Comparative Bandpass Filter Design in HFSS and CST with Machine Learning

Prabhroop Singh and Kalyana Abenanth Gurunathan
Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada
prabhro1@ualberta.ca kalyanaa@ualberta.ca

***Abstract*—This project presents a comparative study of bandpass filter design using Ansys HFSS and CST Studio Suite combined with machine learning to accelerate design exploration and optimization. A common parameterized bandpass filter topology is implemented in both tools under diverse electrical specifications, including target center frequency, bandwidth, and passband return and insertion loss requirements. For each solver, datasets are generated via automated parameter sweeps, and performance metrics are extracted from the simulated S-parameters which include center frequency, 3dB bandwidth, insertion los, and return-loss measures. These datasets are then used to train regression based machine learning models that map geometric parameters to filter performance and enable rapid screening of candidate designs without repeated full-wave simulation. Model accuracy is evaluated using standard regression metrics such as R^2, MAE, MSE and RMSE. The HFSS results suggest that additional data and broader sampling are needed, whereas the CST results achieve higher R^2 values but indicate potential overfitting. Results are compared in terms of surrogate prediction accuracy and the simulation effort required to reach specifications. Overall, the combined EM and machine-learning workflow reduces iteration cost while preserving simulation-validated performance. Also, the HFSS–CST comparison provides guidance for selecting tool settings and automation strategies for machine learning assisted microwave filter design.**

***Index Terms*—Machine Learning, HFSS, CST**

I. Introduction

THE design of microwave band-pass filters remains a fundamental task in radio-frequency and wireless communication systems, where stringent requirements on bandwidth, insertion loss, and impedance matching must be satisfied simultaneously. Planar microstrip filters are widely adopted due to their compact size, ease of fabrication, and compatibility with printed circuit technologies. However, achieving optimal filter performance often requires careful tuning of multiple geometric parameters, which can be both time-consuming and computationally expensive when relying solely on full-wave electromagnetic (EM) simulation tools.

Full-wave electromagnetic (EM) solvers such as Ansys HFSS and CST Studio Suite are industry-standard tools for accurately analyzing microwave structures[1]. By numerically solving Maxwell's equations, these solvers capture complex electromagnetic phenomena including coupling effects, resonant behavior, and radiation losses. While HFSS and CST provide highly reliable results, their computational cost can become prohibitive when extensive parametric sweeps or optimization loops are required. Each design iteration involves a complete EM simulation, which limits rapid exploration of the design space and increases overall development time.

To address these challenges, data-driven techniques have gained increasing attention as a complementary approach to physics-based simulation. Machine learning (ML), when trained on high-fidelity EM simulation data, can serve as a surrogate model that approximates [2] the relationship between geometric design parameters and performance metrics. Once trained, such models can generate predictions in milliseconds, enabling fast evaluation of candidate designs without repeatedly invoking a full-wave solver. This capability is particularly attractive for microwave filter design, where performance trends are often smooth but nonlinear with respect to geometry.

Despite the growing interest in ML-assisted EM design, several practical questions remain. The accuracy and reliability of surrogate models depend strongly on the quality and size of the training data, which must still be generated using computationally expensive solvers such as HFSS or CST. In addition, different ML models exhibit distinct trade-offs among accuracy, robustness, interpretability, and scalability. A systematic comparison of these models using realistic EM-generated datasets is therefore essential to assess their suitability for practical design workflows.

In this work, a hybrid EM–ML framework is developed for the design and analysis of a planar band-pass filter using both HFSS and CST Studio Suite. HFSS is employed to establish baseline electromagnetic behavior and validate filter performance, while CST [1] is used to generate large-scale datasets through automated parameter sweeps enabled by Python-based control. The resulting CST-generated data are used to train and evaluate multiple regression-based machine learning models, including linear baselines, nonlinear regressors, and ensemble-based approaches.

The main contributions of this paper are threefold. First, an automated EM simulation workflow is established using HFSS and CST to efficiently generate physically consistent datasets for machine learning. Second, a comprehensive evaluation of machine learning models is conducted across multiple dataset sizes, highlighting the strengths and limitations of each approach in predicting key filter performance metrics. Third, the practical impact of surrogate modeling is demonstrated by comparing ML predictions with full-wave EM results and a manual design baseline, illustrating the potential for significant reduction in design iteration time.

By integrating high-fidelity EM simulation with data-driven surrogate modeling, this work presents a practical pathway

toward accelerated microwave filter design. The proposed methodology leverages the accuracy of HFSS and CST while mitigating their computational cost, offering an efficient and reliable framework for modern RF and microwave engineering applications.

II. Theory and Background

In electronics and signal processing, a *filter* is typically modeled as a two port network that shapes a signal spectrum by passing desired frequency components while attenuating unwanted ones. A *bandpass filter* transmits frequencies within a specified interval (the *passband*) and suppresses components below and above that interval. This contrasts with a *low-pass filter*, which passes frequencies below a cutoff and a *high-pass filter*, which passes frequencies above a cutoff. Although filters are often realized as physical circuits, the same frequency-selection operation can also be implemented digitally using software-based filtering algorithms. The notion of a bandpass response also appears in other domains; for example, optical bandpass filters transmit a selected wavelength range (e.g., in photography and theatre lighting) and acoustic filters isolate a chosen band of sound frequencies.

In analogue circuits, a band-pass response is commonly obtained using an RLC network. More generally, band-pass behaviour can be achieved by cascading a high pass stage with a low pass stage so that only the overlapping frequency range is transmitted. The resulting output is often described as a *bandpass signal* which means that most of its energy is concentrated within a frequency band away from DC. An ideal bandpass filter would exhibit a perfectly flat passband (no gain variation), zero attenuation within the passband and complete rejection outside it. But practical filters transition gradually between passband and stopband. This transition is characterized by the *roll-off* and is commonly expressed in decibels per octave or per decade. While a steeper roll-off improves selectivity, it can also increase ripple in the passband or stopband [3]. Key performance parameters include the *cutoff frequencies* and the *bandwidth*, defined as the difference between the upper and lower cutoff frequencies. Selectivity is also captured by the *shape factor* and it compares the bandwidths measured at two different attenuation levels. Many practical systems require isolating a narrow band from a broadband signal containing multiple components, which further enables development of the circuit structures that effectively combine low pass and high pass characteristics within a single network. Bandpass filters can be implemented using inductors and capacitors depending upon frequency and fabrication constraints. Capacitor based reactive behavior is often preferred due to comparatively lower loss in many practical implementations [4].

A. HFSS and Machine Learning

Full wave electromagnetic solvers such as Ansys HFSS are widely used to evaluate ban-pass filters and antennas with high fidelity but repeated simulation during parameter sweeps and iterative tuning can be time intensive. As a result, recent research increasingly combines HFSS-driven datasets with machine learning to (i) construct surrogate models that approximate electromagnetic responses and (ii) guide optimization toward feasible designs with fewer expensive simulations.

Ke *et al.* present an automated synthesis framework for parallel-coupled microstrip bandpass filters that combines classical synthesis with machine learning assisted optimization and explicit incorporation of prior knowledge [5]. Their workflow starts from a normalized low-pass prototype and uses coupling-matrix concepts and external quality-factor targets to obtain a structured initial design. Rather than relying solely on data-driven search, the optimization loop incorporates electromagnetic and geometric constraints to avoid physically implausible solutions. The design is refined in stages and then improved using local Latin hypercube sampling around the initial point with an online-updated Gaussian process model and a lower-confidence-bound prescreening strategy [5]. MATLAB–HFSS automation is used to validate the approach on hairpin and interdigital examples. The results indicate that the synthesized designs satisfy typical specifications such as return loss, insertion loss, and stopband attenuation [5].

Beyond single-band designs, dual-band operation is often desired to reduce component count and footprint. Kuo *et al.* develop dual-passband microstrip bandpass filters using stepped-impedance resonators where the ratio between the first two resonant frequencies can be tuned by selecting impedance and electrical-length ratios [6]. To control bandwidth in both passbands, the design must satisfy coupling requirements at two distinct frequencies. This is addressed using fractional-bandwidth design graphs that map required coupling coefficients to coupled-line parameters in both bands while exploiting additional degrees of freedom (e.g. coupling length and gap) so that a single coupled section can realize appropriate coupling at both operating frequencies [6]. Prototypes in parallel-coupled and vertical-stacked configurations show good agreement between simulation and measurement and support the proposed synthesis methodology [6].

Machine learning is also used to accelerate antenna design by learning surrogate mappings from geometry to scattering response. In [7], a cylindrical dielectric resonator antenna is simulated in HFSS and a regression dataset is created by sweeping resonator height. Several regressors are compared for predicting the reflection coefficient across frequency. Similarly, Sharma *et al.* formulate antenna optimization as a supervised learning problem for a dual-band double T-shaped monopole antenna, where multiple models are trained on HFSS generated samples and then used for a dense design-space search. This enabled evaluation of a large number of candidate designs without additional simulations [8]. These studies highlight the general utility of surrogate modeling for speeding up parametric studies and optimization.

Recent work has also applied supervised learning directly to microstrip bandpass filter prediction and refinement. Kumar and Jhariya construct regression datasets for both S_{11} and S_{21} from HFSS parametric sweeps of key geometrical variables in a wideband microstrip bandpass filter. They also compare multiple regressors using standard error metrics and their results indicate that ensemble tree based models provide strong prediction accuracy [9]. In a similar direction, Javadi *et al.*

propose a coupling-matrix-driven workflow in which a regression model (emphasizing XGBoost) predicts key geometric parameters from target specifications, followed by full-wave validation and iterative refinement [10]. By integrating prior synthesis knowledge (e.g. separate handling of external and inter-resonator coupling and scaling/interpolation for resonator lengths) and using coupling-matrix extraction from simulated scattering parameters to guide tuning, their approach aims to reduce the number of optimization iterations required to reach a compliant design [10].

B. CST and Machine Learning

CST Studio Suite is a full-wave electromagnetic simulation platform based on numerical solutions of Maxwell's equations. It is widely used for the analysis and design of microwave, RF, and high-frequency structures, including planar transmission lines and resonant filters. CST supports multiple numerical solvers, such as the frequency-domain solver, which is particularly suitable for narrow-band microwave filter analysis due to its accuracy in resolving resonant behavior.

In CST, electromagnetic structures are defined through geometric modeling, material assignment, and boundary condition specification. The electromagnetic response is obtained by discretizing the computational domain using a finite [2] integration technique (FIT), which transforms Maxwell's equations into algebraic equations solved numerically. This approach enables accurate computation of S-parameters, field distributions, and other performance metrics relevant to microwave filter design.

