# HIGH-SPEED SIGNAL INTEGRITY ANALYSIS AND CHANNEL MODELING USING NEURAL NETWORKS

BY

JUHITHA KONDURU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2025

Urbana, Illinois

Doctoral Committee:

Professor JOSÉ SCHUTT-AINÉ, Chair
Professor Jennifer Bernhard
Professor Zhen Peng
Professor Pavan Kumar Hanumolu

## ABSTRACT

The analysis of high-speed networks is often carried out using transistor level simulation tools which have large computational time. This leads to a limitation in terms of the amount of time spent generating an optimal design and accurately analyzing the system. Therefore, there is a need for fast and accurate modeling of packages and boards, which is the key for developing high performance devices. With increasing complexity, thermal effects significantly impact the systems performance as well. Hence, the fast model should be able to perform electro-thermal co-simulations as well. By integrating thermal analysis with electrical simulations, we can optimize designs for efficiency without overheating issues. This thesis discusses a machine learning based approach using neural networks to generate a fast model, eliminating the need to run long simulations using EM solvers often. This helps in creating the most optimal design faster without going through many iterations. ML based fast learned model is obtained for a differential PTH. An effective way to generate datasets for training the ML model is discussed. The generated ML model shows a 200X improvement over HFSS while simulating a single design using the Inference model of the neural network. This thesis also discusses a method using machine learning to perform electro-thermal simulations. The proposed method shows a 220X speedup when compared to the two-way coupling process for electro-thermal simulations.

*To my mom, dad and my guru*

## ACKNOWLEDGMENTS

My journey with the University of Illinois started off with a huge leap of faith into an unknown territory. To find my way this far would have been impossible without the incredible people who held a guiding light at every step and helped me find my path.

It all started off with an opportunity given by my advisor, Professor José Schutt-Ainé. An opportunity to join his research group and pursue my higher education. Ever since, he has played an incredible part in shaping who I am. His teachings introduced me to ways of research. He has always been there, supporting and guiding me even through the most challenging of times. His patience, compassion and kindness are immeasurable. He trusted in me at times when I did not trust myself. He always inspires me, and I hope to learn much more from him in the future. I humbly bow down and thank him for giving me such an amazing opportunity to learn under him. I am forever grateful.

I offer my salutations at the lotus feet of my Guru, Mata Amritanandamayi Devi. She is my guiding angel. I pray that her blessings are always upon me.

My mother, Padmavathi's love and support throughout my life has been limitless. I thank her for being so courageous and doing everything at her will in giving me the best possible opportunities to help me grow. I also thank my sister, Jeeshitha for making me want to be a better person every day. I would like to thank Dr. Shekar babu and Mrs. Parvathy Ramamoorthy for their endless love and support. They keep me motivated and focused on my goals. It is their vision that led me to pursue my higher studies in such a great institution. I would also like to thank my fellow students at UIUC for helping me. I thank my friend Amulya for making me feel at home every day.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1. Problem Statement

In recent decades, the scaling of electronic packaging technologies has become a key enabler in meeting the growing demands for higher data rates, increased integration density, and improved power efficiency in modern computing and communication systems. Advances in packaging—such as 2.5D interposers, 3D stacked integrated circuits (3D ICs), and system-in-package solutions—have significantly improved system performance by reducing form factor, interconnect length, and power consumption. However, these benefits come at the cost of increased complexity in electromagnetic behavior due to high-density interconnects, tighter pitch vias, and heterogeneous integration across multiple silicon and package layers [1].

As packaging technologies evolve, signal integrity (SI) has emerged as a critical consideration in the design of high-speed systems. SI refers to the ability of a signal to propagate from the transmitter to the receiver without significant distortion, loss, or interference. In high-speed digital systems, even minor discontinuities in interconnects—such as vias, connectors, or transmission line stubs—can lead to reflection, crosstalk, jitter, and inter-symbol interference (ISI), all of which degrade system performance. These effects become especially pronounced at multi-gigabit data rates, where the signal rise/fall times are on the order of tens of picoseconds and wavelengths become comparable to physical feature sizes on the board or package [2]. As a result, accurate modeling and analysis of SI behavior in the package and board have become integral to the early stages of system design.

Traditionally, signal integrity analysis has relied heavily on electromagnetic (EM) solvers and circuit simulators that numerically solve Maxwell's equations to predict the behavior of interconnect structures. Full-wave 3D EM solvers, such as Ansys HFSS, CST Microwave Studio, etc., provide highly accurate broadband characterization of packages and PCBs by employing methods like the Finite Element Method (FEM), Finite Difference Time Domain (FDTD), or Method of Moments (MoM). These tools are capable of extracting S-parameters, impedance profiles, and time-domain responses from complex geometries and multi-layer structures, including vias, transmission lines, and connectors. Circuit-level simulators like HSPICE are often used in conjunction with extracted parasitics to evaluate eye diagrams, bit error rates (BER), and jitter performance under realistic signaling conditions [3], [4].

Despite their accuracy, these traditional EM-based methods suffer from significant computational bottlenecks. As modern packages incorporate multiple layers, high via densities, and complex transitions, the mesh size required to resolve fine geometrical details leads to an exponential increase in simulation time and memory consumption. It is not uncommon for a single high-fidelity full-wave simulation of a via or BGA breakout region to take several hours to days, especially when broadband responses over 0–100 GHz are required. Furthermore, design processes increasingly demand rapid iterations for design space exploration (DSE), sensitivity analysis, and optimization, which require hundreds or thousands of simulation runs. Performing such tasks using conventional EM solvers is not only time-consuming but often computationally infeasible [5].

Another challenge associated with traditional methods is the lack of scalability in supporting uncertainty quantification (UQ) and statistical analysis, which are essential in addressing process variations and manufacturing tolerances. Monte Carlo simulations, which require repeated runs with varied parameters, become prohibitive in terms of computational effort

when relying solely on full-wave solvers. This constraint hinders the ability to explore worst-case scenarios and robust design margins effectively. Additionally, the iterative nature of co-design—where package, board, and IC design must be tuned concurrently—further amplifies the demand for fast, accurate, and flexible analysis tools [6].

To address these limitations, there is a growing interest in developing surrogate modeling approaches that can replicate the behavior of EM solvers with a fraction of the computational cost. These include reduced-order modeling (ROM), machine learning (ML) techniques, and behavioral modeling using neural networks. Such methods aim to bridge the gap between accuracy and computational efficiency, enabling faster design cycles while retaining the fidelity required for high-performance applications.

This dissertation discusses a method using ML techniques to develop fast and accurate models for SI analysis. The design of complex interconnection schemes involves the characterization of individual components within the system. This means that the frequency response of these components needs to be analyzed. The obtained frequency response in terms of S-parameters is then also used for time-domain simulations. The S-parameters of the designed components are generally obtained using EM solvers like HFSS, which, as discussed, can be time consuming. Recent literature has shown that traditional EM solvers can be replaced by a machine learning model that is fast and accurate. The machine learning (ML) model can predict the S-parameters for a particular design space once it is trained with the data in the design space. This trained ML model generates the predicted S-parameters accurately and is much faster than the traditional EM solvers and therefore can replace the EM solvers during the design process [5].

This dissertation is structured as follows: The motivation for developing a ML model is discussed in Chapter 1 along with an overview of S-parameters and Machine Learning algorithms.

Chapter 2 discusses the neural network architecture used to develop the ML model. In Chapter 3, the effectiveness of using neural networks is tested by developing a ML model for a plated through hole package. The technique used for generating efficient datasets for training purposes is also described along with the performance results when compared to traditional simulations. Chapter 4 expands the scope of the NN framework to electro-thermal analysis by developing an ML model of a PTH that can also predict the temperature. Two-way coupling is used for generating the dataset for training the model for electro-thermal co-design. Chapter 5 summarizes the advantages of using ML models over the traditional solvers and discusses its limitations and future scope.

## 1.2. Overview of Scattering parameters and Signal Integrity (SI) analysis

Scattering parameters, commonly referred to as S-parameters, are fundamental to the characterization of high-frequency electronic systems. They describe how electrical signals behave as they encounter discontinuities in a multi-port network, such as vias, connectors, transmission lines, and package interfaces. Unlike impedance or admittance parameters, which require voltage and current measurements at each port, S-parameters relate incident and reflected voltage waves, making them particularly suitable for high-frequency and RF applications where direct current and voltage measurement is challenging [3].

S-parameters are defined in the frequency domain and are typically extracted using vector network analyzers (VNAs) or electromagnetic (EM) solvers. For a given frequency, they provide a complete description of how a network reflects and transmits signals between ports. As a result, they are extensively used in SI analysis to model and simulate the performance of interconnects in printed circuit boards (PCBs), packages, and integrated circuits (ICs). They help quantify losses, impedance mismatches, and crosstalk that affect the quality of signals propagating in the system.

Their ability to capture broadband frequency behavior makes them indispensable in high-speed digital design, where signal degradation can result in eye closure, jitter, and bit errors. As frequency increases, transmission losses typically become more pronounced due to dielectric loss, conductor skin effect, and radiation. Reflections may also increase if impedance discontinuities are present. Therefore, accurate modeling and understanding of S-parameters across a broad frequency range is critical for designing robust high-speed systems.

### 1.2.1. Definition of S-parameters

For an N-port network, let $a_i$ represent the incident wave at port $i$, and $b_i$ represent reflected wave at the same port. The S-parameter matrix **S** is defined as:

$$b_i = \sum_{j=1}^{N} S_{ij} a_j , \quad \text{for } i = 1,2, \dots, N \tag{1.1}$$

or in matrix form:

$$\boldsymbol{b} = \boldsymbol{S} \cdot \boldsymbol{a} \tag{1.2}$$

where $\boldsymbol{a} = [a_1, a_2, \dots, a_N]^T$ is the vector of incident waves, $\boldsymbol{b} = [b, b, \dots, b_N]^T$ is the vector of reflected waves, $\boldsymbol{S}$ is the N x N S-parameter matrix

Each element $S_i j$ describes the ratio of the reflected wave at port $i$ to the incident wave at port $j$, with all other ports terminated in matched loads is:

$$S_{(ij)} = (b_i / a_j)\big|_{(a_j = 0, ¥k \neq j)} \tag{1.3}$$

$S_{ii}$ is the reflection coefficient at port $i$, $S_{ij}$ is the transmission coefficient from port $j$ to port $i$.

### 1.2.2. S-parameters of a 4-port Network

A 4-port system is common in differential signaling environments such as high-speed serial links, Plated-through hole (PTH), etc. The S-matrix of a 4-port system is:

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{S_{11}} & \boldsymbol{S_{12}} & \boldsymbol{S_{13}} & \boldsymbol{S_{14}} \\ \boldsymbol{S_{21}} & \boldsymbol{S_{22}} & \boldsymbol{S_{23}} & \boldsymbol{S_{24}} \\ \boldsymbol{S_{31}} & \boldsymbol{S_{32}} & \boldsymbol{S_{33}} & \boldsymbol{S_{34}} \\ \boldsymbol{S_{41}} & \boldsymbol{S_{42}} & \boldsymbol{S_{43}} & \boldsymbol{S_{44}} \end{bmatrix} \tag{1.4}$$

Here, each term has a specific representation: $S_{11}$ is the reflection at port 1, $S_{21}$ is the transmission from port 1 to port 2, $S_{31}$ $S_{41}$ represent the crosstalk from port 1 to ports 3 and 4, $S_{34}$ is the transmission from port 4 to port 3, etc. If the system is reciprocal, then $S_{ij} = S_{ji}$ and if the system is lossless, then the S-matrix is unitary ($S^H S = I$).

S-parameters provide a fundamental and efficient way to model and analyze high-frequency behavior in multi-port systems. In the domain of signal integrity, they enable designers to understand and mitigate losses, reflections, and crosstalk in interconnect structures with great accuracy. Their matrix-based formulation and compatibility with both frequency- and time-domain simulation frameworks make them indispensable for modern high-speed system design. Therefore, we use S-parameters for SI analysis of packages in the frequency domain in this work.

## 1.3. Review of Machine Learning Algorithms

Machine learning (ML) is a branch of artificial intelligence that allows systems to automatically learn from data and improve from experience without being explicitly programmed. It has become a cornerstone technology in a wide range of applications, from natural language processing to computer vision and, more recently, electronic design automation and SI analysis.

