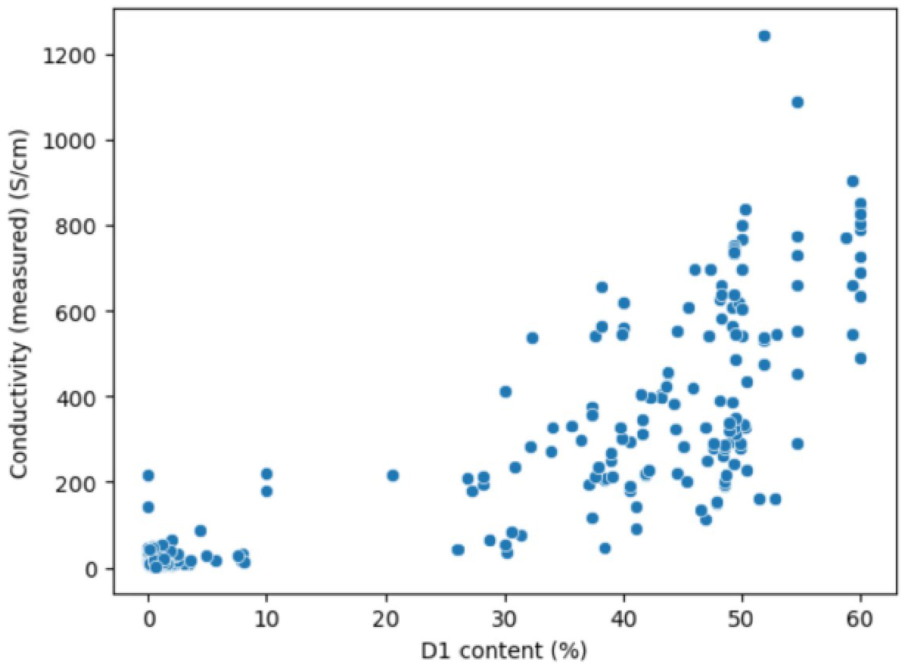

Figure 4: D1 content percentage feature vs conductivity

acquisition value to each unobserved measurement and then finding which yields the maximal acquisition value. This new measurement is then added to the list of observed measurements to form the updated posterior distribution which will in turn be used as the following iterations prior distribution. At the end of each iteration once a new measurement has been chosen by the acquisition function, this measurement is compared to the top five percent optimal measurements from the full dataset in order to determine if the algorithm was able to find an optimal follow-up measurement at its cycle in the learning process. This iterative process continues until the full dataset has been observed. It should be noted that the random sampling acquisition comparison goes through the exact same process, but rather than choosing a next best measurement based on an acquisition function, a measurement is randomly selected from the list of unobserved measurements.

3.2 Specifics to this project's implementation

The code used to implement this algorithm relies largely on scikit-learn libraries. For both the GP-ARD and Random Forest models, the scikit learn GaussianProcessRegressor and RandomForestRegressor functions (both of which use a bfgs based minimizer as the optimizer) along with their predict methods were used. In the case of the GP regressor, the lengthscale parameter is set to an array of five lengthscales in order for each feature dimension to have its own independent lengthscale thus allowing for the anisotropic kernel required for ARD. The kernel used in the case of the GP surrogate model is Scikit-Learn's Matern. The EI and LCB acquisition functions were both created manually by creating a function which takes in the prior distribution characteristics such as mean, standard deviation, and best observed label and then calculating the acquisition values as defined in the original article.

3.3 Hyperparameters

In the case of GP-ARD, the hyperparameters are found in the theta vector, which has an element for input feature dimension. Each of these elements are the lengthscale for their respective input feature. These independent lengthscales are what allow the kernel to be anisotropic, thus the A in ARD. The relevance determination part comes about as a result of the lengthscales being related to the importance of each input feature relative to the objective function. A small lengthscale, or theta vector element, indicates that the kernel is effectively "zooming in" on that dimension and thus that input feature has greater relevance in capturing the variation of the objective function. This is put simply in most of the literature by defining the relevance of a given feature as the inverse of its lengthscale. In the code implemented for this project, the theta vector is given as an argument to the Matern kernel, where it is initialized as a vector of ones (1.0's). The evolution of the lengthscales over the learning iterations can be seen by printing out or writing the kernel_.theta variable of the model, and over iterations the lengthscales will increase or decrease as a result of the model determining which input features have greater or lesser weight in determining the variation of the objective function. The initialization was left as ones rather than attempting to make informed guesses about a better initialization based on previous learning cycles as to not bias any of the learning with information not yet observed by the model in a given learning cycle. Ideally, an estimation of each input's relative relevance could be made and an anisotropic initialization could be given to the kernel, but sticking to the vector of ones seemed to work fine for this project's purposes and it was the method implemented in the research article which this project attempts to reproduce.