Parameterization is a key capability of CST, allowing geometric dimensions such as resonator length and coupling gap to be defined as variables. By performing parameter sweeps, CST can systematically evaluate the effect of design variations on filter performance [11]. Although full-wave simulations provide high accuracy, they are computationally expensive, motivating the development of efficient surrogate modeling techniques to reduce simulation time in design optimization.

CST provides a Component Object Model (COM) interface that enables external control using scripting languages such as Python. Through this interface, simulation parameters can be programmatically modified, simulations can be executed automatically, and results can be extracted without manual intervention. This capability is essential for generating large datasets required for data-driven modeling.

In an automated CST workflow, geometric parameters are swept across predefined ranges, and corresponding S-parameter responses are computed using the frequency-domain solver [12]. Key performance metrics, such as center frequency and insertion loss, are then extracted and stored in a structured dataset. While automation significantly improves efficiency, each simulation remains a full-wave electromagnetic solve, making large-scale data generation computationally intensive.

The resulting CST-generated dataset captures the physically consistent relationship between geometry and electromagnetic response, providing a reliable foundation for training machine learning models. However, numerical noise arising from meshing, solver tolerances, and convergence behavior introduces realistic variability into the data, which must be accounted for during model training and evaluation.

Machine learning provides a data-driven approach for approximating complex input–output relationships without explicitly solving the underlying physical equations. In electromagnetic design, machine learning models are often used as surrogates to predict performance metrics from geometric parameters, significantly reducing the need for repeated full-wave simulations.

Regression-based models are particularly well-suited for surrogate modeling tasks where the objective is to predict continuous-valued quantities such as center frequency, insertion loss, or bandwidth. Linear regression models offer simplicity and interpretability but are limited in their ability to capture nonlinear relationships. In contrast, nonlinear models such as k-nearest neighbors, support vector regression, decision trees, and ensemble methods can model complex interactions present in electromagnetic systems.

Ensemble-based approaches, such as Random Forest and Gradient Boosting, combine multiple weak learners to improve prediction accuracy and robustness. [13] Gaussian Process regression provides probabilistic predictions and often achieves high accuracy for smooth functions, but its computational complexity scales poorly with dataset size. Neural networks offer flexibility but require careful tuning to avoid overfitting.

The combination of CST-generated data and machine learning enables an efficient surrogate-assisted design methodology. Full-wave CST simulations provide accurate, physics-based training data, while machine learning models learn the mapping between geometric parameters and performance metrics. Once trained, these models can rapidly predict filter behavior, enabling fast exploration of the design space.

This integrated approach does not replace full-wave simulation but complements it. Machine learning surrogates are primarily used for interpolation within the sampled design space, while CST simulations remain necessary for final validation and verification. By reducing the number of required CST runs during early-stage design, surrogate modeling significantly accelerates the overall design workflow while maintaining physical reliability.

III. Simulation and Design Methodology

A. HFSS Workflow

The bandpass filter structure is designed in HFSS using multiple PEC resonator strips. These strips are arranged in a parallel configuration and the resonator geometry is parameterized primarily through the strip lengths and the spacing between adjacent resonators. Figure 1 depicts the designed filter structure. The dimensions of the designed structure are detailed in TABLE I & TABLE II. Since during machine learning pipeline the resonator lengths vary, only the initial dimensions for the resonator strips are detailed in TABLE I.

Energy is excited and extracted through PEC cylindrical feed lines placed at the input and output of the structure. These feeds provide coupling into the resonator network which allow the incident signal to transfer into the resonant sections and propagate through the coupled resonator chain. The overall filter response is formed through inter-resonator coupling, where

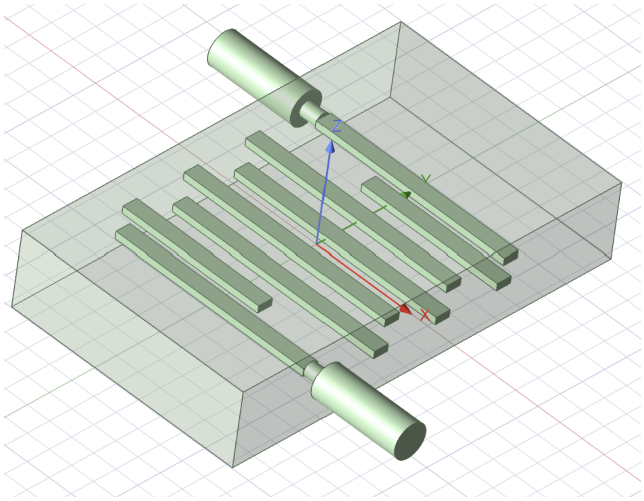

Fig. 1. BandPass Filter Design

TABLE I: Bandpass Filter Resonator Strips Initial Dimensions

Strip Name	X-size (in.)	Y-size (in.)	Z-size (in.)
L1	1.7	0.125	0.06
L2	1.818	0.125	0.06
L3	1.818	0.125	0.06
L4	1.818	0.125	0.06
L5	1.7	0.125	0.06
L6	1.818	0.125	0.06
L7	1.818	0.125	0.06
L8	1.818	0.125	0.06

TABLE II: Bandpass Filter Other Components Dimensions

Strip Name	Radius (in.)	Height (in.)
Feed 1 (Outer Coax Diameter)	0.14	0.75
Feedpin 1	0.06	0.75
Feedprobe 1	0.06	0.15

electromagnetic energy is exchanged between neighboring resonators to shape the passband and stopband characteristics. Efficient transmission occurs only for frequencies near the natural modes of the resonators and their coupled modes because of the resonant nature of the structure. Frequencies outside this region experience weak coupling and increased reflection which further result in attenuation in the stopbands.

The simulated electric-field distribution shows strong field

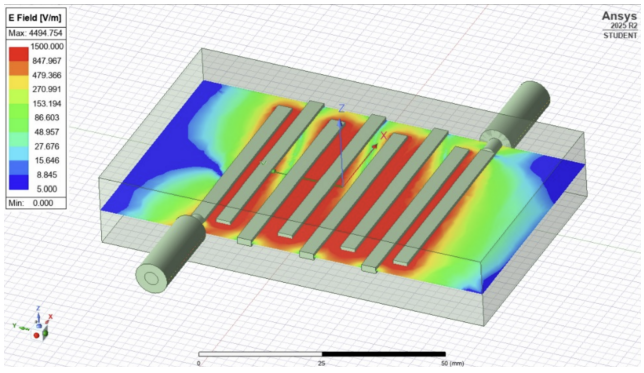

Fig. 2. Electric field magnitude visualization for the designed Bandpass filter

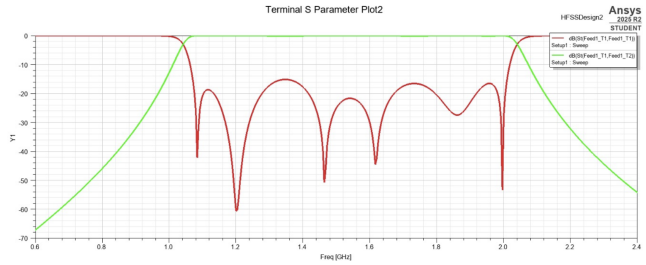

Fig. 3. Simulated S11 and S21 magnitude (dB) of the bandpass filter obtained from HFSS

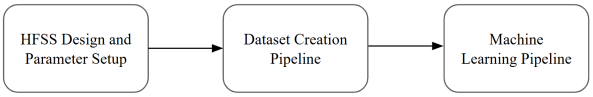

Fig. 4. High-level methodology pipeline

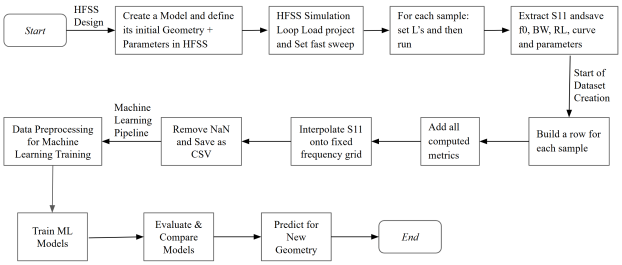

Fig. 5. HFSS-to-ML workflow for bandpass filter design

concentration within the resonator sections. The electromagnetic energy is primarily stored in these resonator sections during passband operation. Distinct field interaction between adjacent resonators shows that the inter-resonator coupling is responsible for transferring energy through the structure and shaping the overall bandpass response. In contrast, relatively low field levels are present outside the resonator network which indicate that the fields are well confined to the intended structure with minimal unwanted radiation or leakage. Figure 2 depicts this electric field distribution.

Figure 3 shows the HFSS simulation environment and the resulting terminal S-parameter response of the designed bandpass filter. A properly formed passband is indicated by low S_{11} (good impedance matching) together with high S_{21} (low insertion loss), while frequencies outside the operating band exhibit reduced S_{21} and increased reflection. Overall, the response verifies the frequency-selective behavior of the resonator-based structure.

Figure 4 depicts the high-level methodology used in this part of the project. The process begins with HFSS design and parameter setup, followed by dataset creation through automated simulations and metric extraction. The obtained dataset is then used in the machine learning pipeline for model training and evaluation of filter performance.

1) Workflow for HFSS Dataset Generation: For the HFSS workflow, the PyAnsys ecosystem was used to automate simulation and data collection. In particular, PyAEDT (the Python API for Ansys Electronics Desktop) controlled HFSS

programmatically. It allowed geometry updates, setup changes and fast sweeps to run without manually modifying the parameters in HFSS. This enabled large parameter sweeps, direct S-parameter extraction and consistent saving of results into CSV files. The automation reduced manual effort and made it practical to generate the large datasets needed for machine learning training. The joint dataset was generated by varying the four geometric parameters $(L1, L2, L3, L4)$ simultaneously and extracting bandpass performance metrics from HFSS. The workflow is summarized as follows.