At the heart of many ML models lie neural networks—computational structures inspired by the human brain. Neural networks have demonstrated state-of-the-art performance in tasks involving pattern recognition, regression, classification, and complex nonlinear modeling.

### 1.3.1. Fundamentals of Neural Networks

Neural networks are composed of layers of interconnected units called artificial neurons or nodes. These neurons mimic the function of biological neurons by aggregating weighted inputs and applying an activation function to produce an output.

An artificial neuron takes multiple inputs, each associated with a weight, and computes a weighted sum. A bias term is added to shift the activation function. This sum is passed through an activation function to produce the final output. Mathematically, the output of a single neuron can be represented as:

$$z = \sum_{i=1}^{n}(w_i * x_i) + b \tag{1.5}$$

$$y = \varphi(z) \tag{1.6}$$

where $x_i$ are the inputs, $w_i$ are the weights, b is the bias, z is the linear combination, and φ(z) is the activation function.

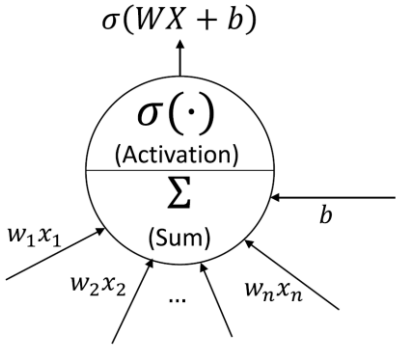



Figure 1.1: Single artificial neuron [6]

Common activation functions include sigmoid in Equation (1.7), rectified linear unit (RELU) in Equation (1.8), hyperbolic tangent in Equation (1.9) and exponential linear unit (ELU) in Equation (1.10).

$$\varphi(z) = \frac{1}{1+e^{-z}} \tag{1.7}$$

$$\varphi(z) = max(0, z) \tag{1.8}$$

$$\varphi(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \tag{1.9}$$

$$\varphi(z) = \begin{cases} z \ \ if \ z > 0 \\ \alpha * (e^{z} - 1) \ where \ \alpha > 0 \end{cases} \tag{1.10}$$

### 1.3.2. Simple Neural Network Framework

Neural networks are organized into layers: an input layer, one or more hidden layers, and an output layer. Each layer contains several neurons, and each neuron in a layer is connected to every neuron in the subsequent layer, forming a fully connected network or feed-forward neural network (FNN).

### 1.3.3. Phases involved in Machine Learning

#### 1.3.3.1. Training Phase

Training is the process through which a neural network learns the optimal weights and biases that minimize the difference between predicted and actual outputs. This process can be broken into multiple stages:

Stage 1: Forward propagation

In forward propagation, input data is passed through the network layer by layer. At each neuron, the weighted sum of inputs is computed and passed through the activation function to produce the output as per Equation (1.11). The final output layer produces the network's prediction. The forward propagation operation can be seen in Figure 1.2.

Let $\boldsymbol{x}$ be the input vector and $\hat{y}$ be the predicted output:

$$\hat{y} = f(\mathbf{x}; \mathbf{W}, \mathbf{b}) \tag{1.11}$$

Where $\mathbf{W}$ and $\boldsymbol{b}$ represent the set of all weights and biases in the network respectively.

Stage 2: Reduce Loss

The network's performance is evaluated using a loss function, which quantifies the difference between the predicted output $\hat{y}$ and the true output $y$. Common loss functions include mean squared error (MSE) in Equation (1.12) and normalized mean squared error (NMSE) in Equation (1.13).

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1.12}$$

$$NMSE = \frac{MSE}{Var(y)} \tag{1.13}$$

where MSE is the Mean Squared Error and Var(y) is the variance of the true values. NMSE provides a normalized metric that makes performance evaluation more consistent across different datasets.

Stage 3: Backpropagation

Backpropagation is an algorithm used to update model weights by computing gradients of the loss function with respect to each parameter. The process involves computing the loss, calculating the gradient of the loss with respect to each parameter using the chain rule, propagating the error backward through the layers, and updating weights using an optimization algorithm. The update rule using gradient descent is given in Equation (1.14). The backpropagation operation can be seen in Figure 1.2.

$$w_{ij}{}^{t+1} = w_i{}^{t} - \eta \frac{\partial L}{\partial w_{ij}} \tag{1.14}$$

where $\eta$ is the learning rate, $\frac{\partial L}{\partial w_{ij}}$ is, the gradient of the loss with respect to weight $w_{ij}$ and $t$ denotes the iteration step.

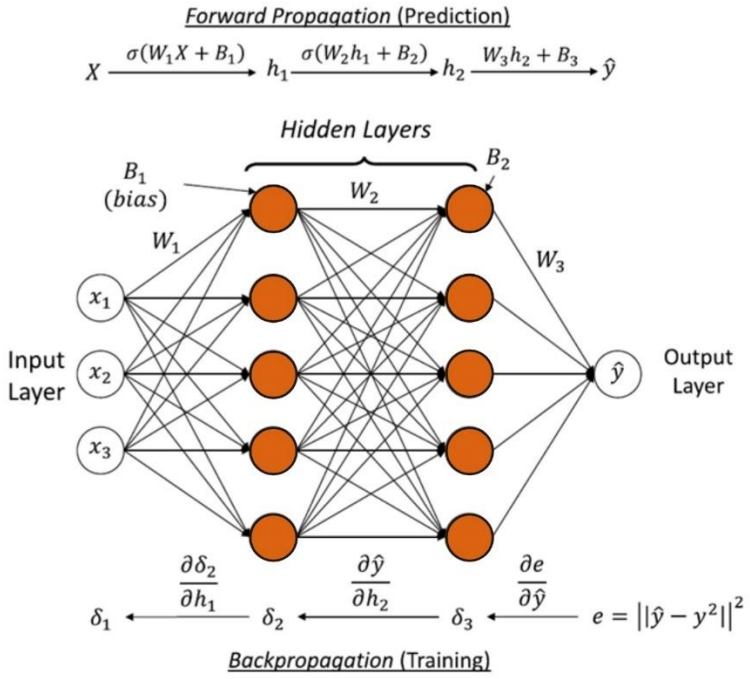



Figure 1.2: Forward and Backpropagation in Feed Forward Neural Network [6]

#### 1.3.3.2. Inference Phase

Once the model is trained, it enters the inference phase, where it is used to make predictions on new, unseen data. In this phase, only forward propagation is used—no gradients or weight updates are performed. The model takes in an input vector, computes activations layer by layer, and outputs the prediction. Inference is typically fast and efficient, making trained neural networks suitable for deployment in real-time applications

Neural networks represent a powerful class of machine learning models capable of approximating highly complex functions. The core component, the artificial neuron, enables networks to learn nonlinear relationships through a process involving forward propagation, error measurement via a loss function, and weight updates through backpropagation. The flexibility and adaptability of neural networks make them especially suitable for domains where traditional models fail to capture underlying data patterns.

There are several types of neural networks that are available namely the Feed-forward neural networks (FNNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Autoencoders and Variational autoencoders (VAEs) and many more. Each of these neural networks provide useful for a different type of application. Here, FNNs and CNNs are discussed in detail.

### 1.3.4. Feed-Forward Neural Networks (FNNs)

A Feed-forward Neural Network (FNN) is the simplest form of artificial neural network wherein the information moves only in one direction—from input nodes through hidden layers to output nodes. There are no cycles or loops in the network. Each layer in an FNN consists of neurons

that are connected to all neurons in the subsequent layer, making the architecture fully connected as shown in Figure 1.2.

The structure of a typical FNN includes an input layer that receives the raw data, one or more hidden layers where intermediate processing occurs, and an output layer that delivers the final prediction or classification. At each neuron, a weighted sum of inputs is calculated and passed through a nonlinear activation function, such as ReLU, sigmoid, or tanh, to introduce non-linearity into the model. This allows FNNs to learn complex relationships between input features and output targets.

FNNs are particularly suited for supervised learning tasks, where they are trained on labeled data. One of the core applications of FNNs is in regression problems, where the goal is to predict a continuous-valued output from a set of input features. Another key application area is classification, where FNNs are used to assign input data to predefined categories. One of the key advantages of FNNs is their simplicity and interpretability. Because of their straightforward architecture, FNNs are relatively easy to implement and train. They are also versatile, capable of approximating any continuous function given sufficient neurons and layers. Moreover, the lack of feedback loops makes FNNs stable during training and suitable for static datasets, where temporal dependencies are not involved.

Despite their strengths, FNNs also exhibit several limitations. One major drawback is their inefficiency in handling data with spatial or temporal structures, such as images or time series. In such cases, FNNs fail to exploit local correlations, which results in a large number of parameters and reduced performance. They also tend to struggle with scalability; as the size of the input increases, the number of connections—and thus computational costs grow rapidly. Furthermore,

FNNs are prone to overfitting, especially when trained on small datasets without proper regularization techniques [7].

In recent years, alternative architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been developed to overcome these limitations by incorporating mechanisms for spatial feature extraction and temporal memory, respectively. Nonetheless, FNNs remain a foundational architecture in neural network research and continue to be widely used, particularly in structured data modeling and function approximation tasks.

### 1.3.5. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed to process data with a grid-like topology, such as images or time series. Inspired by the structure and functionality of the animal visual cortex, CNNs are particularly effective for image classification, object detection, and feature extraction tasks. Unlike traditional feed-forward neural networks, which treat all inputs with equal connectivity, CNNs take advantage of spatial locality by enforcing a sparse connectivity pattern between neurons [1].

The different layers in the CNN architecture are described below:

1) Convolutional Layers

The core building block of a CNN is the convolutional layer, which applies a set of learnable filters (or kernels) that slide across the input data to extract spatial features. Each filter is convolved with a portion of the input data (receptive field) to produce a feature map, which captures local patterns such as edges, corners, or textures in images.

Mathematically, for a 2D input $I$ and a kernel $K$ of size $m\ x\ n$, the convolution operation producing an output $O$ at position $(i, j)$ is given by:

$$O(i,j) = \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} K(u,v).I(i+u, j+v) \quad (1.15)$$

This operation is repeated for each kernel across the spatial dimensions of the input. In deep CNNs, multiple filters are stacked, allowing the model to learn hierarchical features—from low-level textures in early layers to high-level objects in deeper layers [7].

2) Activation layers

After the convolutional operation, the output is passed through a nonlinear activation function to introduce non-linearity into the model, enabling it to learn complex mappings. Some of the most used activation functions are listed in section 1.3.1.

3) Pooling layers

To reduce the spatial dimensions of the feature maps and improve computational efficiency, CNNs incorporate pooling layers between successive convolutional layers. Pooling helps in making the representations approximately invariant to small translations and distortions in the input.

The most common pooling operation is max pooling, which selects the maximum value from a region of the feature map. For example, in a $2 \times 2$ max pooling operation, the output value for each sub-region is:

$$P(i,j) = \max\{O(2i, 2j), O(2i, 2j+1), O(2i+1, 2j), O(2i+1, 2j+1)\} \quad (1.16)$$

Other variants include average pooling and global pooling, which take the average or overall maximum, respectively, across spatial dimensions.

A typical CNN architecture consists of multiple blocks of convolutional, activation, and pooling layers, followed by one or more fully connected layers at the end. These final layers integrate high-level features extracted from the previous layers to perform classification or regression tasks. The network ends with an output layer, often using a softmax function for classification to convert the final outputs into probabilities over the target classes.

CNNs offer several key advantages. They drastically reduce the number of parameters compared to fully connected networks by sharing weights across spatial locations. This allows them to scale better with input size and makes them more efficient. CNNs also learn spatial hierarchies, making them highly effective at capturing both low-level and high-level features in data. These capabilities of the CNN make it highly advantageous for signal integrity modeling as spatial and frequency patterns are present in the data.

# CHAPTER 2

# MACHINE LEARNING FRAMEWORK

ML has proven to be very useful for the semiconductor industry, especially in packaging. A major limit in modern electronic design automation (EDA) is design respins due to hardware complexity. Many of the failures causing respins can be attributed to insufficient modeling capability where using simulations in the design loop is oftentimes too slow and frustrating. With design complexity and performance increasing, any approximations or assumptions made during the design process can only lead to errors. In such scenarios, using Machine Learning can be very beneficial. Therefore, a model-based design paradigm can be developed where fast to evaluate "learned" model replaces the conventional "slow" model in design and design optimization. This is shown in Figure 2.1 where the data from the "slow" but detailed simulator are used to develop a machine learned "fast" model. The fast model is quick to compute, is expected to have the same accuracy as the detailed simulator and can therefore be used in the design loop for simulation-based design and optimization. Moreover, since the fast model is expected to capture the entire design space including process variations, the probability of introducing errors can be minimized.