In the case of Random Forest, the hyperparameters include the number of trees, max depth of each tree, and the inclusion of bootstrapping in the sampling process when building the trees. This project found, similarly to what was described in the original research article, that the performance of the RF surrogate model was largely dependent upon the number of trees used. When using around 150 or more, the performance seemed stable.

4 Model Performance

4.1 Top% vs Learning Cycle Comparisons

To get a sense of each model's performance relative to each other's, the first metric under review is the measure of top% vs learning cycle. Top% is the percentage of measurements chosen to be performed by the BO process which lie in the top five percent of highest conductivity measurements available from the total dataset. Recall from the data overview that the feature of D1 content percentage had an extremely high correlation with the conductivity, and is therefore a reasonable candidate to contain most of the relevant dependence of the conductivity. Considering this, a comparison was made between the top% over all learning cycles for each model (both GP-ARD and RF with either EI or LCB as the acquisition function) with all five input features given to the model and the top% of the same models but with only the D1 content percentage feature given to them. The question at hand here really is, does D1 content percentage contain all of the necessary information to find the highest conductivity material? In the case of GP-ARD this is really no longer ARD since the kernel in question is only one dimensional and thus cannot be anisotropic. As seen in figure 5, for each model aside from RF with LCB as the acquisition function, the simple D1 fit was able to find the top five percent of highest conductivity measurements in fewer learning cycles. This seems to indicate that the D1 feature contains enough of the relevant information regarding what makes these materials conductive such that effectively fitting D1 to conductivity is more efficient than trying to find the contributions from all five features. This consideration was not brought up by the original research article, but it is somewhat ironic to see that potentially simplifying these models to basically fit a single variable may in some cases be more efficient in optimization even though the main drive of BO is to efficiently optimize the regions in which to search in feature space. Despite this result, this project will continue to analyze the use of BO with the models making use of all five features.

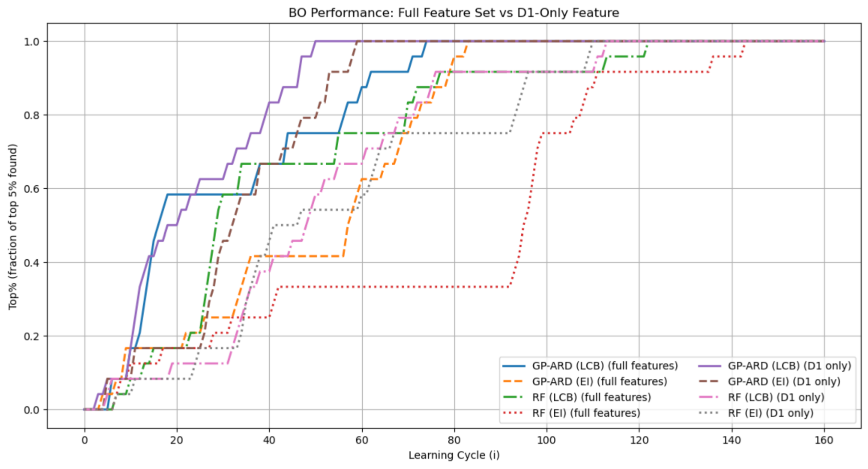

Figure 5: Comparison between the performance of all four model surrogates using only the D1 content feature versus their full feature counterparts.

Next under review as shown in figure 6, is the comparison of each model's top% evolution over learning cycles along with a comparison to simply choosing randomly ordered observations of the dataset. This is done to ensure that the models are able to, through their observations and respective acquisition functions, determine follow up measurements in a manner which efficiently identifies the regions of feature space which maximize conductivity. All models prove to be choosing measurements leading to higher conductivity much more efficiently than the random sampling which itself follows basically a consistent linear trend. The RF with EI as its acquisition function seems to get stuck for quite a while, allowing for the random sampling to briefly catch up, but then eventually corrects itself and sees a dramatic jump in top% observations. This comparison shows that GP-ARD with either acquisition function are the most efficient at finding the highest conductivity measurements, with RF using LCB performing comparably. Note that comparison to the original research articles results will be saved for the enhancement and acceleration factor discussion along with the challenges discussion for easier quantitative comparisons.