1) Define design space and simulation settings. The HFSS project and design were loaded and the parameter bounds for $L1$–$L4$ were specified based on prior sweep limits. A frequency sweep range of 0.6–2.4 GHz was selected with 451 frequency points.
2) HFSS was executed in non-graphical mode (GUI off) to reduce overhead. The solution setup was tuned for faster runs by disabling field saving and limiting the adaptive passes.
3) Configure a fast frequency sweep. A fast (interpolating) sweep was created under the selected setup to efficiently evaluate the scattering parameters across the frequency band of interest.
4) Generate samples using Latin Hypercube Sampling (LHS). Latin Hypercube Sampling was used to draw N samples across the four-dimensional parameter space. This provides better coverage than purely random sampling because each parameter range is stratified while still exploring diverse joint combinations.
5) Run HFSS simulations for each sampled geometry. For each LHS point, all four parameters $(L1, L2, L3, L4)$ were updated in HFSS and the design was simulated using the configured setup and sweep. The scattering parameters S_{11} and S_{21} were extracted over the frequency grid.
6) Compute band-pass metrics and assign labels. Band-pass metrics were computed from S_{21} and S_{11} Each sample was assigned a status label (`OK` or `FAIL`) based on successful simulation.
7) Results were written to a CSV file at regular intervals to prevent data loss during long runs. Each row stored $(L1, L2, L3, L4)$, the extracted metrics $(f_0, BW_{3\text{dB}}, IL, RL_{\text{worst}}, RL_{\text{avg}})$ and the status/error message.
8) After completing all samples, the final dataset was saved and the HFSS session was closed.

2) Machine Learning Workflow: The machine learning workflow was applied to each dataset size (10, 150, 500, and 1000 samples) as well as the combined joint set (1000+500). For each case, the dataset was loaded from CSV and filtered to retain only samples with `OK` status and non-missing target values. The input features were the four geometric parameters $(L1, L2, L3, L4)$ (in inches), and the prediction target was selected as f_0 (GHz). The data were randomly split into training and testing subsets using a 90/10 split with a fixed random seed.

For machine learning training and prediction, the following regression models were implemented to estimate the target filter performance metrics from the input design parameters:

- Linear Regression
- Ridge Regression
- Lasso Regression
- ElasticNet Regression
- Bayesian Ridge Regression
- Huber Regressor
- Polynomial Regression (degree 3)
- k-Nearest Neighbors (kNN) Regressor (k=7)
- Decision Tree Regressor
- Random Forest Regressor
- Extra Trees Regressor
- Gradient Boosting Regressor
- AdaBoost Regressor
- Support Vector Regression (SVR) with RBF kernel
- Multi-Layer Perceptron (MLP) Regressor
- Gaussian Process Regressor (Constant kernel $\times$ RBF kernel)

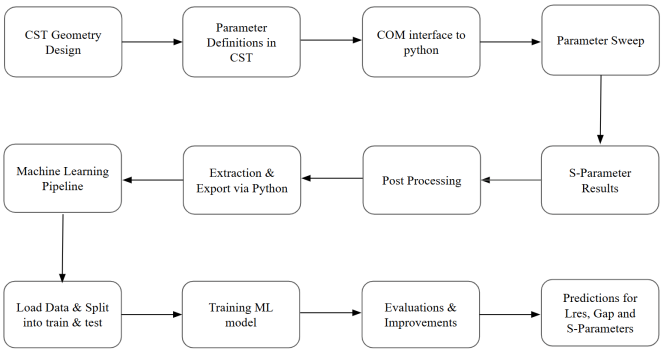

Fig. 6. CST–Python automation and ML workflow

For models sensitive to feature scaling, standardization was applied using `StandardScaler` within a `Pipeline`. After training, predictions on the held-out test set were evaluated using R^2, mean absolute error (MAE), mean squared error (MSE) and root mean squared error (RMSE). In addition to reporting metric tables and R^2 bar plots for model comparison, the best-performing model (selected by test R^2) was further analyzed using diagnostic plots, including true-versus-predicted scatter, sorted true-versus-predicted curves, residual histograms, residuals versus predicted values and absolute-error boxplots for the top five models. When the best model was tree-based, feature-importance scores were also extracted to assess the relative contribution of each geometric parameter to the predicted f_0.

B. CST Workflow

The CST portion of this project follows an automated simulation-to-learning pipeline for generating a high-volume dataset and training ML surrogates. The overall sequence is summarized in CST–Python automation and ML workflow. The band pass filter geometry is first created in CST as a parameterized model, where the key geometric variables like the resonator length, coupling gap, and related layout dimensions are defined as global parameters. A frequency-domain

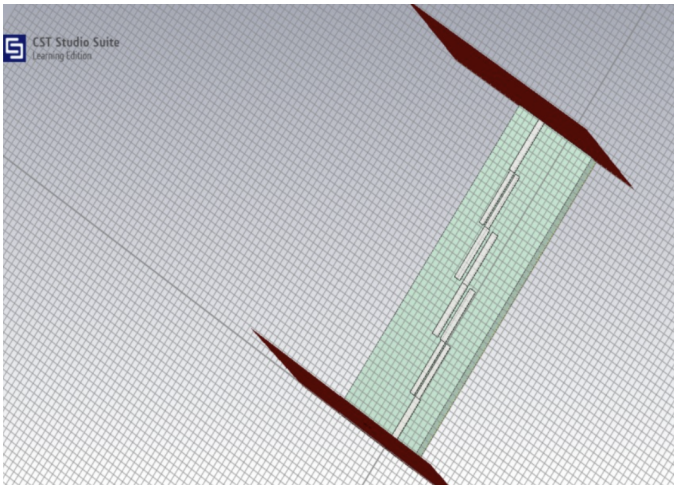

Fig. 7. Design in CST

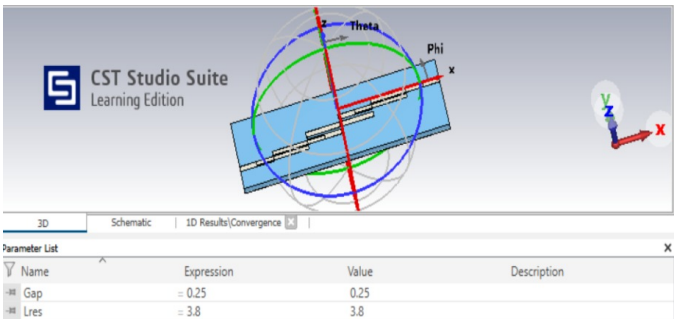

Fig. 8. Overall View in CST

solver setup is used to obtain two-port S-parameters over the design band. The simulation environment includes appropriate open boundaries and waveguide/discrete ports configured to excite the input and output feed regions consistently across the sweep.

To enable scalable dataset generation, CST is controlled from Python via the CST COM interface [1]. For each parameter sample, Python updates the CST design variables, launches the solver, and exports the resulting S-parameters. This automation loop enables the creation of hundreds to thousands of labeled EM samples without manual interaction. The exported results are stored with a consistent naming and indexing convention to preserve the mapping between each geometry configuration and its corresponding frequency response.

After each CST run, S-parameter responses are post-processed to extract compact performance metrics used as learning targets. In addition to saving the raw spectral response derived metrics are computed such as estimated center frequency , 3-dB bandwidth, insertion loss at fo, and return loss over the pass band. These metrics provide a practical representation of filter performance suitable for regression-based learning and design-space exploration.

1) Dataset Generation and Labeling in CST: To enable machine-learning-based surrogate modeling, a large labeled dataset was generated using automated full-wave simulations in CST Studio Suite. The band pass filter geometry was parameterized using a small set of physically meaningful design variables, primarily the resonator length and the inter-resonator coupling gap. These parameters directly control the center frequency, bandwidth, and impedance matching characteristics of the filter.

An automated parameter sweep was implemented using the CST–Python COM interface. For each sweep point, Python updated the CST global parameters, executed the frequency-domain solver, and exported the resulting two-port S-parameters. This process eliminated manual intervention and enabled scalable dataset generation.

In total, 1500 distinct CST simulations were performed for the final dataset. The sweep range for L res was selected based on initial hand-tuned designs to ensure that the simulated responses covered both well-matched and degraded filter behaviors. This ensured sufficient diversity in the dataset for effective ML training. Each simulation produced a frequency response over the operating band, from which performance metrics were extracted.

The resulting dataset was organized in tabular CSV format, where each row corresponds to one CST simulation and its associated labels. Prior to training, the data was randomly shuffled and split into 80% training (1200 samples) and 20% testing (300 samples) sets. Continuous input features were standardized to zero mean and unit variance to improve numerical stability during training.

To avoid learning unrealistically idealized relationships, small variations consistent with numerical EM simulation behavior like the meshing sensitivity and solver convergence variation were retained in the extracted metrics. This results in more realistic prediction errors and improves model robustness when predicting unseen geometries.

2) Machine Learning Pipeline and Model Training: Following dataset generation and labeling, a supervised machine learning pipeline was constructed to learn the relationship between CST design parameters and filter performance metrics. The objective of the ML models is to act as fast surrogate predictors that approximate the behavior of full-wave CST simulations while significantly reducing computational cost during design exploration.

The learning problem is formulated as a regression task, where the input feature vector consists of the geometric design parameters extracted from CST like the resonator length and coupling gap and the output targets correspond to filter performance metrics derived from S-parameter responses. Specifically, the models are trained to predict quantities such as insertion loss, return loss, and center frequency, which collectively characterize band pass filter performance.

Prior to training, the dataset is randomly shuffled and divided into training and testing subsets using an 80/20 split. Feature standardization is applied to all continuous input variables to ensure zero mean and unit variance, which improves numerical conditioning and convergence for gradient-based learning algorithms. The test set is held out entirely during training and used only for final performance evaluation.

To assess the suitability of different learning paradigms, multiple regression models of increasing complexity are trained and compared. These include linear models (Linear Re-

gression, Ridge, and Lasso) to establish baseline performance, distance-based models such as k-Nearest Neighbors to capture local nonlinear trends, kernel-based models such as Support Vector Regression (SVR) for smooth nonlinear mappings, and ensemble-based models including Random Forest and Gradient Boosting regressor to model higher-order interactions. This presents the significant findings derived from of the trade-offs between model complexity, prediction accuracy, and generalization.

Each model is trained independently for each target metric. Model performance is evaluated on the held-out test set using standard regression metrics including mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R^2). In addition to numerical metrics, parity plots (predicted versus true values) and residual distributions are analyzed to identify bias, variance, and systematic prediction errors across the design space. These visual diagnostics are particularly important for EM-based problems, where localized resonant behavior can lead to non-uniform error patterns.

The resulting trained models provide predictions in milliseconds, enabling rapid evaluation time-intensive date geometries that would otherwise require expensive full-wave CST simulations. This surrogate modeling capability forms the foundation for ML-assisted design optimization and design-space exploration.

3) CST-to-ML Integration and Design Feedback Loop: The integration of CST simulations with the machine learning pipeline establishes a closed-loop design framework that combines high-fidelity electromagnetic modeling with data-driven acceleration. This workflow begins with a parameterized CST geometry and proceeds through automated simulation, data extraction, surrogate modeling, and design evaluation, as summarized in the CST, Python and ML workflow diagram.