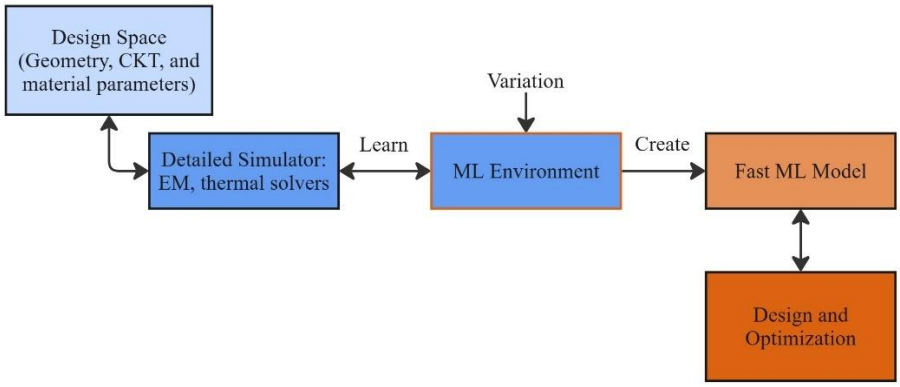



Figure 2.1: Machine Learning based Design Flow

## 2.1 Process for developing a fast ML model

The analysis of high-speed complex systems is usually performed using simulation tools that model the real-time physical parameters to provide an accurate measure of the system behavior. These simulators—such as full-wave electromagnetic (EM) solvers, finite element methods, or circuit-level SPICE simulators—provide precise results but are computationally expensive and time-consuming, especially when multiple parameter sweeps or optimization runs are required. To overcome these limitations, a fast surrogate model based on machine learning (ML) can be developed by learning from the data generated by the slow simulator.

The process begins with data generation using the traditional simulator. In this stage, simulation experiments are performed by varying key input parameters across their feasible ranges. These parameters may include geometric features, material properties, frequencies, or source/load conditions. For example, in signal integrity analysis, simulations might output S-parameters or eye diagrams for varying trace widths, spacings, and dielectric constants. The result is a high-quality dataset that pairs input parameters with accurate simulation outputs.

Once the dataset is prepared, data preprocessing is carried out. This involves normalizing or standardizing the input features, removing redundant or noisy data, and splitting the dataset into training, validation, and testing subsets. Careful preprocessing ensures that the learning model receives clean, well-scaled data, which is crucial for stable and accurate training.

The next step is model selection and training. A suitable learning algorithm is chosen based on the nature of the data and the problem. Common choices include feed-forward neural networks (FNNs) for structured input-output mappings, convolutional neural networks (CNNs) for spatial data, and Gaussian processes or support vector machines for smaller datasets. The model is trained using simulation data, typically by minimizing a loss function such as mean squared error (MSE)

or normalized mean squared error (NMSE). During training, the model learns the underlying functional relationship between the input parameters and the simulation outputs.

An essential aspect of the development process is model validation and hyperparameter tuning. The model is evaluated on the validation set to monitor its generalization ability and avoid overfitting. Techniques such as cross-validation, dropout, and early stopping are often used. Hyperparameters, such as the number of hidden layers, learning rate, batch size, and activation functions, are optimized to improve performance.

Once trained and validated, the ML model enters the inference phase, where it can quickly predict outputs for new, unseen input parameters in a fraction of the time required by the original simulator. This surrogate model thus acts as a real-time approximation engine, making it highly suitable for tasks like design space exploration, sensitivity analysis, and optimization [5] [7].

This surrogate modeling approach offers significant advantages: orders-of-magnitude reduction in computation time, faster prototyping, and the ability to perform large-scale optimization and what-if analysis. However, it also requires careful handling to ensure that the surrogate model does not extrapolate beyond the range of its training data, as this can lead to inaccurate predictions.

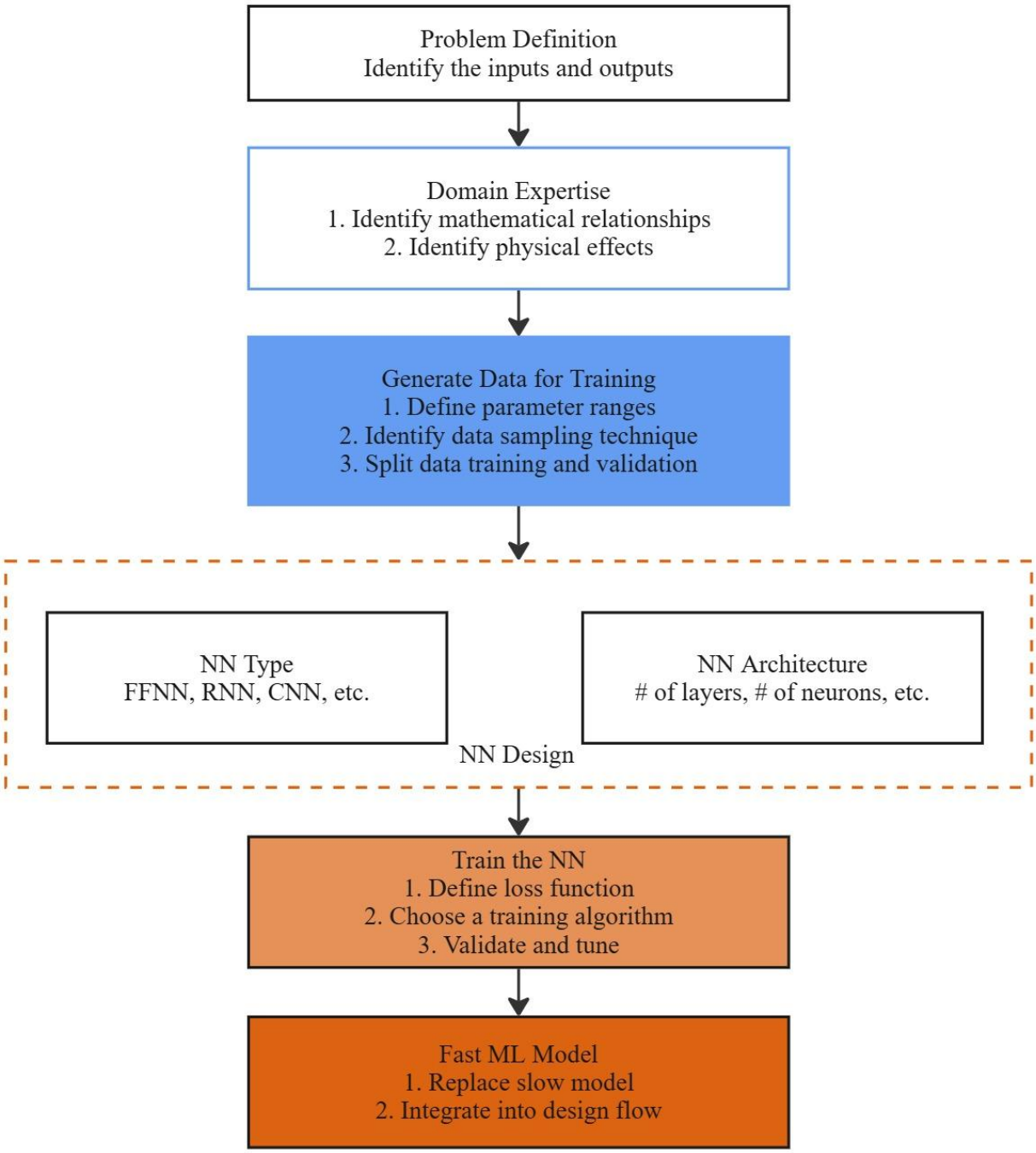



Figure 2.2: Flowchart for developing a Fast NN Model

## 2.2 Neural Network Architecture

NN architecture was first proposed in [5]. Spectral Transposed Convolutional Neural Network (STCNN) architecture is used as our ML model. The STCNN is a specialized neural network architecture that extends traditional convolutional models to operate in the frequency domain, particularly for tasks involving the prediction and reconstruction of frequency-dependent characteristics such as S-parameters in high-speed interconnects. This architecture combines feed-forward neural network layers for learning high-level representations with transposed convolutional layers to enable frequency-domain feature expansion and resolution enhancement.

The STCNN architecture generally consists of three main stages: the input embedding or encoding layers (i.e., feed-forward layers), the transposed convolutional layers and the output reconstruction or mapping layers. This design is particularly suited for spectral regression problems, where the goal is to model or generate frequency-dependent responses such as the real and imaginary parts of S-parameters over a frequency sweep.

### 2.2.1 Feed-Forward Layers (Encoders)

The feed-forward layers in an STCNN serve as the initial feature extractors. These layers are typically fully connected (dense) and are responsible for mapping input features—such as geometrical parameters, material properties, and other design variables—to a high-dimensional latent space.

$$\mathrm{h}^{(l)} = \phi\left(\mathrm{W}^{(l)}.\mathbf{h}^{(l-\mathbf{1})} + \mathrm{b}^{(l)}\right) \tag{2.1}$$

Where $\mathbf{h}^{(l)}$ is the output of layer $l$, $\mathbf{W}^{(l)}$is the weight matrix, $\mathbf{b}^{(l)}$ is the bias vector, $\phi$ is the activation function, $\mathbf{h}^{(0)} = x$

This stage allows the network to extract and encode interactions between various input parameters, essentially learning a compact representation that is more meaningful for the spectral reconstruction task. These layers reduce the complexity of raw inputs and transform them into a latent code that encodes meaningful representations required for the spectral reconstruction. The final dense layer outputs a latent code $\boldsymbol{x}$ which is a vector with abstract high-level features. This space serves as the input to the decoder portion of the network.

The latent vector $\boldsymbol{x}$ is reshaped into a 2D feature map to serve as input to the transposed convolutional layers. For instance:

$$Z_{\text{reshaped}} \in R^{C \times W} \tag{2.2}$$

where C is the number of feature maps and W is the width (or sequence length in 1D convolutions) This transformation allows the network to treat the frequency-domain data generation as a sequence modeling task, where each output location corresponds to a specific frequency bin.

### 2.2.2 Transposed Convolutional layers (Decoders)

The transposed convolutional layers form the core of the STCNN architecture. Unlike standard convolutional layers that reduce spatial resolution, transposed convolutions are used to upsample or expand the feature maps to reconstruct output data of higher resolution—in this case, frequency responses. These layers are crucial for generating a smooth and continuous spectrum from a compact latent vector. Each layer learns to generate higher-resolution features from lower-resolution ones.

A 1D transposed convolution is defined as:

$$y[n] = \sum_{k=0}^{K-1} x[k] \cdot w[n - k \cdot s] \tag{2.3}$$

where $y[n]$ is the output signal, $x[k]$ is the input signal, $w$ is the filter (kernel) and $s$ is the stride.

Unlike standard convolutions that aggregate local features, transposed convolutions distribute information across a larger output space. The stride controls how much the input is "spread out." If padding and stride are carefully chosen, the final output length matches the target frequency resolution.

These layers reconstruct the frequency-dependent S-parameter response across the spectrum, ensuring continuity and smoothness between neighboring frequency points.

After each transposed convolution, activation functions (typically ReLU or ELU) are applied to introduce non-linearity and help the model learn complex patterns. These activations are crucial to ensure the model does not reduce to a linear mapping, especially for intricate behaviors like resonance or high-frequency losses in S-parameter curves.

### 2.2.3 Output Mapping layer

The final layer of an STCNN is often a linear layer (without nonlinearity), which maps the expanded feature representation to the target spectral output, such as the magnitude and phase, or real and imaginary parts of S-parameters across a frequency range. Figure 2.3 shows the entire STCNN architecture framework.