4.2 Enhancement and Acceleration Factor Comparisons

Enhancement Factor(EF) is the ratio of the top% value for a given model to the top% value using random sampling. Acceleration Factor(AF) is the ratio of the number of cycles needed for a model to reach a certain top% value to the number of cycles needed for random sampling to reach that same top% value.

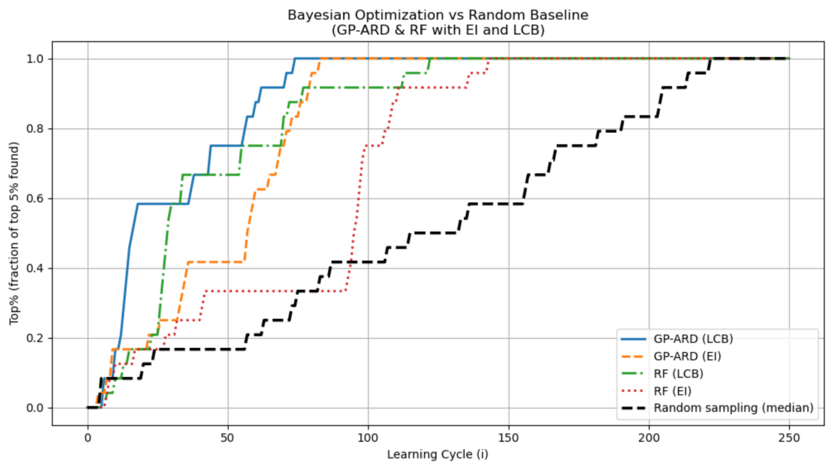

Figure 6: Performance of Bayesian Optimization of all four models.

While these metrics essentially contain the same information found in the top% vs learning cycle plots, they are helpful in explicitly showing the efficiency advantages of BO when compared to just randomly choosing experiments to jump around between. EF quantifies how much better BO is at choosing good candidates over given learning cycles, while AF shows how much faster BO can choose a given quality of candidates.

For the enhancement factor evaluations vs learning cycle shown in figure 7, generally for most models, rising enhancement can be seen until a peak is reached at which point BO begins to slow down in finding the highest conductivity candidates. Similarly to the analysis of top% vs cycle, GP-ARD with the LCB acquisition function dominates in terms of this metric, where it is able to reach the highest EF (seven times more top five percent candidates found compared to random sampling) in fewer learning cycles than any other model. RF with LCB and GP-ARD with EI perform a bit more comparably and reach an EF of about four in later learning cycles. Once again, RF with EI can be seen to perform relatively poorly with an EF dipping below one at the point where it gets stuck and random sampling briefly has a higher top% value. It is worth noting that in comparison to the original research results, similar results are seen when comparing the respective models (although RF with EI doesn't perform quite as poorly), however the EF calculated in this project is about half as large as that shown in the original article. This is to be expected given the slower increase of top% values at similar learning cycles seen in this project's results. Further discussion and conjecture on why this may be is saved for the challenges discussion.

The AF plot shown in figure 8 tells the same story as the EF and top% plots, where GP-ARD with LCB has significantly higher acceleration than the

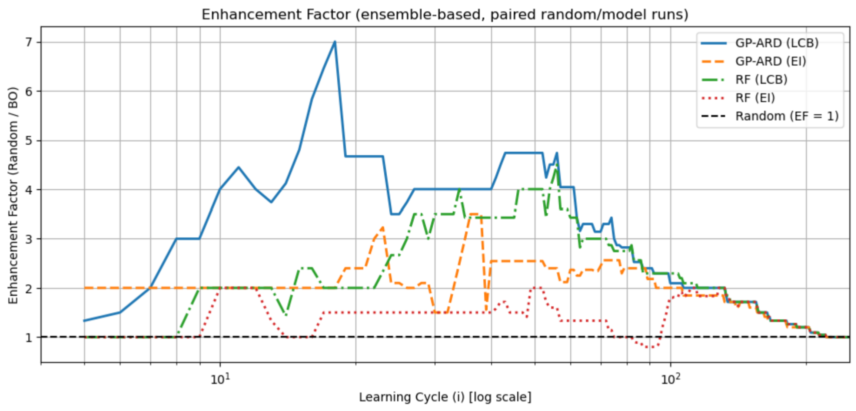

Figure 7: Enhancement factor vs learning cycle for all evaluated models and acquisition functions

other models up until a top% value of roughly 0.6 where diminishing returns begins to take over and its AF becomes comparable to that of RF with LCB and GP-ARD with EI. This result is quite similar to that of the original research, however once again the AF factor found there is roughly twice what is seen from this project's result.