Once trained, the ML models are used to predict filter performance for previously unseen geometry configurations. Candidate designs that are predicted to meet or improve upon target specifications (lower insertion loss or improved return loss) are identified directly from the surrogate model without invoking CST. These ML-predicted geometries are then re-inserted into CST and simulated using the full wave solver to validate the accuracy of the predictions.

This validation step is critical, as it ensures that the surrogate model remains physically consistent with the underlying EM solver. Any discrepancy between ML predictions and CST-validated results can be used to further refine the dataset or re-train the model, thereby improving robustness. In practice, the surrogate-assisted approach significantly reduces the number of CST simulations required to reach acceptable performance, compared to manual tuning or solver-only optimization.

An important advantage of this framework is its modularity. While this work focuses on a planar band pass filter topology, the same CST, Python and ML integration can be extended to other microwave components by modifying the parameterized geometry and extracted performance metrics. The methodology is also compatible with both forward prediction and inverse design scenarios, where ML models propose candidate geometries based on desired specifications.

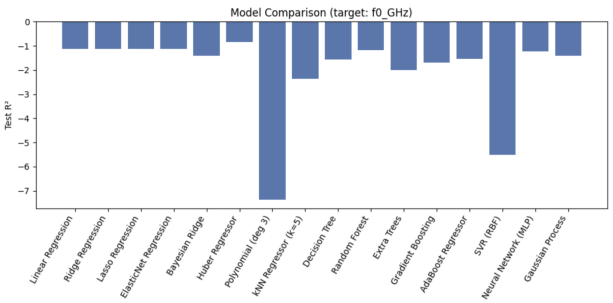

Fig. 9. R^2 results using 10 points

Overall, the CST to ML integration enables practical and scalable design methodology and was found that dataset size significantly computational cost. This hybrid approach provides a flexible foundation for accelerated microwave component design ,and optimization.

IV. Results and Analysis

The results obtained after dataset generation and ML predictions are discussed as follows.

A. HFSS Results

The dataset was generated in multiple batches of increasing size (10, 150, 500, and 1000 samples). Model performance was evaluated separately for each dataset size to compare how the accuracy and generalization changed with the increase in training data.

Figures 9–12 compare the test R^2 scores obtained for predicting the center frequency f_0 as the dataset size increases from 10 to 1000 samples.

With only 10 samples (Fig. 9), all models yield negative R^2 which depict that they generalize worse than a mean-value baseline. This behavior is expected because the training set is too small to capture the nonlinear relationship between the geometric parameters and f_0. In this regime, higher-capacity models are particularly unstable; for example, polynomial regression and the radial-basis-function support vector regressor show the largest degradation in R^2. This degradation could be dure severe overfitting or poor extrapolation. When the dataset is increased to 150 samples (Fig. 10), most models improve substantially and cluster close to $R^2 \approx 0$. This shows that the learned mappings begin to capture limited structure in the data but still provide weak predictions on unseen samples.

At 500 samples (Fig. 11), several models start to achieve positive R^2 values. The instance based and ensemble methods generally performed better than simple linear baselines. On the other hand, some single-tree and kernel-based models remained sensitive and still produced negative R^2 results.

With 1000 samples (Fig. 12), performance improves consistently across all evaluated models. The test R^2 values become positive for all the used models. Tree-ensemble regressors provide the strongest results and achieve the highest R^2 (approximately around the 0.5 range), linear models and robust linear variants remain lower (around the 0.2–0.25 range). Overall, these results show that accurate prediction of f_0 is

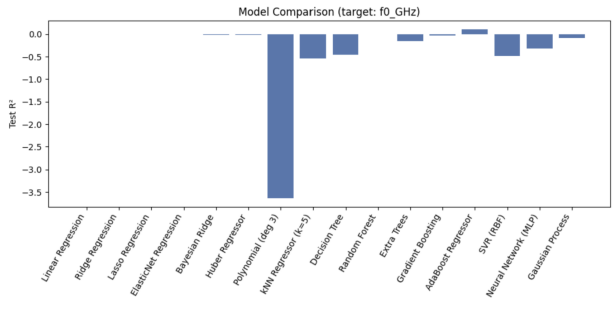

Fig. 10. R^2 results using 150 points

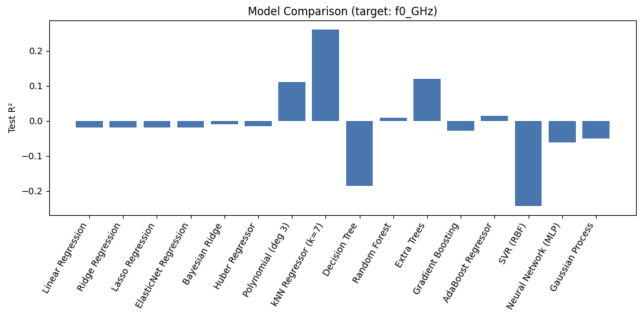

Fig. 11. R^2 results using 500 points

strongly data-dependent. We can also say that the ensemble-based models tend to benefit most from the larger training set probably due to their ability to model nonlinear interactions without requiring explicit feature engineering.

In addition to training on each dataset size individually, the 500 and 1000 sample datasets were merged to form a single joint training set in order to test whether additional diversity and volume further improve generalization. Figure 13 depicts the test R^2 scores obtained using this combined dataset. Overall, most models achieve positive R^2 values. The tree-based ensemble regressors remained the best performers with Extra Trees and Random Forest achieving the strongest scores (approximately in the 0.5 range). This trend is consistent with the 1000 sample case and supports the case that ensemble methods continue to benefit from larger and more varied training data. But the combined dataset does not improve all models. Some models show only a minor change as compared to the 1000 sample results. The single Decision Tree regressor performs noticeably worse and still gives a negative R^2 and

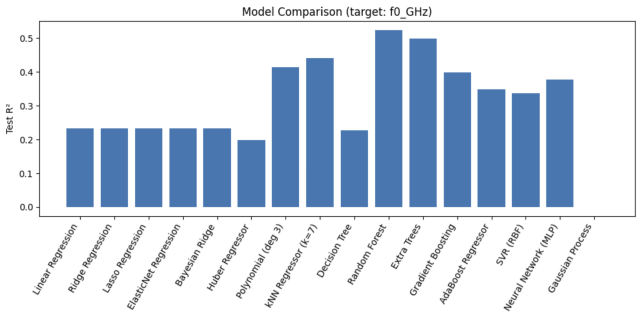

Fig. 12. R^2 results using 1000 points

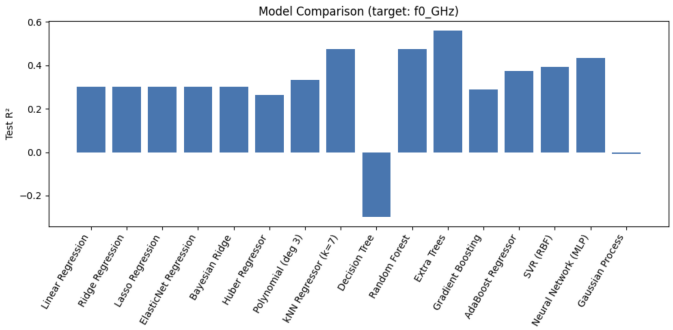

Fig. 13. R^2 results using joint set

the Gaussian Process regressor achieves an R^2 close to zero. Some models can still remain unstable even when using larger datasets.

TABLE III: Model performance metrics (150 Points)

Model	Test R^2	MAE	MSE	RMSE
Linear Regression	-0.0050	0.2290	0.0743	0.2726
Ridge Regression	-0.0050	0.2290	0.0743	0.2726
Lasso Regression	-0.0047	0.2287	0.0743	0.2726
ElasticNet Regression	-0.0048	0.2288	0.0743	0.2726
Bayesian Ridge	-0.0119	0.2283	0.0748	0.2736
Huber Regressor	-0.0116	0.2319	0.0748	0.2735
Polynomial (deg 3)	-3.6387	0.3760	0.3431	0.5857
kNN (k=5)	-0.5445	0.2643	0.1142	0.3380
Decision Tree	-0.4645	0.2665	0.1083	0.3291
Random Forest	0.0001	0.2391	0.0739	0.2719
Extra Trees	-0.1505	0.2400	0.0851	0.2917
Gradient Boosting	-0.0286	0.2367	0.0761	0.2758
AdaBoost Regressor	0.1070	0.2279	0.0660	0.2570
SVR (RBF)	-0.4903	0.2714	0.1102	0.3320
Neural Network (MLP)	-0.3191	0.2601	0.0976	0.3123
Gaussian Process	-0.0840	0.2426	0.0802	0.2831

Table III depicts the test set metrics for predicting f_0 using 150 samples. Most models achieve R^2 values close to zero or negative. The linear baselines (linear, ridge, lasso, and elastic net) perform similarly with RMSE ≈ 0.2726, suggesting that a purely linear mapping from $(L1, L2, L3, L4)$ to f_0 is insufficient. AdaBoost provides the best overall performance with the highest R^2 (0.1070) and the lowest RMSE (0.2570). Random Forest is approximately neutral in terms of R^2 (0.0001) but achieves competitive RMSE (0.2719). Polynomial regression (degree 3) performs poorly ($R^2 = -3.6387$ and RMSE $= 0.5857$) possibly due to strong overfitting or instability under limited data.

Table IV summarizes the test-set performance for predicting f_0 using 500 samples. Compared to the 150-sample case, several models show clear improvement and achieve positive R^2 values. The strongest result is obtained by the kNN regressor ($k = 7$), which achieves the highest R^2 (0.2608) and the lowest RMSE (0.2411). Polynomial regression (degree 3) and Extra Trees also perform well with positive R^2 (0.1108 and 0.1194, respectively) and reduced RMSE values (0.2644 and 0.2631). In contrast, the linear baselines remain near zero or slightly negative R^2 (around -0.018) with RMSE ≈ 0.2830, their results indicate that a linear relationship is still insufficient to model the underlying mapping. Several higher variance models, such as a single Decision Tree and SVR

TABLE IV: Model performance metrics (500 Points)

Model	Test R^2	MAE	MSE	RMSE
Linear Regression	-0.0188	0.2174	0.0801	0.2830
Ridge Regression	-0.0188	0.2174	0.0801	0.2830
Lasso Regression	-0.0184	0.2174	0.0801	0.2830
ElasticNet Regression	-0.0185	0.2174	0.0801	0.2830
Bayesian Ridge	-0.0103	0.2180	0.0794	0.2818
Huber Regressor	-0.0157	0.2135	0.0799	0.2826
Polynomial (deg 3)	0.1108	0.1952	0.0699	0.2644
kNN Regressor (k=7)	0.2608	0.1576	0.0581	0.2411
Decision Tree	-0.1862	0.2183	0.0933	0.3054
Random Forest	0.0087	0.1892	0.0779	0.2792
Extra Trees	0.1194	0.1804	0.0692	0.2631
Gradient Boosting	-0.0286	0.2094	0.0809	0.2844
AdaBoost Regressor	0.0147	0.2211	0.0775	0.2783
SVR (RBF)	-0.2431	0.1973	0.0977	0.3126
Neural Network (MLP)	-0.0610	0.1927	0.0834	0.2888
Gaussian Process	-0.0507	0.2335	0.0826	0.2874

(RBF), perform poorly with negative R^2 and larger errors. Overall, the results indicate that increasing the dataset to 500 samples begins to reveal the benefit of nonlinear models, especially instance-based and ensemble approaches.