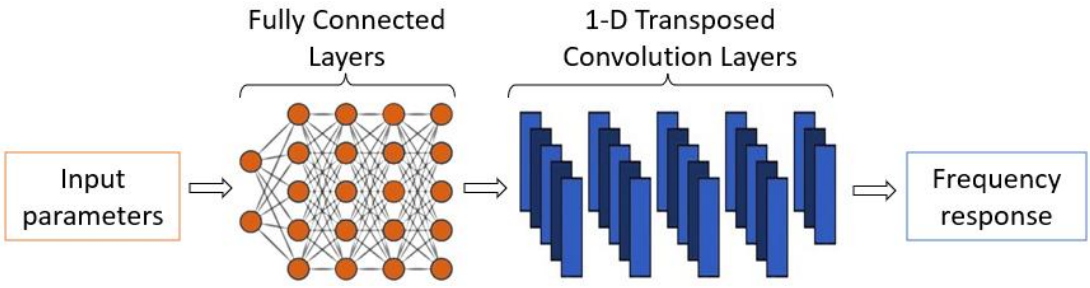



Figure 2.3: STCNN architecture

## 2.3 Causality and Passivity Enforcement

In high-speed interconnect modeling, ensuring physical consistency of predicted responses is crucial. Two important physical properties of S-parameters are causality and passivity. When using machine learning models (like STCNNs or other neural networks) to predict S-parameters, there is a risk that these predictions may violate basic physical laws, leading to non-realizable or unstable systems. To address this, Causality Enforcement layer (CEL) and Passivity Enforcement Layer (PEL) are incorporated into the neural network architecture or post-processing pipeline. Implementation of CEL and PEL are discussed in [6][8].

Causality ensures that the output of a system depends only on present and past inputs, not future ones. In the frequency domain, this is reflected by the Kramers-Kronig relations between the real and imaginary parts of the S-parameters.

Passivity implies that the system does not generate energy. This requires that the magnitude of the S-parameter matrix does not exceed unity for passive devices (i.e., no gain), and the real part of impedance or admittance matrices remains positive.

Machine learning models, especially deep neural networks, are data-driven and not inherently bound by these physical laws. Without explicit constraints, the models can produce non-causal or non-passive S-parameters, leading to simulation failures, instabilities, or non-physical interpretations when these are used in system-level EDA tools.

### 2.3.1 Causality Enforcement

#### 2.3.1.1 Frequency domain interpretation of causality

Causality in the time domain translates into analytic constraints in the frequency domain. If $H(f)$ is the frequency response (such as an S-parameter), then for it to be causal, the real and imaginary parts must be Hilbert transform pairs.

Mathematically, the Kramers-Kronig relations describe this:

$$\mathrm{Re[H(f)]} = \frac{1}{\pi}\mathcal{P}\int_{-\infty}^{\infty}\frac{\mathrm{Im[H(f')]}}{\mathrm{f'-f}}\,\mathrm{df'} \tag{2.4}$$

$$\mathrm{Im[H(f)]} = -\frac{1}{\pi}\mathcal{P}\int_{-\infty}^{\infty}\frac{\mathrm{Re[H(f')]}}{\mathrm{f'-f}}\,\mathrm{df'} \tag{2.5}$$

where $\mathcal{P}$ indicates the Cauchy principal value of the integral

#### 2.3.1.2 Implementation in neural networks

A Causality Layer can be implemented as a penalty term added to the loss function during training that measures the deviation from the Hilbert pair relationship. Alternatively, enforce the frequency response to be the Fourier Transform of a time-domain response that is strictly zero for $t < 0$. This can be done by constraining the network to output a time-domain impulse response $h(t)$ and applying a zero-mask for negative $t$, then transforming to frequency domain using FFT.

Causality loss example is given below:

$$\mathcal{L}_{causality} = |\mathrm{Re[H(f)]} - \mathcal{H}(\mathrm{Im[H(f)]})|_2^2 + |\mathrm{Im[H(f)]} + \mathcal{H}(\mathrm{Re[H(f)]})|_2^2 \tag{2.6}$$

Where $\mathcal{H}$ represents the Hilbert transform operator

### 2.3.2 Passivity Enforcement

#### 2.3.2.1 Passivity conditions

A multi-port S-parameter matrix $S(f)$ is passive if:

i. $||S(f)||_2 \leq 1$ for all $f$

ii. For all input vectors $v$, the output power $P_{\text{out}} \leq P_{\text{in}}$

Alternatively, the scattering matrix must satisfy:

$$S(f)^T S(f) \preceq I \tag{2.7}$$

That is, the Hermitian part of $\boldsymbol{S}(f)^T \boldsymbol{S}(f)$ must not exceed the identity matrix

#### 2.3.2.2 Implementation in Neural Networks

To ensure passivity, various methods can be used. The first method is by adding a projection layer: After each prediction, the output S-parameter matrix is projected onto the space of passive matrices. This can be done via eigenvalue decomposition of the Hermitian matrix and clamping eigenvalues to the allowable range. The second method is by constrained training: Train the network using spectral normalization or Lipschitz constraints to bound the network's output response. The third method is using loss function penalty: This penalizes S-matrix outputs whose gain exceeds 1. The loss term can be given as follows:

$$\mathcal{L}_{passivity} = \sum_f \max(0, |\boldsymbol{S}(f)|_2^2 - 1)^2 \tag{2.8}$$

### 2.3.3 Integration in Network Architecture

These enforcement mechanisms can be embedded as dedicated enforcement layers or post-processing filters applied to the network output or differentiable constraints included during training via augmented loss functions. The total loss becomes:

$$\mathcal{L}_{total} = \mathcal{L}_{MSE} + \lambda_1 \mathcal{L}_{causality} + \lambda_2 \mathcal{L}_{passivity} \quad (2.9)$$

where $\lambda_1$ and $\lambda_2$ are weights that control the influence of each constant

Causality and passivity are essential physical constraints for S-parameter models to be valid and usable in real-world simulations. Neural networks trained without these constraints can produce responses that violate basic physical principles, making them unsuitable for design or verification. Incorporating Causality and Passivity Enforcement Layers ensures that the network outputs can be directly used in EDA tools and SPICE-like simulators, frequency responses exhibit correct phase and energy behavior, and the surrogate models are robust, stable, and physically realizable. Therefore, our entire Neural Network architecture consists of the S-TCNN along with the CEL and the PEL layers as shown in Figure. 2.4.

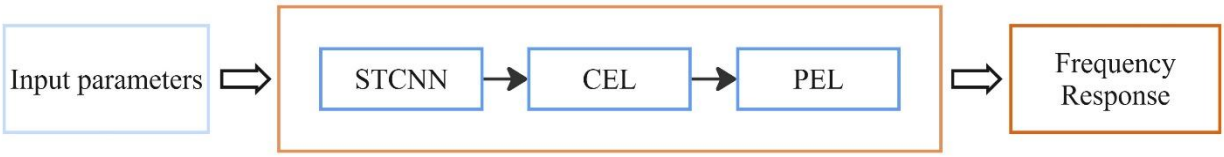



Figure 2.4: Neural Network architecture with CEL and PEL

# CHAPTER 3

# HIGH-SPEED SIGNAL INTEGRITY ANALYSIS OF A CHANNEL IN FREQUENCY DOMAIN USING NN

## 3.1 Introduction

With the increasing demand for high performance products, the complexity of electronic systems has been growing steadily. Complex high-performance systems scale with device technology which makes the design of packages and boards more tedious. Hence, there is a need for fast and accurate methods for adaptation that can reduce the number of iterations during the design process. At present, engineers are restricted by the large computational time of the traditional electromagnetic (EM) solvers while making design choices. Design closure can only be attained after multiple iterations through time-consuming EM extractors. This reduces the ability to explore entire design spaces for optimizing geometry, materials and other parameters that comprise the board layout. This traditional process of board design has been summarized in Figure 3.1.

Using ML to develop a fast NN model to replace the traditional EM solvers allows the designer to explore the entire design space and come up with the most optimal design without having to go through multiple iterations. This ML model can be incorporated in the early design stages to narrow down the design space for a particular set of design rules. The new design flow using ML model is shown in Figure 3.2. The ML model can provide the S-parameters data for any set of parameters within a few milliseconds, thereby making optimization more efficient.

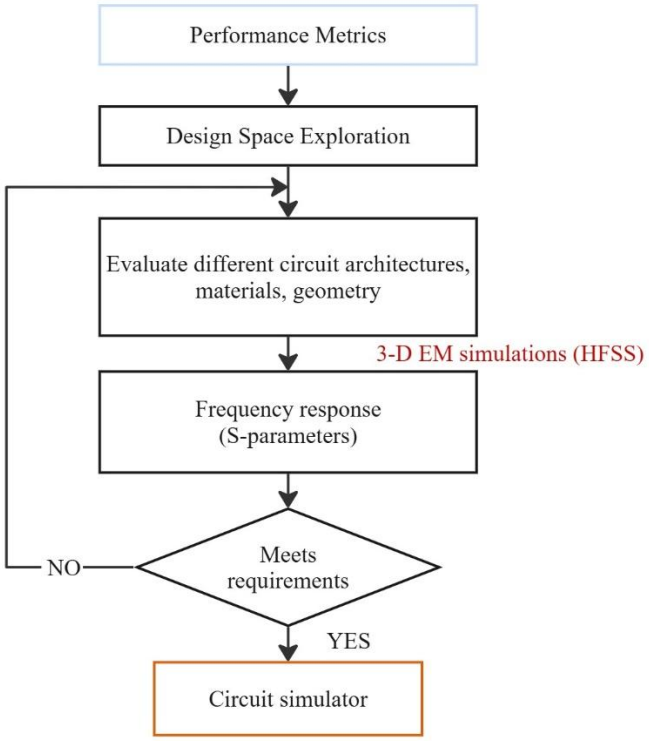



Figure 3.1: Flowchart of the traditional approach to board design

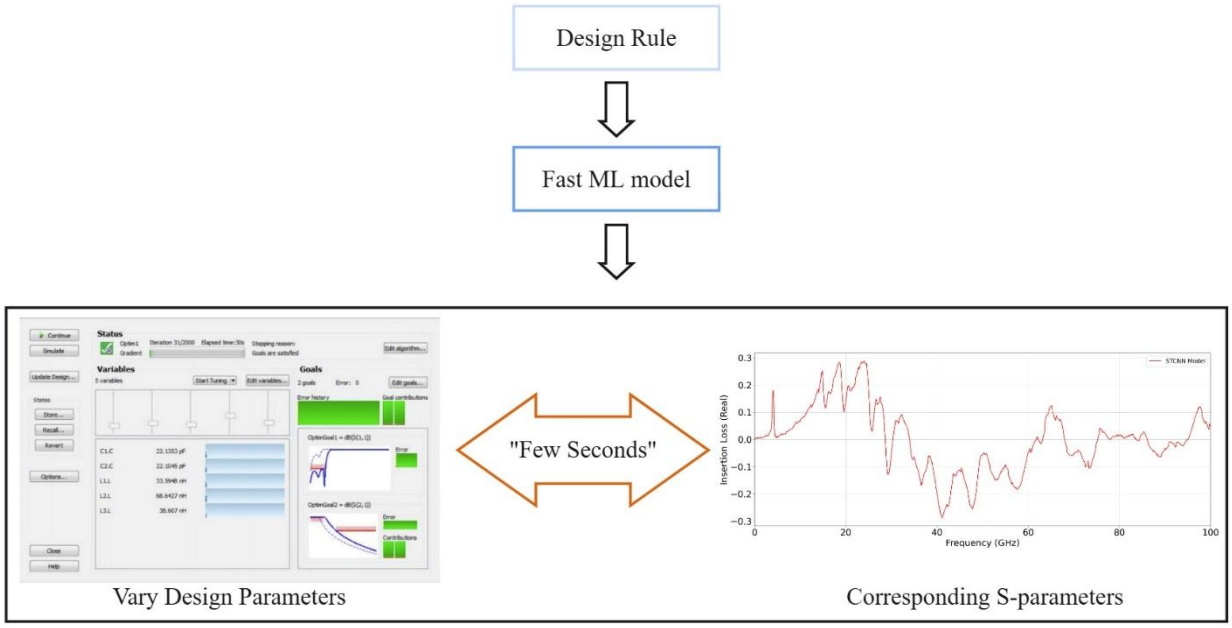
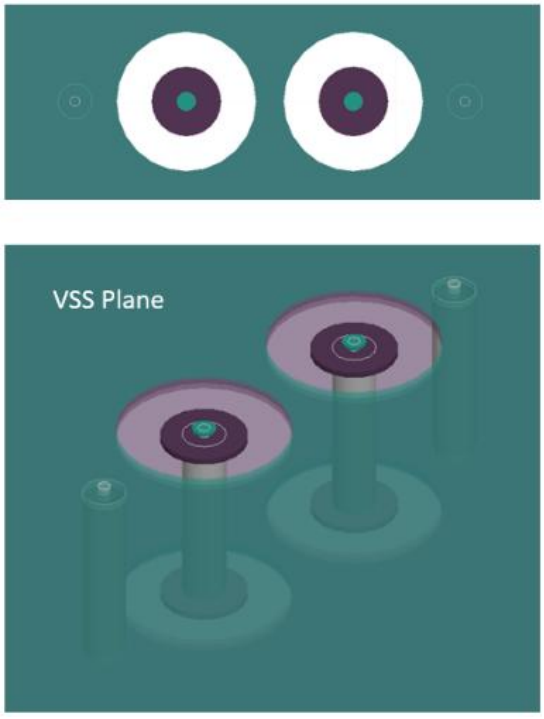