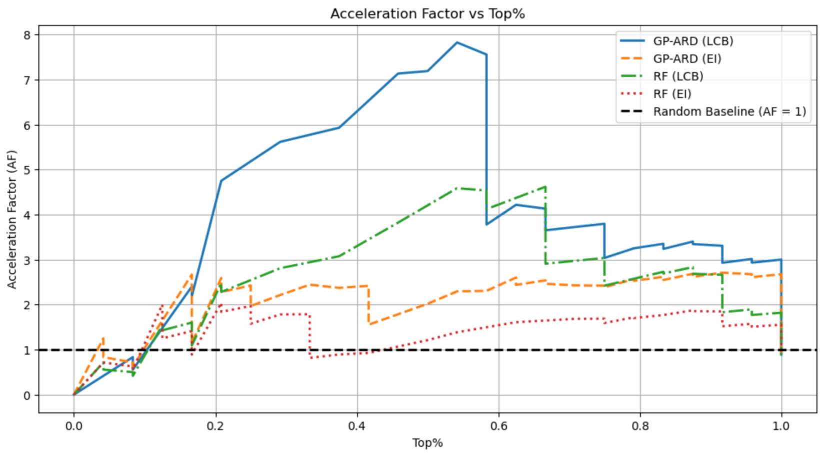

Figure 8: Acceleration factor vs top% value for all evaluated models and acquisition functions

4.3 BO Data Observation and Acquisition Ordering

For simple visualization of all data point's conductivity, PCA was performed in order to reduce the features down to one dimension. In figure 9, the acquisition order can be seen via a color-coded heatmap (thanks to Ashley from our class for inspiring the idea of this type of plot), where the color axis indicates the observation order. From this, a clear grouping can be seen at the higher cycle observations (only shown up to the 50th cycle) where the acquisition function is choosing points with very high conductivity (more negative, due to negating all labels in preproccessing, but data is left unscaled) as intended. Similarly in figure 10 (where conductivity is negated and scaled), BO is choosing to observe points with higher conductivity. This plot is interesting, as it shows points on the higher end of the PCA scale (onto which all data is projected) are not being observed. Perhaps this is due to some correlation between features unseen by the PCA projection which causes these points in the full five dimensional feature space to be suboptimal from the perspective of the acquisition functions despite their high conductivity.

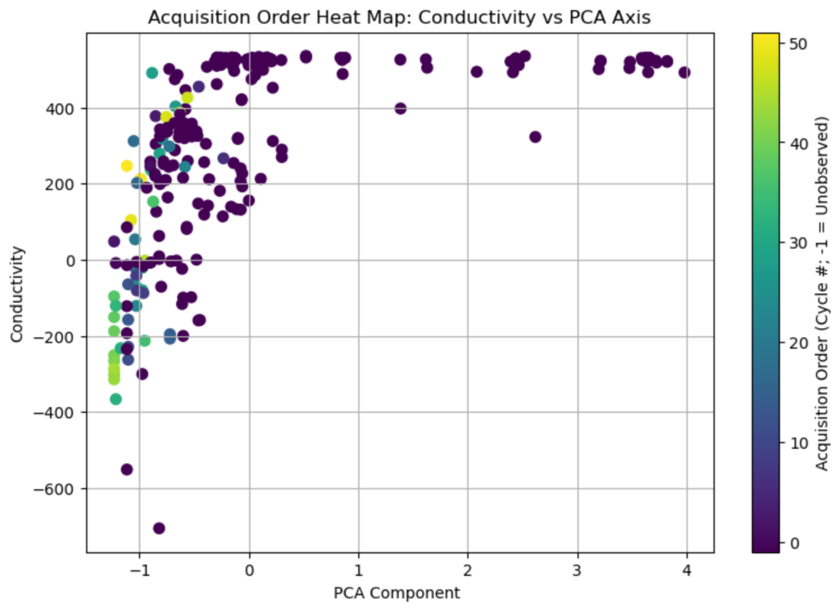

Figure 9: Plot that shows the order at which point were chosen during the Bayesian Optimization algorithm, the heat map shows in what order they were picked.