TABLE V: Model performance metrics (1000 Points)

Model	Test R^2	MAE	MSE	RMSE
Linear Regression	0.2335	0.2440	0.0853	0.2920
Ridge Regression	0.2335	0.2440	0.0853	0.2920
Lasso Regression	0.2335	0.2441	0.0853	0.2920
ElasticNet Regression	0.2335	0.2441	0.0853	0.2920
Bayesian Ridge	0.2327	0.2446	0.0854	0.2922
Huber Regressor	0.1980	0.2317	0.0892	0.2987
Polynomial (deg 3)	0.4136	0.1924	0.0652	0.2554
kNN Regressor (k=7)	0.4405	0.1782	0.0623	0.2495
Decision Tree	0.2277	0.1996	0.0859	0.2931
Random Forest	0.5237	0.1658	0.0530	0.2302
Extra Trees	0.4977	0.1689	0.0559	0.2364
Gradient Boosting	0.3989	0.1944	0.0669	0.2586
AdaBoost Regressor	0.3478	0.2326	0.0726	0.2694
SVR (RBF)	0.3366	0.2030	0.0738	0.2717
Neural Network (MLP)	0.3770	0.1972	0.0693	0.2633
Gaussian Process	-0.0005	0.2863	0.1113	0.3336

Table V depicts the test performance for predicting f_0 using 1000 samples. With this larger dataset, most models achieve clearly positive R^2 values which is a substantial improvement in generalization compared to the smaller training sets. The best overall performance is obtained by Random Forest ($R^2 = 0.5237$) with the lowest RMSE (0.2302), followed closely by Extra Trees ($R^2 = 0.4977$, RMSE $= 0.2364$). These results indicate that tree-ensemble methods capture the nonlinear dependence of f_0 on the geometric parameters more effectively than simpler baselines. Several other nonlinear models also perform well, including kNN ($R^2 = 0.4405$, RMSE $= 0.2495$) and polynomial regression (degree 3) ($R^2 = 0.4136$, RMSE $= 0.2554$). In comparison, the linear regression family (linear, ridge, lasso, elastic net, and Bayesian ridge) shows moderate performance with $R^2 \approx 0.23$ and RMSE ≈ 0.292, suggesting that linear trends are present but do not fully explain the data. Notably, the Gaussian Process regressor performs poorly ($R^2 \approx 0$) with the largest error metrics, indicating limited effectiveness under the current kernel and scaling settings for this dataset size.

Figure 14 shows the true-versus-predicted scatter for the Random Forest model trained on the 1000-sample dataset.

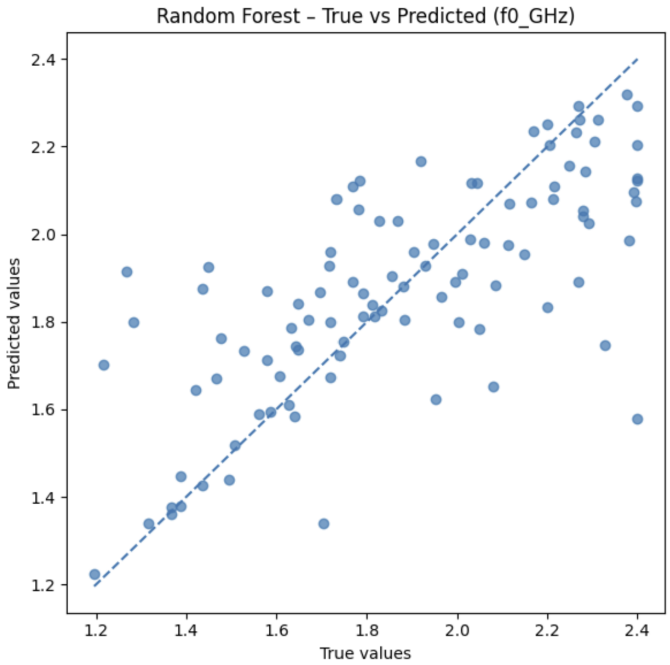

Fig. 14. Random Forest true vs predicted f_0 on the 1000 sample dataset.

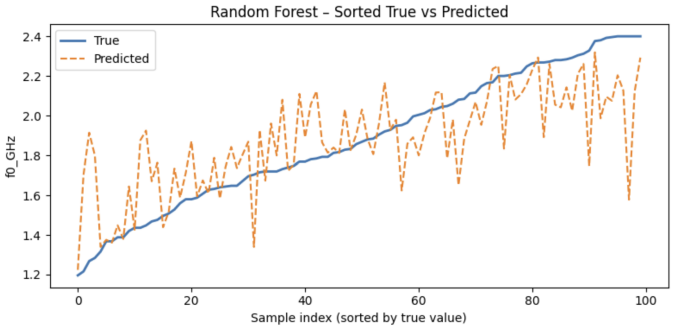

Fig. 15. Sorted true and predicted f_0 for the Random Forest model on the 1000-sample dataset (samples ordered by true f_0).

Most points cluster around the $y = x$ reference line. This shows that the model predicts f_0 reasonably well across the full range. The spread around the diagonal increases slightly at higher f_0 values, which suggests larger errors for some designs in that region.

Figure 15 presents the sorted true and predicted f_0 values (ordered by the true target). The predicted curve follows the overall increasing trend of the ground-truth values, confirming that the model captures the monotonic relationship between the design parameters and f_0. However, the visible oscillations and occasional deviations from the true curve indicate localized prediction errors, consistent with the scatter observed in Fig. 14.

Figure 16 shows the residual histogram (true minus predicted f_0) for the Random Forest model. The histogram indicates that the model does not exhibit a strong overall bias toward overprediction or underprediction. Most residuals fall within a relatively narrow range, but a few larger residuals appear in the tails corresponding to occasional higher-error test cases.

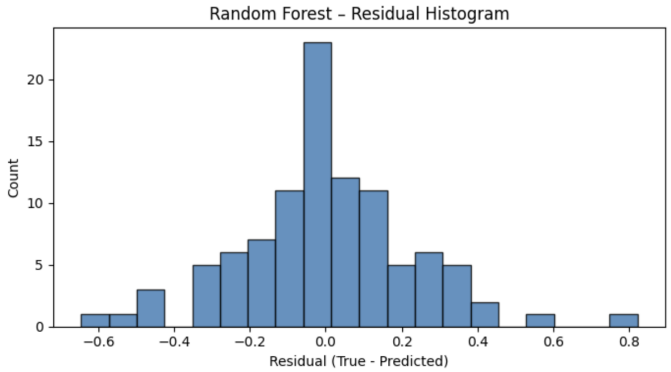

Fig. 16. Residual histogram for Random Forest f_0 predictions on the 1000 sample dataset.

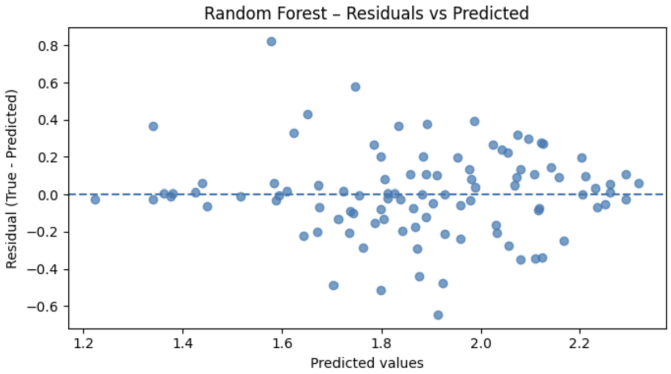

Fig. 17. Residuals versus predicted f_0 for Random Forest on the 1000 sample dataset.

Figure 17 plots residuals versus predicted f_0. The residuals are scattered around the zero line without a clear trend. A small number of outliers remain visible, which is consistent with the histogram tails in Fig. 16.

Figure 18 compares the absolute error distributions for the top-performing models on the 1000-sample dataset. The boxplots indicate that the ensemble-based models generally achieve lower median errors and tighter interquartile ranges than the weaker baselines, while all methods still show some larger-error outliers.

Finally, Fig. 19 depicts the Random Forest feature importances. Among the four geometric inputs, $L3$ contributes the most to the prediction of f_0, followed by $L4$, while $L2$ and $L1$

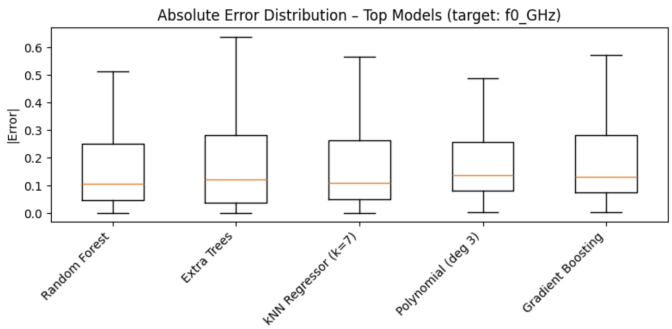

Fig. 18. Absolute error distributions for the top models on the 1000 sample dataset (target: f_0).

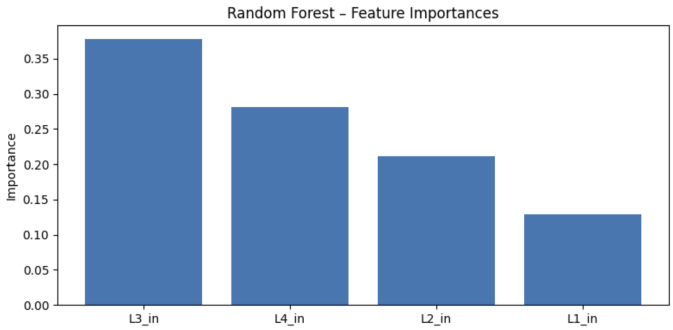

Fig. 19. Random Forest feature importances for predicting f_0 using 1000 samples.

have lower relative importance. This suggests that variations in $L3$ and $L4$ have the strongest influence on the center-frequency prediction within the explored design space.