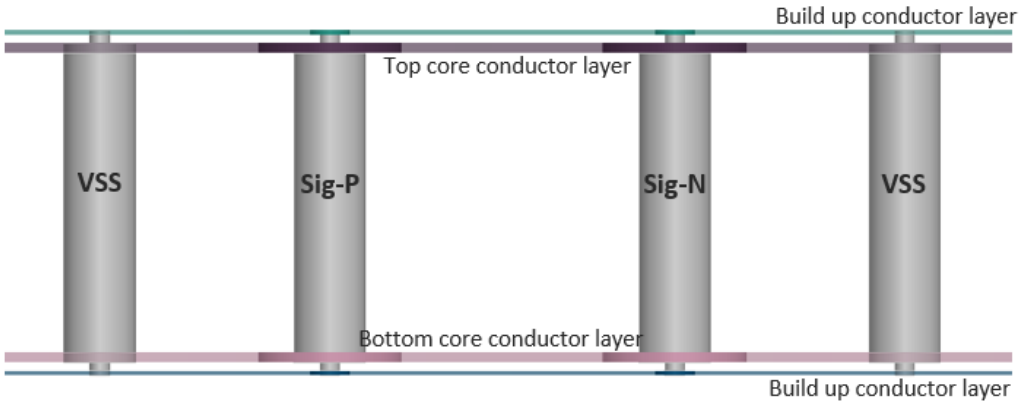

Figure 3.2: Flowchart of design process using ML model

## 3.2 Training dataset preparation for SI analysis

A simple differential PTH layout was created in Ansys HFSS 3D layout to perform the EM simulation to generate the necessary S-parameter files as the input dataset for training the ML model. Figure 3.3 shows the top and cross session physical view of the PTH layout. The PTH layout included the conductor layers above and below the core substrate as well as with the on top build up conductor layer at the top and bottom side. All four of these conductor layers were assigned as ground planes surrounding the PTH and μvia pads. Ground PTH was inserted next to the differential pair PTHs as the return path. All the supported parameters were set as variables in the tool so that it can be easily varied without re-drawing the layout for each dataset generation. Some examples of the generated data set with different physical configurations are captured in Figure 3.4.




(a)




(b)

Figure 3.3: Physical view of the PTH modelling with the HFSS 3D layout. a) PTH Top and 3D view, b) PTH cross section view

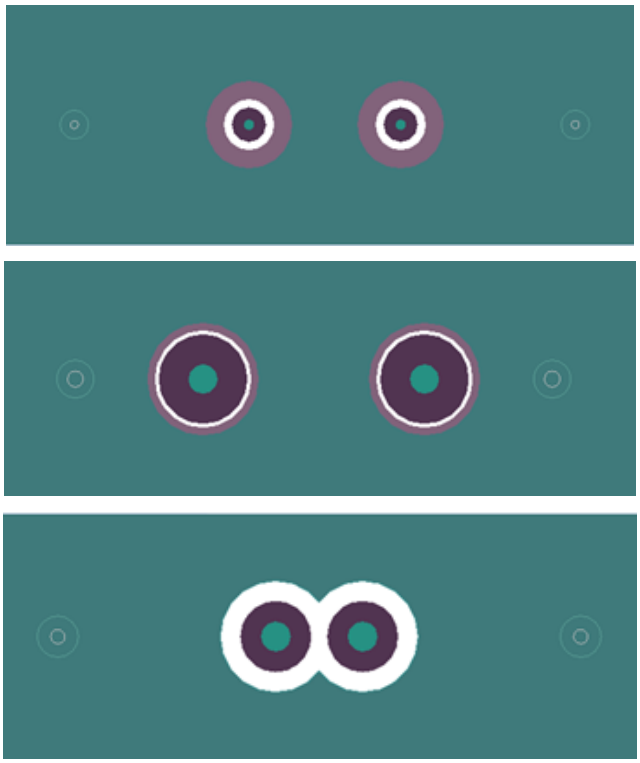


Figure 3.4: Examples of the generated dataset with different physical configurations

With the number of supported parameters and their wide range of value coverage, total combination of the use cases can be significant. With consideration of the EM tool runtime and computing resources, it is time consuming to generate a large dataset. Hence only a limited number of samples will be generated, which ensures good coverage of use case combinations to obtain a robust dataset for the training. Biasing and underfit issues during training are common problems if the generated dataset was limited and not well distributed across the supported parameters space. In this application, Quasi-random generator was used to generate a quasi-random dataset with highly uniform samples across all parameters space. Sobol sequences were chosen as the quasi-random sequence for the generator. *Sobol sequences* are a particularly common example of quasi-random sequences. They are deterministic sequences of numbers that converge quickly to a uniform distribution. The computational cost of Sobol sampling is relatively low, only marginally more expensive than random sampling and significantly cheaper than Latin Hypercube Sampling. Sobol sequences often show superior performance in comparison with random and LHS sampling.

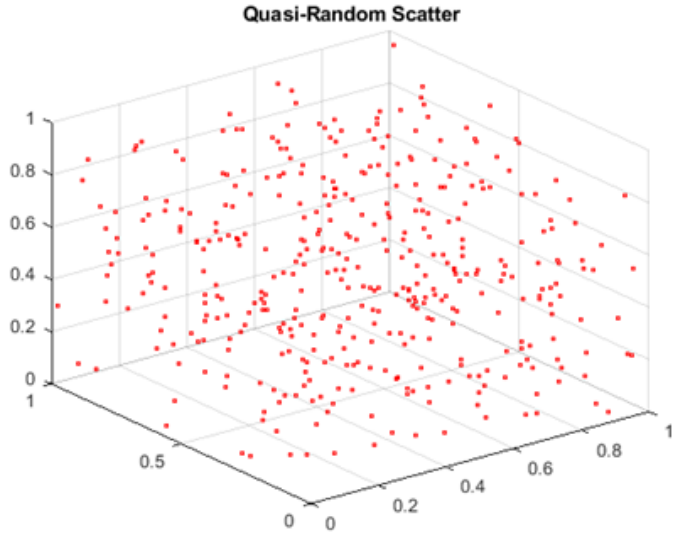



Figure 3.5: 3 out of 9 parameters Quasi-random scatter plot for the PTH modelling

9 parameters were varied within a certain range of minimum and maximum values as shown in Table 3.1. Figure 3.5 illustrates the quasi-random plot for the 3 out of the 9 supported parameters. It minimized the discrepancy among the distribution of the generated data points. A total of 393 samples were generated for this application based on the quasi-random data points. EM simulation time for each model took on average around 40 minutes to complete.

TABLE 3.1 VARIABLE PARAMETERS OF THE PTH FOR SI ANALYSIS

| | ***Parameter*** | ***Unit*** | ***Min*** | ***Max*** |
|---|---|---|---|---|
| 1 | µVia Diameter | µm | 30 | 70 |
| 2 | µVia Pad Diameter | µm | 31 | 140 |
| 3 | µVia Antipad Radius | µm | 100 | 500 |
| 4 | PTH Pitch | µm | 300 | 1200 |
| 5 | Core Thickness | µm | 100 | 1200 |
| 6 | PTH Diameter | µm | 100 | 250 |
| 7 | PTH Pad Diameter | µm | 110 | 500 |
| 8 | PTH Antipad Radius | µm | 50 | 500 |
| 9 | Signal-Ground Via Pitch | µm | 300 | 1200 |

## 3.3 Simulation Setup

Generating a machine learning model using STCNN architecture consists mainly of two parts, namely training and validation. During the training process, the ML model learns the best function that maps the relationship from the input parameters to the output parameters. Once the training process is complete, we validate the model for minimal error between the mapping of the input parameters to the output parameters. Once the training and validation is complete, we have our learned machine learning model that can replace the EM solvers. We then use this learned ML

model to generate the output parameters for any set of input parameters within the trained design space. This process is called *inference* as shown in Figure 3.6.

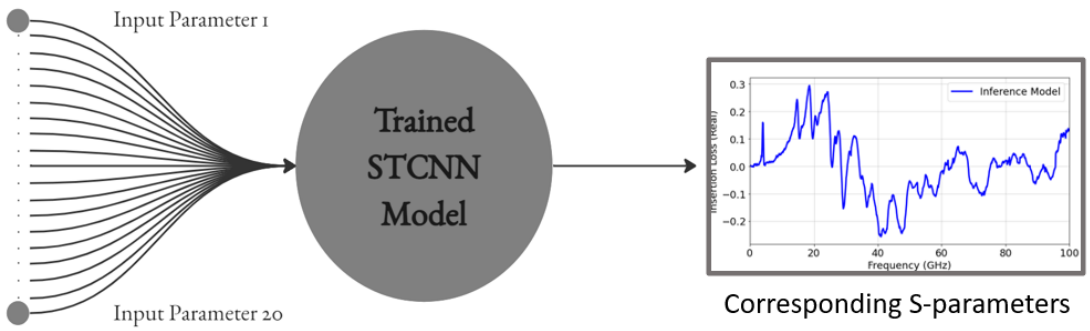



Figure 3.6: Inference model

The dataset for the PTH, which consists of 393 samples, was used to run HFSS simulation to generate their corresponding 4-port S-parameters. The simulation was run between 0.1GHz to 100GHz with a 100MHz step. The input data consisted of 9 variable geometric parameters along with 11 parameters that were set to be a constant value. These additional parameters were included with the intent of extending the usage of this ML model to a larger design space with other parameters depending on the user's preference. Hence a total of 20 input parameters are mapped to 4-port S-parameters. To reduce the output dimensionality, as the PTH structure used is partially symmetric and reciprocal, only the real and imaginary parts of the frequency responses of $S_{11}$, $S_{12}$, $S_{13}$, $S_{14}$, $S_{33}$ and $S_{34}$ are used. As we have 1000 frequency points and 12 responses, the output dimensionality is 12000. The S-TCNN model is used to generate a learned ML model that maps the 20-D input space to a 12000-D output space. Out of the 393 samples in the dataset, 390 samples were used for the training process and 3 samples were used for validation. Figure 3.7 shows the 9 variable input parameters with 390 training samples shown as black dots and 3 validation samples shown as colored dots.