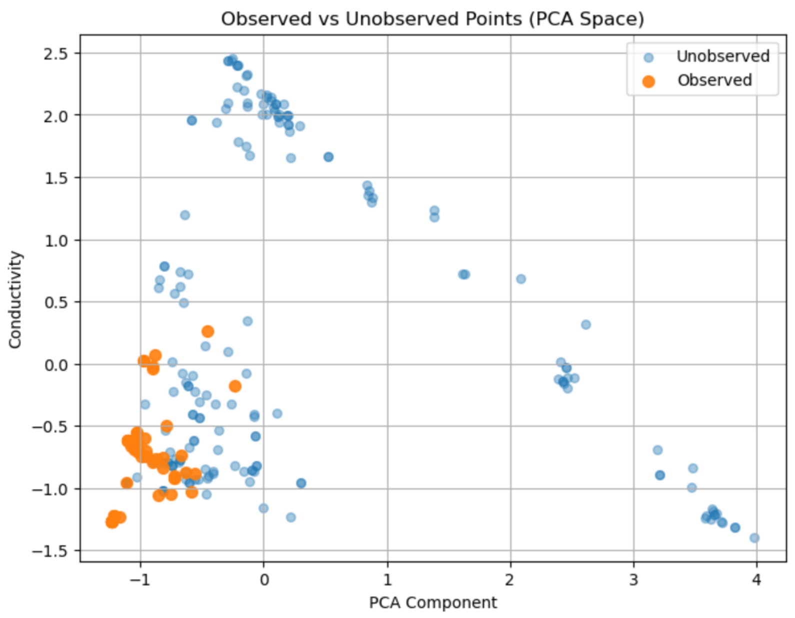

Figure 10: This shows the points that were observed by the Bayesian Optimization algorithm and what points were left unobserved.

5 Challenges and Advantages of Bayesian Optimization

BO is a good candidate for a materials science study like this due to the low dimensionality of input features which need to be efficiently explored. In the case of GP-ARD the initialization and hyperparameter tuning was fairly simple and straightforward. The lengthscale vector is an intuitive extension of the kernel from standard GP. Luckily, the use of GP in general is well documented through the Scikit-Learn tutorials. RF is very well documented as well, and was fairly straightforward once using enough trees. The primary difficulty encountered in this project was handling acquisition functions after negating the labels due to using a minimizer in both the GP-ARD and RF models. There was initially some confusion on how to handle the output of the surrogate model such as mean and standard deviation, but it was quickly apparent that no matter the sign of the prior distribution, the acquisition value should be maximized in order to find the best follow up measurement. A similar line of confusion serendipitously led to an initial result in which the least conductive materials were discovered by the models with similar efficiency to the results presented here, which turned out to be quite helpful in understanding the entire pipeline from prior distribution to model prediction to acquisition and finally to the posterior distribution.

In comparison to the results from the original research paper which this work attempts to recreate, general characteristics are largely the same but with the key discrepancy of slower growth of the top% values of all models. When comparing the code used between this work and the original, all seems equal mechanically with the key difference being the use of the GPy and PyTorch learning libraries for the original work and Scikit-Learn learning libraries for this work. Upon attempting to run the original research's code available on github, there were several issues which seemingly required updated use of libraries, so direct comparison of the functionality was difficult. This work attempted to use the same hyperparameters and initializations as the original, however this could be a major source of the discrepancy since models can be quite sensitive to these inputs. Another possibility is that the differing libraries have implementations under the hood with differences which are not understood. For example, the difference between the Gpy and Scikit-Learn Matern kernels or the additional noise kernel added to the Matern kernel. With the available documentation, these kernels and other parameters used seem to be comparable, but slight differences over the course of the framework could add up to account for some of the performance discrepancies.

References

[1] PV-Lab. *Benchmarking the Performance of Bayesian Optimization.* `https://github.com/PV-Lab/Benchmarking`. Accessed: 2025-12-11.

[2] Qiaohao Liang et al. "Benchmarking the Performance of Bayesian Optimization Across Multiple Experimental Materials Science Domains". In: (2021). Preprint; PDF provided by user.

[3] Scikit-learn Developers. *Scikit-learn Documentation.* `https://scikit-learn.org/stable/index.html`. Accessed: 2025-12-11.