TABLE VI: Model performance metrics (Joint Set 1000 + 500)

Model	Test R^2	MAE	MSE	RMSE
Linear Regression	0.3028	0.2407	0.0822	0.2868
Ridge Regression	0.3028	0.2407	0.0822	0.2868
Lasso Regression	0.3026	0.2408	0.0823	0.2868
ElasticNet Regression	0.3026	0.2407	0.0823	0.2868
Bayesian Ridge	0.3007	0.2413	0.0825	0.2872
Huber Regressor	0.2634	0.2329	0.0869	0.2948
Polynomial (deg 3)	0.3342	0.2117	0.0785	0.2803
kNN Regressor (k=7)	0.4739	0.1855	0.0621	0.2491
Decision Tree	-0.2987	0.2760	0.1532	0.3914
Random Forest	0.4762	0.1891	0.0618	0.2486
Extra Trees	0.5602	0.1732	0.0519	0.2278
Gradient Boosting	0.2897	0.2293	0.0838	0.2895
AdaBoost Regressor	0.3736	0.2296	0.0739	0.2718
SVR (RBF)	0.3927	0.2027	0.0716	0.2677
Neural Network (MLP)	0.4333	0.1899	0.0668	0.2585
Gaussian Process	-0.0087	0.2908	0.1190	0.3450

Table VI presents the model performance for the joint dataset formed by combining the 500 and 1000 sample sets. Overall, most models improve compared to the 500 sample case and several achieve higher R^2 than the 1000 sample results. Extra Trees provides the best performance ($R^2 = 0.5602$) and the lowest RMSE (0.2278), followed by Random Forest ($R^2 = 0.4762$, RMSE $= 0.2486$) and kNN ($R^2 = 0.4739$, RMSE $= 0.2491$). The neural network also performs competitively ($R^2 = 0.4333$, RMSE $= 0.2585$). These results further prove that larger and more diverse training data can benefit higher-capacity models. The linear regression family shows moderate performance improvement ($R^2 \approx 0.30$, RMSE ≈ 0.2868). In contrast, the single Decision Tree performs poorly ($R^2 = -0.2987$), and the Gaussian Process regressor remains near zero/negative R^2 indicating that these models are less robust under the current settings even when more data are available.

Figure 20 shows the sorted true vs predicted comparison for f_0 using the best-performing model (Extra Trees) trained on the joint dataset (500+1000 samples). The true f_0 values increase smoothly because the samples are ordered by the ground-truth target, while the predicted curve follows the overall upward trend across the full range. Most predictions remain close to the true values establishing that the model captures the general relationship between $(L1, L2, L3, L4)$

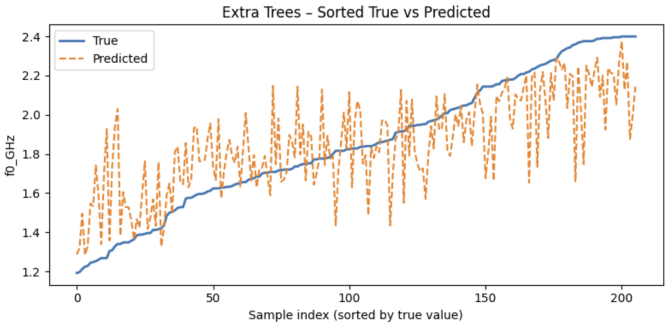

Fig. 20. Sorted true versus predicted f_0 for the Extra Trees regressor trained using the joint dataset.

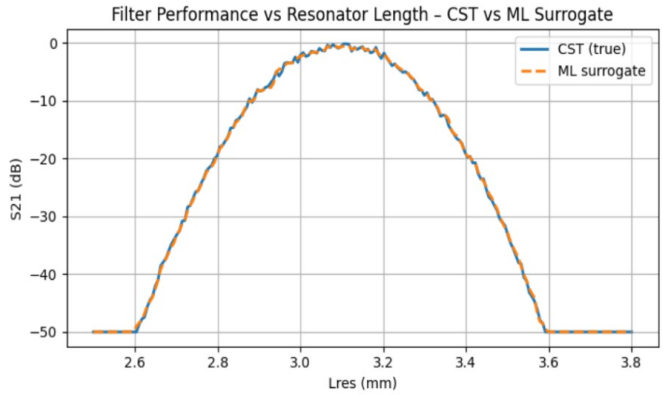

Fig. 21. CST vs ML Surrogate Filter Response

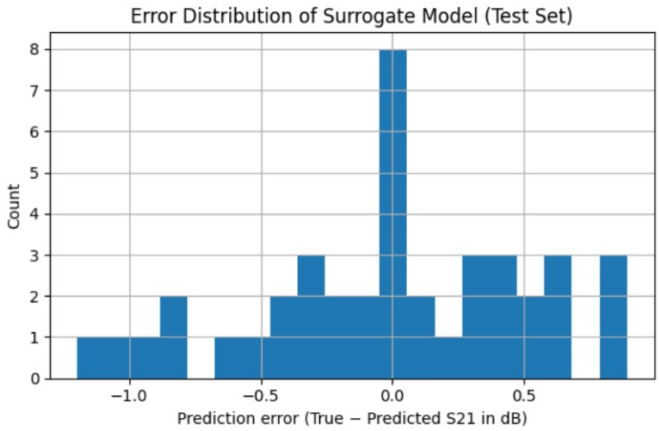

Fig. 22. Error Distribution of ML Surrogate Model

Fig. 23. Manual Design vs ML Model Error Comparison

and f_0. The larger deviations at some indices suggest localized prediction errors for specific design regions, but the overall alignment supports the strong R^2 and RMSE reported for the joint-set experiment.

B. CST Results

The CST parameter sweep confirms the expected electromagnetic behavior of the coupled microstrip filter. As the resonator length increases, the filter response exhibits a smooth and physically consistent variation in insertion loss and center frequency. These trends form a well-structured mapping between geometry and performance metrics, which is essential for training reliable surrogate models.

Fig. 21 (CST vs ML Surrogate Filter Response) compares the CST-simulated and ML-predicted insertion loss as a function of resonator length. The close overlap between the two curves demonstrates that the trained surrogate accurately captures the underlying CST response across the design space. Minor deviations near the sweep boundaries are attributed to reduced sampling density and increased sensitivity of the EM response at extreme geometries.

To assess the reliability of the predictions, the surrogate model's error characteristics are analyzed in detail. Fig. 22 (Error Distribution of ML Surrogate Model) shows the histogram of prediction errors on the test set. The distribution is centered near zero with a narrow spread, indicating unbiased predictions and low variance. Further insight is provided by Fig. IV-4 (Residuals vs Resonator Length), which plots residuals as a function of Lres. The absence of a systematic trend confirms that the surrogate does not exhibit geometry-dependent bias and remains robust across the swept range. Together, these results validate the stability and generalization capability of the ML surrogate.

The practical benefit of the surrogate model is highlighted by comparison with a manual design baseline. Fig. 23 (Manual Design vs ML Model Error Comparison) contrasts the prediction error of a manual initial guess with that of the Random Forest surrogate. The ML model achieves substantial reductions in both MAE and RMSE, demonstrating its effectiveness in accelerating the design process by providing accurate performance estimates before full-wave CST validation.

A comprehensive comparison of machine learning models is presented using the coefficient of determination (R-squared) and error-based metrics. Fig. 24 and Fig. 25 show the test-set R-squared comparison for 500 and 1000 samples, respectively, when predicting the center frequency f0. Most models achieve very high R-squared values, reflecting a smooth, physics-constrained relationship between geometry and f0.

However, error-based metrics reveal finer distinctions among models. Fig. 26 (RMSE Comparison Across ML Models) demonstrates that nonlinear and ensemble methods

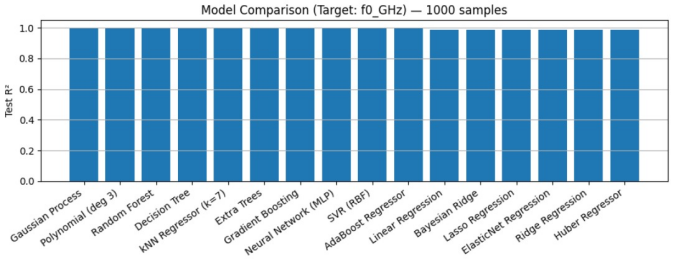

Fig. 24. R-squared 1000 Samples ML Models

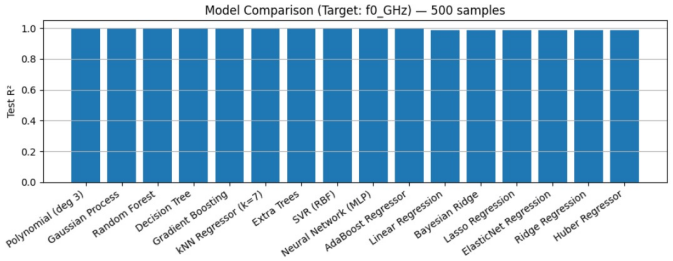

Fig. 25. R-squared 500 Samples ML Models

such as Random Forest, KNN, Gradient Boosting, and SVR significantly outperform linear models in predicting insertion loss. Linear, Ridge, and Lasso regressors exhibit much higher RMSE, indicating their inability to capture nonlinear EM behavior.

The Gaussian Process model provides near-perfect tracking of the center frequency, as shown in Fig. 27 (Gaussian Process Prediction Accuracy – Sorted Values). [2] While its residuals (Fig. 28 and Fig. 29) exhibit slight structured behavior, the overall error magnitude remains extremely small, confirming high predictive accuracy.

Random Forest and KNN models offer a favorable balance between accuracy, robustness, and computational efficiency. Unlike Gaussian Processes, which scale poorly with dataset size, ensemble models remain tractable for large CST-generated datasets while maintaining low prediction error.