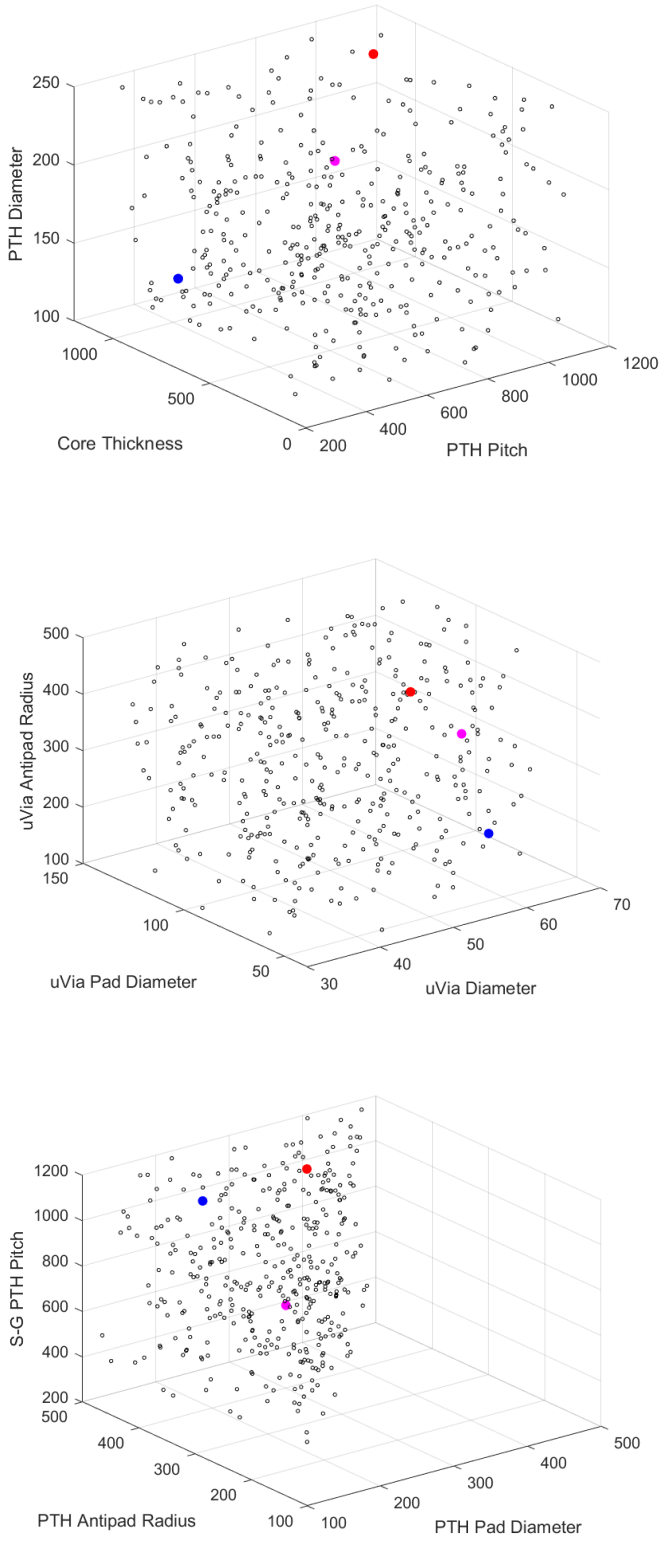


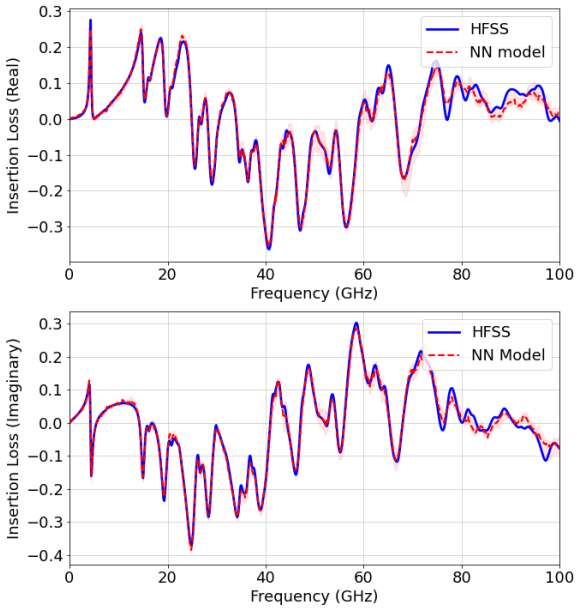

Figure 3.7: 9 variable input parameters with 390 training samples (black dots) and 3 validation samples (colored dots)

S-TCNN architecture has been tested for a several layers for various number of neurons in each layer. The architecture which was able to determine a good learnable model is used. The S-TCNN had four fully-connected layers with 50 neurons (20-50-50-50-50) followed by five 1-D transposed convolutional neural networks each having 50 channels. The kernel size was 39,8,6,4 and 2 for the layers respectively. The stride was 1,2,2,4 and 2 respectively. ELU activation function was used for all the models.

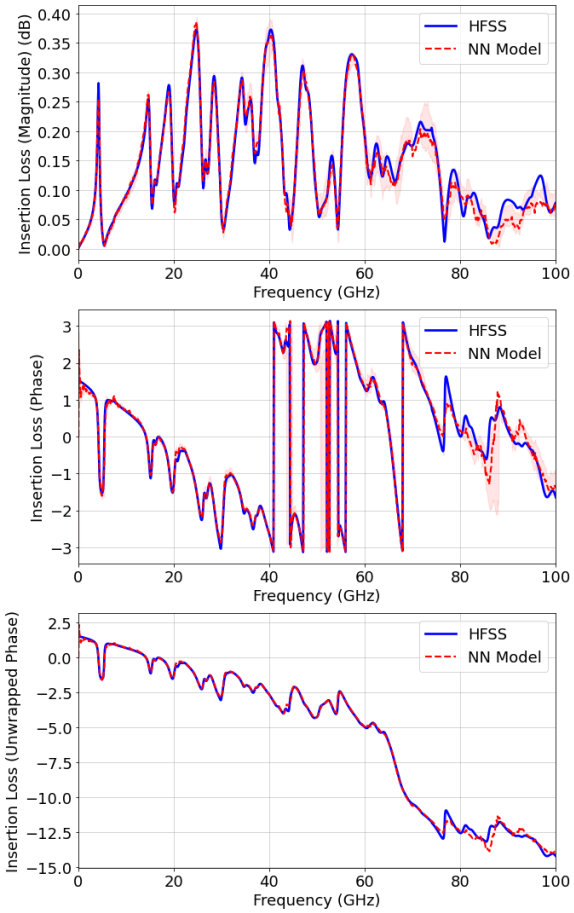
## 3.4 SI analysis of Plated-Through Hole (PTH) with fast ML model

The S-parameters predicted by the S-TCNN model are compared with the S-parameters generated by HFSS and their normalized mean squared error (NMSE) over each frequency response is calculated for the validation samples. The real-imaginary and magnitude-phase plots of the insertion loss of one of the validation samples are shown in Figure 3.8. The NMSE error is 8%. Each HFSS simulation took about 40 minutes to generate the S-parameters. The training of the S-TCNN model took about 250 seconds.

Once the learned model is generated after the training and validation, it took the inference model about 0.2 seconds to generate the S-parameters for a set of 20 input parameters that were not included in the dataset. Figure 3.9 shows the S-parameters generated by the inference model for the input parameter values (45, 120, 120, 600, 600, 120, 150, 300, 600, 35, 30, 3.5660, 4.4950, 10, 10, 10, 10, 10, 10, 10)

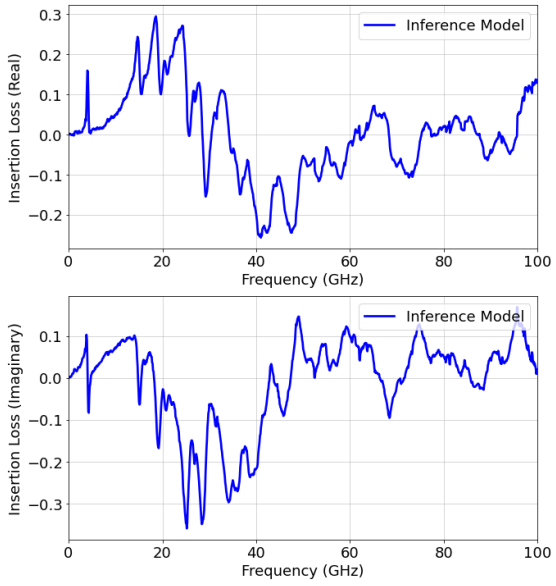



(a)

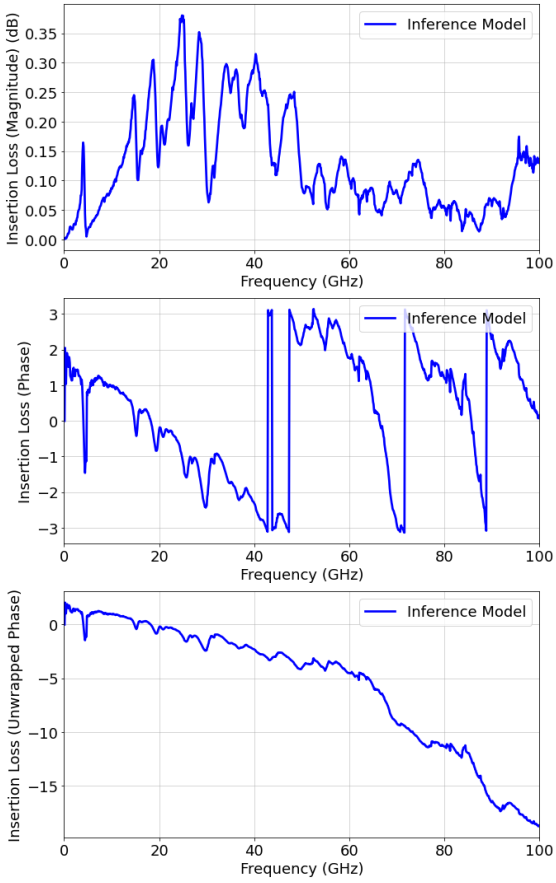



(b)

Figure 3.8: S-parameters comparison between HFSS and S-TCNN Model a) Real-Imaginary plots, b) Magnitude-Phase-Unwrapped phase plots.




(a)




(b)

Figure 3.9: S-parameters generated from the Inference Model a) Real-Imaginary plots, b) Magnitude-Phase-Unwrapped phase plots.

## 3.5 Summary

The design process of packages has been following the same methods for quite some time now. With the need for higher performance packages, it is essential to optimize the design flow of these packages to help with the fast-growing industry needs. Machine Learning techniques have proven to be a good tool to help revamp the conventional methods. By adapting these ML techniques, we can save a considerable amount of time during the design process. In this work, we discussed one such usage of ML techniques in the form of S-TCNN architecture. For a given input design space, the ML model can predict accurate S-parameters in much less time when compared to HFSS. This means that the designer does not have to run hour-long simulations every time a certain parameter is changed.

The method discussed in the work can be used to predict the S-parameters if the set of input parameters were part of the design space during the training process. If a new parameter is introduced, the ML model needs to be re-trained to get a new learned model. The accuracy of the generated learned ML model depends largely on the quality of the dataset used. The dataset generation is limited to the complexity and large simulation times of the EM solvers. This makes it very challenging to develop a learned ML model that can cover a large design space. For this reason, in the future, the dataset can be selected by determining the most significant samples within the design space. This way, the training set will have enough information to accurately describe the entire design space with the selectively chosen dataset samples.

The method used here shows a 200x improvement over HFSS while simulating a single model using the Inference model of the Neural Network. This shows that this ML tool can be a great asset to any SI/PI engineer to design the most optimal package. More adaptations of this technique can help with the optimization of other processes as well.

# CHAPTER 4

# ELECTRO-THERMAL CO-DESIGN FOR PACKAGES USING NN

## 4.1 Introduction

The increasing complexity of semiconductor devices necessitates accurate electro-thermal analysis to ensure reliable and efficient system performance. As power densities rise, and interconnect structures become more intricate, thermal and electrical interactions play a crucial role in determining the overall functionality of integrated circuits (ICs). Traditional analysis techniques have primarily focused on single-physics models, analyzing electrical, thermal, or mechanical aspects independently. However, high-performance systems introduce complex interactions that require multi-physics and multi-scale modeling approaches [9]. Hence, there is a need to create co-designing methods and tools that can help design complex systems.

To ensure effective thermal management, co-simulation approaches are necessary, integrating electrical and thermal models. Although modeling tools are available for predicting electrothermal phenomena at a component level (e.g. hot-spots), there is currently no capability to predict these interactions within a co-design optimization environment where different design teams (chip, package, board) collaborate with design/model data that can be shared to support effective trade-off analysis and optimization for a whole system. New modeling and simulation tools must accurately predict the electro-thermal coupling between multiple semiconductor components and the package/system that contains them. Modern electronic systems require tools capable of accurately predicting temperature distributions at various hierarchical levels (chip, package, and

system), simulating transient and steady-state electro-thermal behavior to detect hotspots and optimize power delivery networks (PDNs) and enabling design-space exploration through machine learning and deep-data analytics [9][10][11]. Such tools allow engineers to balance power consumption, thermal performance, and electrical reliability, ensuring optimal design trade-offs.

Excessive heating in semiconductor devices leads to increased resistance, voltage drops, and performance degradation. Electro-thermal analysis helps in preventing performance degradation and failures caused by these effects. Electro-thermal co-design integrates these analyses into a unified workflow. This enables simultaneous optimization of power integrity (PI) and thermal dissipation, helps detect thermal bottlenecks at an early stage and minimizes electrical variations due to temperature effects, thereby making the systems more reliable.

Several methodologies have been proposed for integrated electro-thermal co-design in the literature, including Physics-Based Reduced Order Models where simplified, computationally efficient models enable rapid co-analysis of electro-thermal interactions in complex systems [12] and Two-Way Coupling Simulations, where Electrical and thermal models exchange real-time data iteratively to update material properties dynamically (e.g., resistivity change due to heating) [13]. Though these methods are advantageous for performing electro-thermal analysis, at present, engineers are restricted by the large computational time of the traditional Multiphysics solvers and Electromagnetic (EM) solvers to understand the coupling between the electrothermal behavior while making design choices. Design closure can only be attained after multiple iterations through time-consuming simulation models. This reduces the ability to explore entire design spaces for optimizing geometry, materials and other parameters that comprise the board layout.

Machine learning (ML) has been proposed as a method (compact model) to co-design electro-thermal behavior in a system. To characterize the electrical behavior in a system, the frequency

response of the components within the system is analyzed. This frequency response is best described using S-parameters, which are generally obtained using EM solvers like HFSS, which, as discussed, can be time-consuming. In addition, to characterize the thermal behavior in a system, the temperature map is obtained using multi-physics simulators like Icepak for solving the energy equation. To obtain electro-thermal behavior, a two-way coupling is set up to analyze both the frequency response and the temperature effects simultaneously. This process can be very tedious and can limit the designer from developing an optimal design for both electrical and thermal effects. Recent literature has shown that the traditional solvers can be replaced by a machine learning-based compact model that is fast and accurate. [5][14] The machine learning (ML) model can predict the S-parameters as well as the temperature distribution for a particular design space once it is trained with the data in the design space. This trained ML model generates the predicted S-parameters and temperature accurately within the specified range and much faster than the traditional solvers and, therefore can replace the solvers during the design process.

In this chapter, we first discuss the two-way electro thermal coupling process. This process is used to generate datasets for training the ML model. A Plated-Through Hole (PTH) model created in Ansys for a two-way coupling analysis is shown. The ML architecture used to create a co-design model is discussed. The ML model results are compared with the generated data to test the accuracy of the model.

## 4.2 Two-Way Electro-Thermal Coupling

Two-way coupling electro-thermal simulations provide an integrated approach for analyzing the interdependent electrical and thermal behaviors in semiconductor devices. This method captures the dynamic interactions between electrical and thermal properties, allowing for

real-time updates to material parameters such as temperature-dependent resistivity, thereby improving the predictability and reliability of system performance [9].

Two-way coupling between Ansys HFSS and Ansys Icepak helps perform electro-thermal co-simulation by iterating between electrical and thermal analyses. The process involves transferring power loss data from HFSS to Icepak and updating temperature-dependent electrical properties back to HFSS. Figure 4.1 shows the steps involved in the two-way coupling process. Using this process, a Plated-Though Hole (PTH) structure is built for Electro-Thermal analysis.

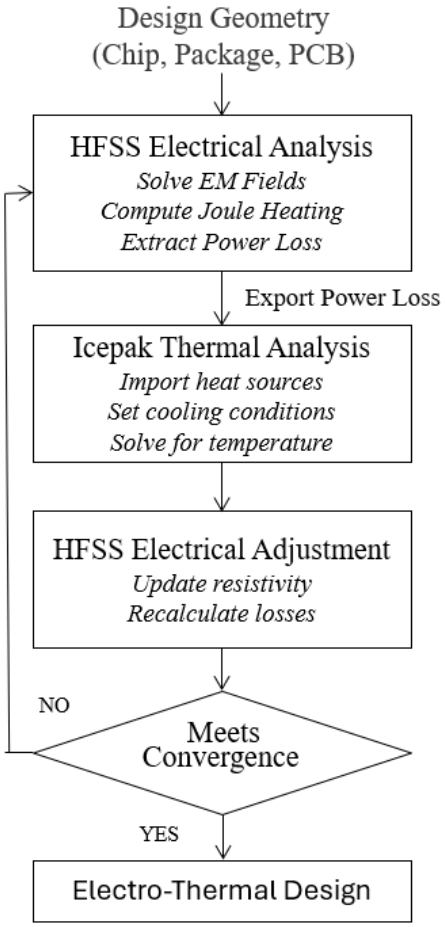



Figure 4.1: Flowchart of two-way coupling process using Ansys HFSS and Icepak

### 4.2.1 Electrical Analysis using HFSS

First, the PTH model is created in HFSS as shown in Figure 4.2. While building the model, all the geometric parameters such as the core thickness, via radius, dielectric thickness, etc., are defined as variables. This helps in setting up a parametric sweep through these geometric parameters. While assigning the materials, the material properties are entered as desired.

As the model is a 4-port structure, excitations are defined at each of the 4 ports and the excitation voltage of 1 Volt is set at ports 1 and 3. After the boundary conditions are defined, a solution setup for a frequency sweep from 0.1GHz-100GHz with 1000 steps is defined.

To perform thermal analysis, the power losses from Electromagnetic (EM) simulation that contribute to heating need to be determined. HFSS computes the EM losses within the model by computing the Surface Loss Density and Volume Loss Density. Surface loss density indicates the losses due to surface currents in conductive materials and volume loss density indicated the losses within the dielectric materials. Upon completing the EM simulation, HFSS generated a loss distribution dataset which is exported to Icepak for thermal analysis.

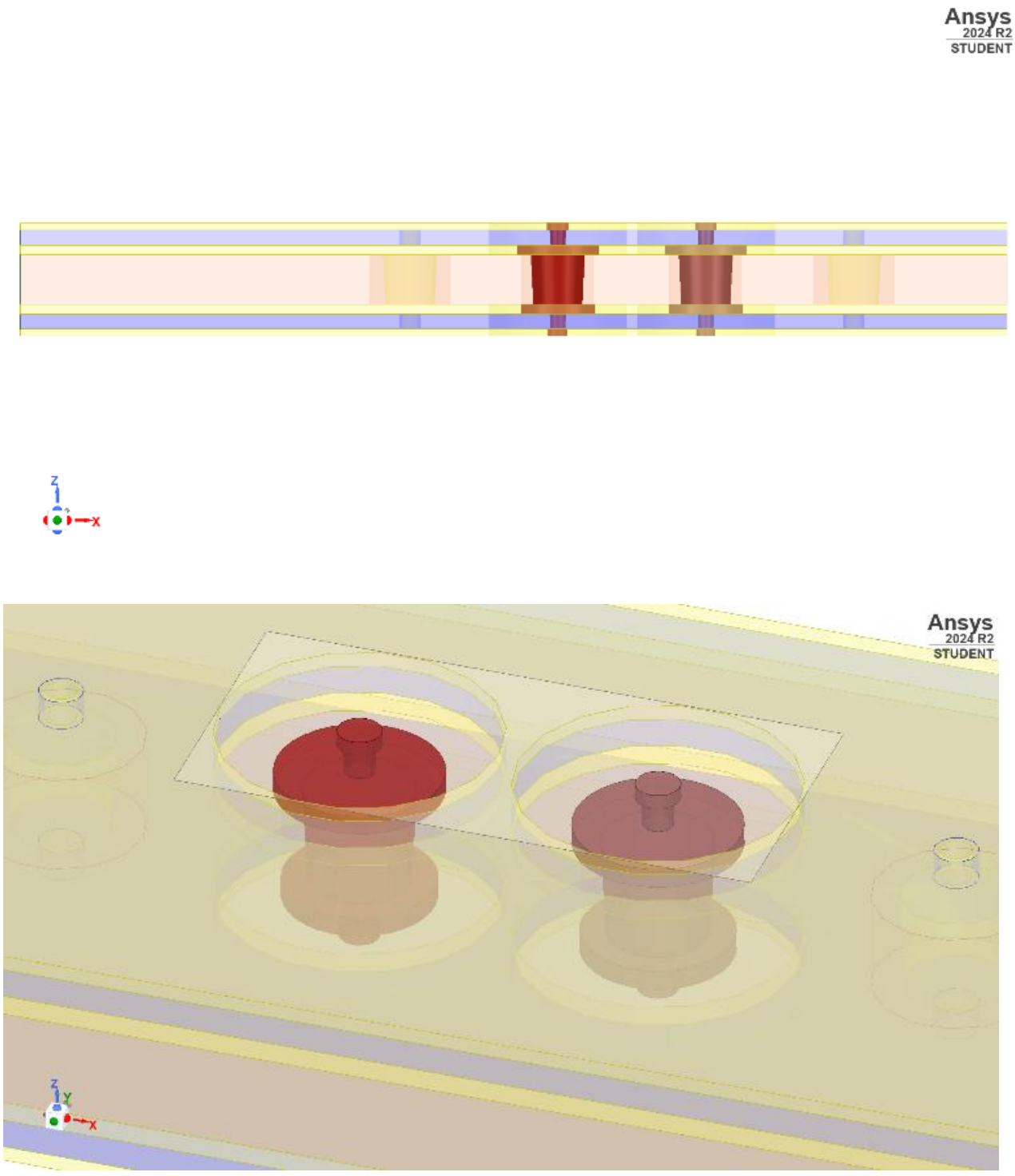



Figure 4.2: Plated Through-Hole model designed in HFSS

### 4.2.2 Thermal Analysis using Icepak

After obtaining the EM loss data, thermal analysis is performed in Icepak to compute temperature distributions. To understand the thermal effects of the materials, before setting up the Icepak model, thermal modifiers are defined for each of the materials in the HFSS model. For the dielectric, FR4_epoxy is used. The material properties for FR4_epoxy are set according to [15]. Copper is used in the model and its bulk conductivity is set according to its temperature coefficient, which is 0.00393. Thermal properties of the metal like the Specific Heat, Mass Density and Thermal Conductivity are varied within a range of values shown in Table.1, while performing parametric sweep simulation. Once the material properties are assigned, the HFSS design is imported into Icepak.

Ansys ACT extension EMtoIcepak is used to transfer the HFSS EM loss data into Icepak. Two-way coupling is enabled to iterate the thermal data into HFSS until convergence is achieved. Thermal boundary conditions, including convection, radiation and fixed temperature constraints are applied. Once the model is ready, the temperature distributions are computed based on the imported EM losses [13].

The iterative nature of the two-way coupling ensures that temperature-dependent material properties are updated for accurate analysis. Once the thermal simulation is completed, the temperature distribution is fed back into HFSS. Temperature-dependent properties such as permittivity and conductivity are updated. HFSS is re-simulated with the updated material properties. The process is repeated until convergence is reached, i.e., the temperature and EM field variations stabilize. Figure 4.3 shows the temperature distribution of the PTH model.

Though the two-way coupling process facilitates Electro-thermal co-simulation, there is high computational complexity due to iterative feedback loops between electrical and thermal solvers,

leading to increased simulation time and memory usage. Additionally, mesh resolution mismatched between electrical and thermal models can cause interpolation errors, reducing accuracy. Due to these challenges, there is a need to develop more effective ways to perform electro-thermal co-simulation. ML based models can help solve these problems.