A consolidated quantitative comparison is provided in Table IV-1 (ML Model Performance Across Dataset Sizes). The table reports MAE, MSE, RMSE, and test-set R-squared for all evaluated models at dataset sizes of 500, 1000, and 1500 samples. While R-squared values remain high across many

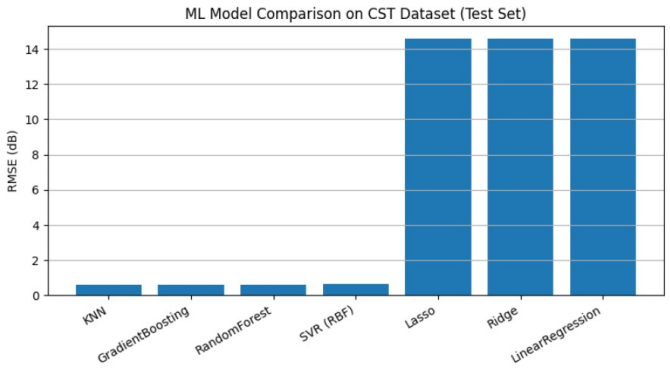

Fig. 26. RMSE Comparison Across ML Models

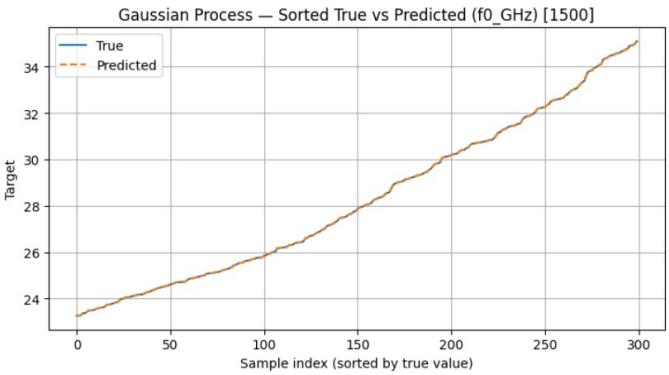

Fig. 27. Gaussian Process Prediction Accuracy – Sorted Values

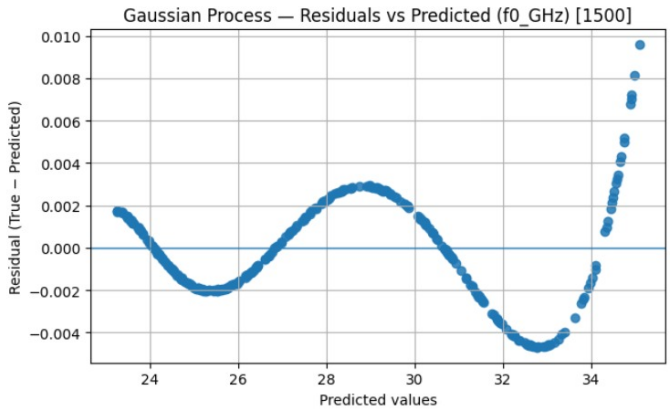

Fig. 28. Gaussian Process - Residual vs Predicted

models due to the deterministic nature of the physics, RMSE and MAE clearly differentiate model quality.

Across all dataset sizes, Gaussian Process, Polynomial Regression, Random Forest, and KNN achieve the lowest errors, whereas linear and regularized linear models consistently underperform. Increasing dataset size leads to systematic reductions in error for nonlinear and ensemble models, confirming the benefit of larger CST-generated training sets [13].

Overall, the results demonstrate that machine learning surrogates trained on CST-generated data can accurately and

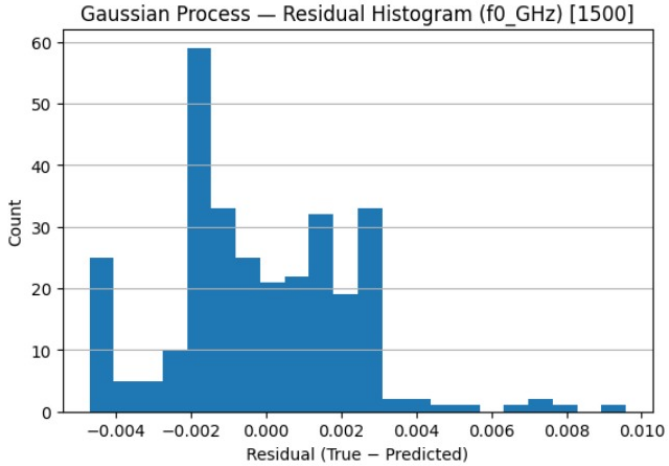

Fig. 29. Gaussian Process - Residual Histogram

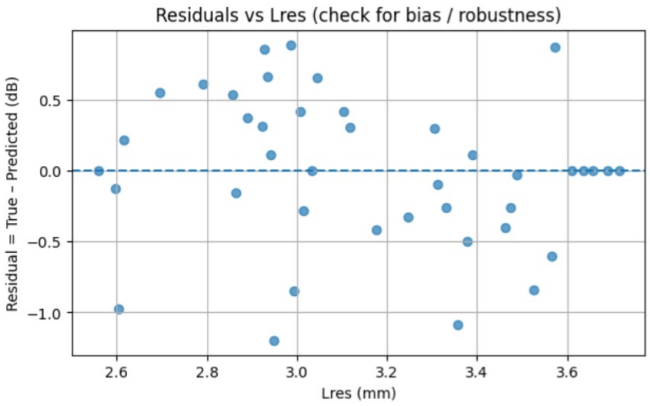

Fig. 30. Residual vs Lres

robustly predict filter performance metrics. While center-frequency prediction is straightforward due to its smooth dependence on geometry, insertion-loss prediction is a more challenging nonlinear problem that clearly benefits from advanced ML models. The Random Forest surrogate, in particular, provides a practical compromise among accuracy, interpretability, and scalability, making it well-suited for ML-assisted CST-based filter design and rapid design-space exploration.

Table VII summarizes the predictive performance of all evaluated machine learning models trained on CST-generated datasets of increasing size (500, 1000, and 1500 samples). For each model, the table reports the test-set coefficient of determination R-squared along with error-based metrics including mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE).

While most models achieve very high R squared values—reflecting the smooth and physics-constrained relationship between geometry and filter performance—the error metrics provide more apparent discrimination among models. Nonlinear and ensemble-based approaches, such as Gaussian Process, Random Forest, polynomial regression, and KNN, consistently achieve the lowest MAE and RMSE across all dataset sizes. In contrast, linear and regularized linear models (Linear, Ridge, Lasso, Elastic-Net) exhibit substantially higher errors, indicating limited capacity to capture nonlinear electromagnetic effects. The progressive reduction in error with increasing dataset size further highlights the benefit of larger CST-generated training sets for surrogate model accuracy.

V. Discussion

This section outlines the major findings based on the obtained results. It also discusses the challenges encountered, key limitations and potential directions for future improvements.

A. Major Findings

1) HFSS and Machine Learning:

- Dataset size is critical for generalization. With 10 samples, all models produced negative test R^2. Performance improved steadily as the dataset increased (150 $\rightarrow$ 500 $\rightarrow$ 1000 samples), with most models reaching positive R^2 only at the larger sizes.
- Tree-ensemble methods provide the strongest and most consistent performance. For 1000 samples, Random Forest achieved the best results ($R^2 = 0.5237$ with the lowest RMSE), and for the joint dataset (500+1000), Extra Trees performed best ($R^2 = 0.5602$ with the lowest RMSE).
- The joint dataset increased performance for most models and further strengthened the top ensemble learners. However, the improvements were not uniform across all model classes.
- The interpretability of the model indicates key geometric drivers. Feature-importance analysis (Random Forest, 1000 samples) shows $L3$ and $L4$ are the most influential inputs for predicting f_0 in the explored design space and $L2$, $L1$ have comparatively smaller impact.

2) CST and Machine Learning:

- The integration of CST Studio Suite with machine learning enabled an efficient surrogate-based design workflow for the microwave band-pass filter. Automated parameter sweeps in CST generated a large, physically consistent dataset that links geometric parameters, such as resonator length and coupling gap, to key filter performance metrics, including center frequency and insertion loss.
- Nonlinear and ensemble-based machine learning models demonstrated superior predictive performance compared to linear baselines. Models such as Random Forest, KNN, and Gaussian Process regression achieved very low prediction error across all dataset sizes, indicating their ability to capture the underlying electromagnetic behavior governed by resonant and coupling effects.
- Increasing the CST-generated dataset size from 500 to 1500 samples resulted in consistent reductions in error-based metrics such as MAE and RMSE, even though test R^2 values remained high for most models. This highlights the benefit of large CST datasets for improving surrogate accuracy and robustness, particularly for nonlinear performance metrics [13].

The HFSS surrogate models appear to be limited by the data since the best test R^2 reaches only about 0.5–0.56 which indicates that more samples and better design-space coverage are needed for stronger generalization. In contrast, the CST models reach much higher R^2 values (up to ~ 0.9), but this likely reflects overfitting and suggests the need for further possible regularization.

B. Challenges Faced

1) HFSS and Machine Learning:

- Computational resources: This work requires significant computational resources, especially during simulation and model training. For this project, a personal computer with NVIDIA 4060 GPU and intel core i9 processor was used.
- Time intensive: The overall pipeline is very time intensive because running simulations and generating enough training data can take many hours or days. Dataset generation for 1000 points took multiple to complete.
- Dataset size: The dataset size is the foremost limiting factor for this project. We started from 30 rows and generated upto 1000 rows for our dataset. From the

TABLE VII: Performance comparison of machine learning models trained on CST-generated data for different dataset sizes.

N	Model	Test R^2	MAE	MSE	RMSE
15*500	Polynomial (deg 3)	0.999998	0.00368	2.01×10^{-5}	0.00449
	Gaussian Process	0.999996	0.00521	4.36×10^{-5}	0.00660
	Random Forest	0.999956	0.01583	5.24×10^{-4}	0.02289
	Decision Tree	0.999954	0.01861	5.51×10^{-4}	0.02347
	Gradient Boosting	0.999918	0.02550	9.79×10^{-4}	0.03129
	kNN (k=7)	0.999894	0.02520	1.27×10^{-3}	0.03569
	Extra Trees	0.999658	0.04580	4.11×10^{-3}	0.06409
	SVR (RBF)	0.999290	0.08497	8.52×10^{-3}	0.09228
	Neural Network (MLP)	0.999086	0.08416	1.10×10^{-2}	0.10473
	AdaBoost	0.995768	0.17961	5.08×10^{-2}	0.22531
	Linear Regression	0.988482	0.31203	0.13817	0.37172
	Bayesian Ridge	0.988481	0.31204	0.13818	0.37173
	Lasso Regression	0.988473	0.31215	0.13828	0.37186
	ElasticNet	0.988463	0.31230	0.13840	0.37202
	Huber Regressor	0.987442	0.31307	0.15065	0.38813
14*1000	Gaussian Process	0.999999	0.00291	1.28×10^{-5}	0.00358
	Polynomial (deg 3)	0.999999	0.00351	1.66×10^{-5}	0.00408
	Random Forest	0.999994	0.00682	7.70×10^{-5}	0.00877
	Decision Tree	0.999991	0.00945	1.09×10^{-4}	0.01045
	kNN (k=7)	0.999989	0.00861	1.40×10^{-4}	0.01182
	Extra Trees	0.999943	0.02274	6.90×10^{-4}	0.02626
	Gradient Boosting	0.999934	0.02265	8.04×10^{-4}	0.02836
	Neural Network (MLP)	0.999720	0.04813	3.40×10^{-3}	0.05831
	SVR (RBF)	0.999342	0.08105	7.99×10^{-3}	0.08938
	AdaBoost	0.996882	0.16081	3.79×10^{-2}	0.19458
	Linear Regression	0.989193	0.30394	0.13124	0.36227
	Bayesian Ridge	0.989193	0.30393	0.13124	0.36227
	ElasticNet	0.989193	0.30365	0.13125	0.36228
	Huber Regressor	0.988615	0.29988	0.13826	0.37184
14*1500	Gaussian Process	0.999999	0.00196	5.86×10^{-6}	0.00242
	Polynomial (deg 3)	0.999999	0.00347	1.57×10^{-5}	0.00397
	kNN (k=7)	0.999998	0.00420	2.85×10^{-5}	0.00534
	Random Forest	0.999997	0.00465	3.06×10^{-5}	0.00554
	Decision Tree	0.999994	0.00821	7.25×10^{-5}	0.00851
	Extra Trees	0.999976	0.01412	2.85×10^{-4}	0.01687
	Gradient Boosting	0.999932	0.02219	8.04×10^{-4}	0.02836
	Neural Network (MLP)	0.999748	0.04414	2.99×10^{-3}	0.05469
	SVR (RBF)	0.999317	0.08265	8.11×10^{-3}	0.09008
	AdaBoost	0.993518	0.24215	7.70×10^{-2}	0.27749
	Ridge Regression	0.989392	0.29953	0.12600	0.35497
	ElasticNet	0.989391	0.29960	0.12602	0.35499
	Lasso Regression	0.989388	0.29975	0.12605	0.35503
	Linear Regression	0.989386	0.29986	0.12607	0.35507

related works as well, it was discovered that dataset size significantly impacts the performance.

- Lock System: Throughout the dataset generation process multiple crashes were encountered. This caused a problem that since the project was not closed properly in Python (PyAEDT). Accessing the same project in the future would prompt messages like: Unable to open this file/Opening this file can damage the contents. This was an issue caused by the lock system of PyAEDT. If the project was not closed properly the lock file would prevent from accessing that project in the future until that file was deleted manually.
- Corrupted Files: The issues caused by the lock system persisted couple of times even after deletion. The contents of the main design file were corrupted and lost after the deletion of the lock file twice.
- System Crashes: Multiple crashes interrupted the dataset generation where the process would have to be restarted and the progress made upto that point would be lost.

2) CST and Machine Learning:

- A primary challenge was the computational cost of large-scale CST simulations. Although Python-based automation reduced manual effort, full-wave frequency-domain simulations remained time-intensive, thereby limiting the generation of large datasets in the overall workflow.
- Automation reliability posed additional challenges during CST–Python integration. Solver interruptions, convergence failures, and licensing-related issues occasionally led to incomplete or corrupted simulation outputs, necessitating careful monitoring and post-processing to ensure dataset integrity.
- From a machine learning perspective, model scalability and generalization were key challenges. Highly accurate models, such as Gaussian Process regression, exhibited poor computational scaling with increasing dataset size, whereas simpler models struggled to maintain accuracy on nonlinear performance metrics. Furthermore, ML predictions outside the parameter sweep bounds risked producing non-physical results, necessitating strict design-space constraints and CST-based validation.

C. Limitations

Despite producing a functional design and useful ML predictions, our methodology and workflow has several practical limitations. These limitations are discussed as follows:

- The developed CST/HFSS + ML workflow is tailored to a very specific band-pass filter topology. Therefore, it may not directly generalize to other filter without retraining and redesigning the feature set and dataset.
- Changes in the software/simulator version or solver configuration for either HFSS or CST can cause small variations in results, which makes it harder to maintain consistent and reproducible dataset generation. The python programs would also need to be altered depending upon the versions of HFSS/CST being used.
- Generating high quality simulation data is very time-consuming and there are very few publicly available datasets for RF band-pass filter structures. As a result, it is difficult to find datasets that match our specific design topology and parameter ranges.
- Since we are working with a limited dataset, it is possible that complex ML models can overfit and look accurate on training data but fail on newer designs.
- Some Machine Learning models provide good predictions but limited insight. This makes it difficult to explain why a design change shifts frequency or bandwidth.
- Small mistakes in wave ports, reference planes or radiation boundaries can significantly impact S-parameters. This can mislead the training dataset and hence produce inaccurate results.
- A model trained on HFSS data may not match CST outputs possibly due to solver and meshing differences. Therefore, data reuse between HFSS and CST is limited.
- Python scripts that controls HFSS/CST runs can break due to GUI changes, license availability, file locks or solver crashes.

D. Future Work

Future works can extend this approach by improving the HFSS/CST–machine learning pipeline for faster data generation, stronger generalization across designs and more robust optimization. Multi-objective optimization frameworks like Pareto-front methods can be used to explicitly trade off between bandwidth, insertion loss, return loss, size etc instead of optimizing them one target at a time. Variations in ε_r, $\tan\delta$, copper roughness, and fabrication tolerances can also be introduced to more closely resemble practical scenarios. This entire process can also be inverted by exploring conditional generative approaches that propose geometries directly from desired specifications then the results can be validated with HFSS/CST.

VI. Conclusion

In conclusion, by combining HFSS and CST simulations with machine learning, this project demonstrates a more efficient methodology for bandpass filter design. The full-wave simulations provided high-accuracy performance data and the ML pipeline accelerated the process by predicting filter responses. The HFSS results show requirement of more training data and the CST results show overfitting of the R^2 values. And although factors like time/computational costs and model generalization remain hurdles, this hybrid approach effectively minimizes trial-and-errors and offers a faster path to high performance hardware development.

VII. Contribution

Prabhroop Singh was responsible for the HFSS components of the project while Kalyana Abenanth Gurunathan led the CST components. The remaining sections were developed collaboratively by both authors. The generated datasets, scripts and supporting files are available on Google Drive at: *Repository Link*.

Acknowledgment

We would like to thank Dr. Rashid Mirzavand for his guidance, feedback and continuous support throughout this project that improved the quality of the design and analysis.

References

[1] Dassault Systèmes. (2024) Cst studio suite — electromagnetic field simulation software. Dassault Systèmes. [Online]. Available: https://www.3ds.com/products-services/simulia/products/cst-studio-suite/

[2] Q.-J. Zhang, F. Feng, T. Zhang, and Q.-J. Zhang, "Machine learning for microwave engineering: A review," *IEEE Transactions on Microwave Theory and Techniques*, vol. 68, no. 11, pp. 4815–4834, Nov. 2020.

[3] Wikipedia contributors. (2025) Band-pass filter. Wikipedia. Last edited 24 September 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Band-pass_filter&oldid=1313212130

[4] T. R. Kuphaldt. Band-pass filters. All About Circuits. [Online]. Available: https://www.allaboutcircuits.com/textbook/alternating-current/chpt-8/band-pass-filters/

[5] F. Ke, Q. Wu, and H. Wang, "Automatic synthesis of parallel-coupled bandpass filter using machine-learning-assisted optimization with prior knowledge integration," in *2025 International Conference on Microwave and Millimeter Wave Technology (ICMMT)*. IEEE, May 2025, pp. 1–3, accessed: Dec. 17, 2025. [Online]. Available: https://doi.org/10.1109/ICMMT65948.2025.11188561

[6] J.-T. Kuo, T.-H. Yeh, and C.-C. Yeh, "Design of microstrip bandpass filters with a dual-passband response," *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, no. 4, pp. 1331–1337, Apr. 2005.

[7] M. V. Suraj, R. Padmasree, and M. Ankitha, "Exploring innovative methods for dielectric resonator antenna design with hfss and machine learning integration," *International Journal of Computer Applications*, vol. 186, no. 24, pp. 17–22, Jun. 2024, iSSN: 0975–8887.

[8] Y. Sharma, H. H. Zhang, and H. Xin, "Machine learning techniques for optimizing design of double t-shaped monopole antenna," *IEEE Transactions on Antennas and Propagation*, vol. 68, no. 7, pp. 5658–5663, Jul. 2020.

[9] A. Kumar and D. K. Jhariya, "Design and optimization of a bandpass filter using machine learning," *Physica Scripta*, vol. 100, no. 7, p. 075574, jul 1 2025, [Online; accessed 2025-12-17].

[10] S. Javadi, B. Rezaee, S. S. Nabavi, M. E. Gadringer, and W. Bösch, "Machine Learning-Driven Approaches for Advanced Microwave Filter Design," *Electronics*, vol. 14, no. 2, p. 367, jan 17 2025, [Online; accessed 2025-12-17].

[11] F. Feng, J. Zhang, W. Na, and Q.-J. Zhang, "Adaptive feature zeroing deep neural networks for microwave device modeling," *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, no. 10, pp. 4166–4180, Oct. 2019.

[12] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[13] J. E. Rayas-Sánchez and V. Gutierrez-Ayala, "Em-based surrogate modeling and optimization," *IEEE Microwave Magazine*, vol. 18, no. 6, pp. 41–51, Sep. 2017.

[14] J. Uher and W. Hoefer, "Tunable microwave and millimeter-wave band-pass filters," *IEEE Transactions on Microwave Theory and Techniques*, vol. 39, no. 4, pp. 643–653, 4 1991, [Online; accessed 2025-12-17].

[15] J. Kumar Rai, P. Ranjan, and R. Chowdhury, "Machine learning enabled al_2o_3 ceramic based dual band frequency reconfigurable dielectric antenna for wireless application," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 31, no. 5, pp. 2840–2849, Oct. 2024.

[16] H. M. El Misilmani, T. Naous, and S. K. Al Khatib, "A review on the design and optimization of antennas using machine learning algorithms and techniques," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 30, no. 10, jul 27 2020, [Online; accessed 2025-12-17].

[17] H. M. E. Misilmani and T. Naous, "Machine Learning in Antenna Design: An Overview on Machine Learning Concept and Algorithms," in *2019 International Conference on High Performance Computing amp; Simulation (HPCS)*. IEEE, 7 2019, pp. 600–607, [Online; accessed 2025-12-17].

[18] J.-S. Hong and M. J. Lancaster, *Microstrip Filters for RF/Microwave Applications*, 2nd ed. Hoboken, NJ, USA: Wiley, 2011.