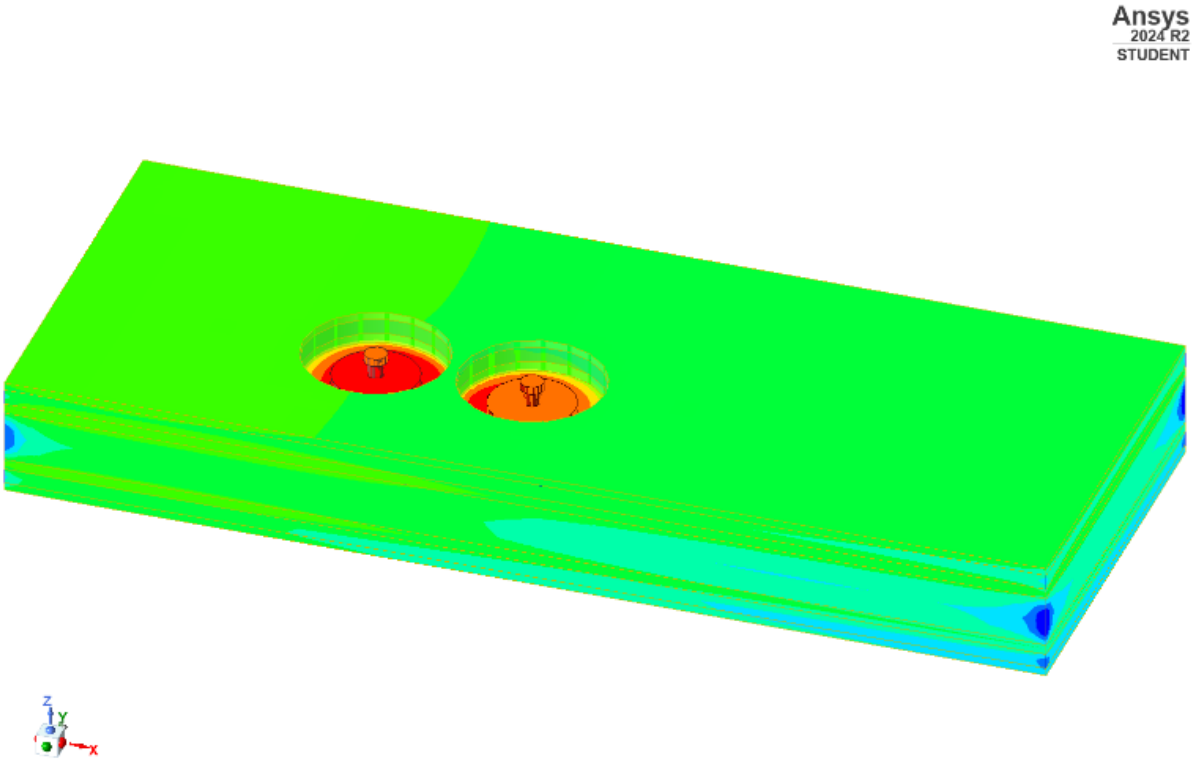



Figure 4.3: Temperature distribution of the PTH using Icepak

## 4.3 Training dataset preparation for Electro-Thermal Analysis

To model the Electro-thermal behavior of a structure using Machine Learning architectures, we need to generate a dataset to train the Neural Networks within the model. Developing a robust ML model requires data including a large design space with various material

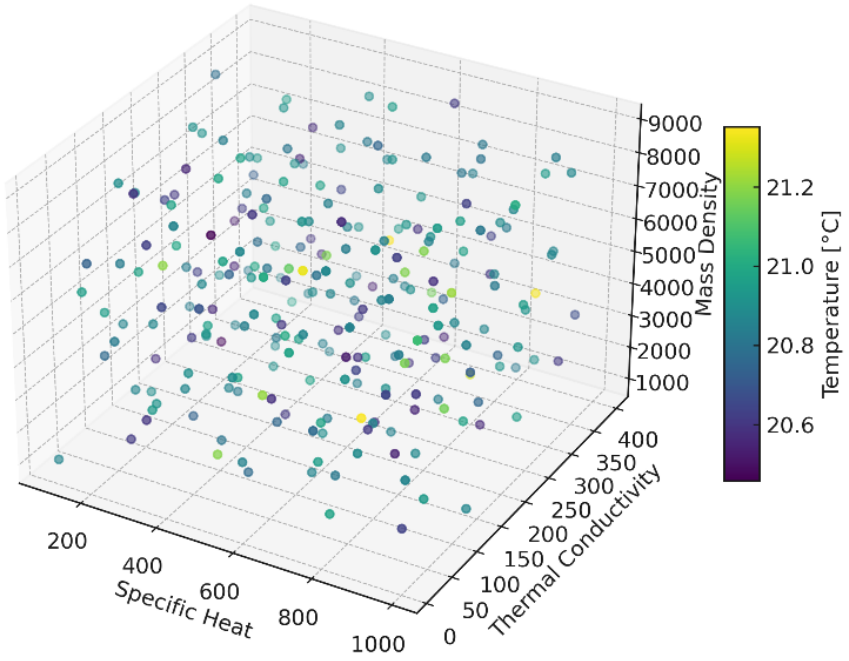
and geometry parameters. This is achieved by performing a parametric sweep simulation on the two-way coupling setup in Ansys.

As discussed earlier, one of the challenges of the two-way coupling using Ansys, is the large computational time to perform the analysis. This makes it difficult to generate huge sets of data for the training process. As the accuracy of the ML model depends largely on the quality of the dataset, we need to ensure good coverage of the entire design space without underfit issues. Recent literature [14] has shown that generating a uniform distribution of data across the design space using a Quasi-random generator, can ensure good coverage across the entire design space. Sobol sequences, a type of quasi-random sequence, are often used for high dimensional sampling. These sequences are designed to cover a space more uniformly than random sampling, reducing variance and improving convergence rates in simulations.

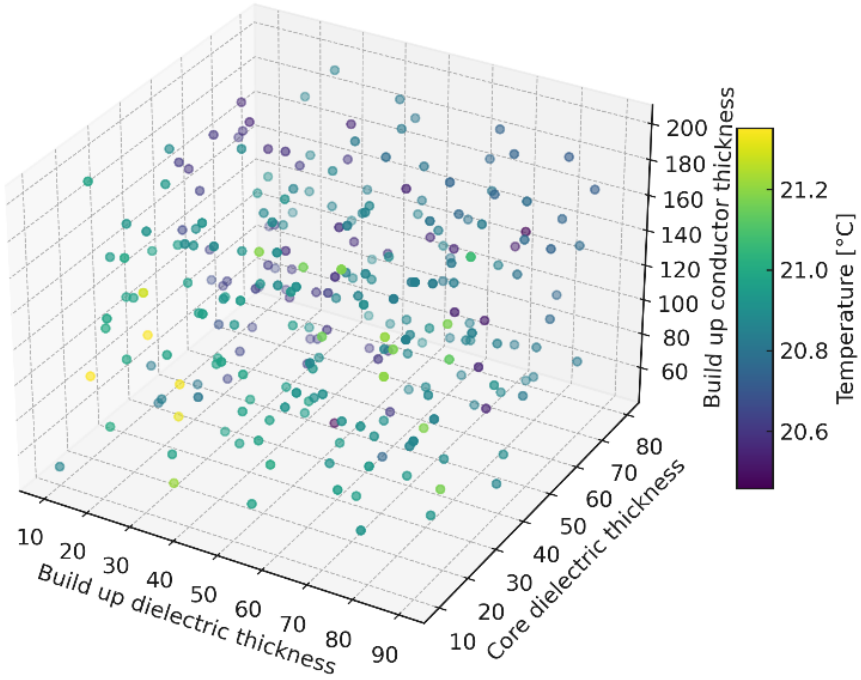
For the PTH model, three thermal parameters and four geometry parameters are varied between a certain range of values shown in Table 4.1. These parameters make the design space of the ML model. Within this design space, 256 Sobol sequences are generated to ensure uniform coverage as shown in Figure 4.4. Parametric sweep simulation is performed with these samples, first in HFSS to extract the electrical analysis data. As we use S-parameters to analyze the electrical performance of the PTH, they are extracted after the HFSS analysis is completed. Then, parametric sweep with the same samples is performed in Icepak to extract the temperature. Figure 4.4 also shows the variation of temperature across different sets of input parameters. With these two-way coupling Ansys simulations, a complete dataset has been generated, which is used to build the ML model.

TABLE 4.1. VARIABLE PARAMETERS OF THE PTH FOR ELECTRO-THERMAL ANALYSIS

| | ***Parameter*** | ***Unit*** | ***Min*** | ***Max*** |
|---|---|---|---|---|
| **Thermal** | Specific Heat | J/Kg K | 100 | 1000 |
| | Mass Density | $Kg/m^3$ | 1000 | 8900 |
| | Thermal Conductivity | W/m K | 10 | 400 |
| **Geometry** | Build up Dielectric Thickness | µm | 10 | 90 |
| | Build up Conductor Thickness | µm | 10 | 80 |
| | Core Dielectric Thickness | µm | 50 | 200 |
| | Core Conductor Thickness | µm | 10 | 40 |




(a)




(b)

Figure 4.4: Temperature of the PTH for various sets of design parameters (a) Thermal parameters (b) Geometry parameters

## 4.4 Electro-Thermal analysis of PTH using fast ML model

Machine learning can significantly enhance electro-thermal co-simulation by addressing computational inefficiencies, improving predictive accuracy, and enabling real-time modeling of complex thermal-electrical interactions. Traditional simulation methods often require solving complex differential equations, which can be computationally expensive and time-consuming. ML can help by learning from precomputed simulation data and experimental results, allowing for fast approximations of thermal and electrical behaviors without repeatedly solving physics-based models [10].

### 4.4.1 Neural Network Architecture for Electro-Thermal Analysis

The ML model being developed for Electro-thermal analysis, takes material and geometry parameters and predicts the S-parameters and the temperature of the system. Spectral Transposed Convolutional Neural Networks (STCNN), which has been discussed in Chapter 2, have shown to effectively predict the S-parameters from the geometry information. Utilizing the correlation information along the frequency axis of the frequency data, Convolutional Neural Networks (CNN) can find the patterns within the data, thereby able to predict the S-parameters accurately. As the electrical characteristics also have a dependency on the temperature, this NN architecture can also predict the thermal characteristics like the temperature, when provided with the accurate training information.

The architecture used for developing an ML model for Electro-thermal co-simulation is shown in Figure 4.5. The geometry and materials parameters are given as the input to a series of fully connected layers. The fully connected layers convert the input parameters into a latent space representation where the most correlated data is placed closer. The latent space is then flattened before being passed through a set of 1-D transposed convolutional neural networks. S-parameters

along with the temperature are generated at the output of the convolutional layers. The NN is trained to reduce the error between the training  and the predicted S-parameters and temperature. After the training process, the predicted temperature data is extracted from the architecture.

At this stage, the NN is able to predict both the S-parameter and temperature data. However, we must ensure that the predicted S-parameters are physically consistent before the final prediction. This is done by passing the frequency response output from the convolutional layers to CEL and PEL as discussed in Chapter 2. The entire NN is now trained to minimize the error between the S-parameters training and prediction data. At the end of the training process, physically consistent S-parameters are predicted by the NN. The resulting trained NN is the ML model that can be used for Electro-Thermal analysis.

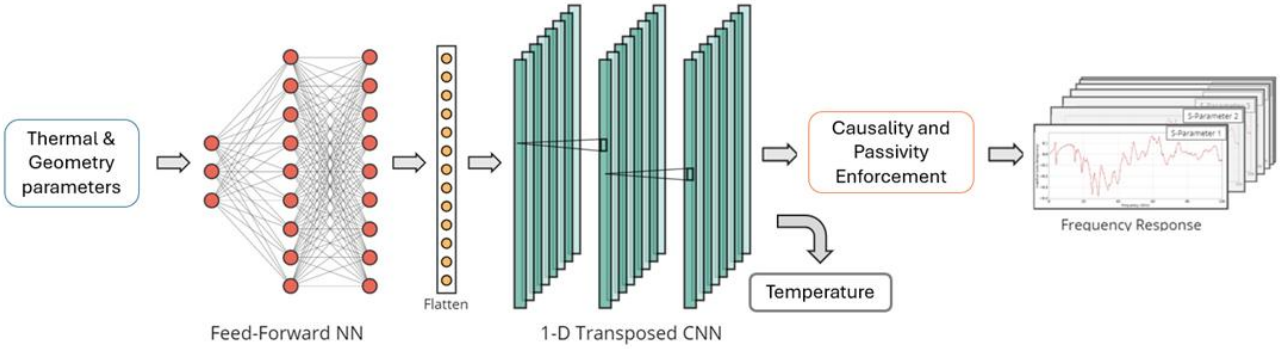



Figure 4.5: Neural Network Architecture for Electro-Thermal Analysis

### 4.4.2 Electro-Thermal Analysis results

The dataset generated from the two-way coupling process consists of 256 samples. Each sample contains 7 input geometry and material parameters along with 4-port S-parameter and temperature data at the output. Out of the 256 samples, 253 were used to train the NN and 3 samples were used to validate the ML model. The validation is done by calculating the normalized root-

mean square error (NMSE) for the S-parameters and the root-mean square error for the temperature.

The real-imaginary and magnitude-phase plots of the insertion loss predicted from the ML model are shown in Figure 4.6. The comparison of the predicted temperatures verses the temperature from the simulation data are shown in Table 4.2 along with the corresponding RMSE error. Though the accuracy for two of the samples is good, there is scope for more accuracy in the temperature predictions. The overall NMSE error for both the S-parameter and temperature prediction is 11%.

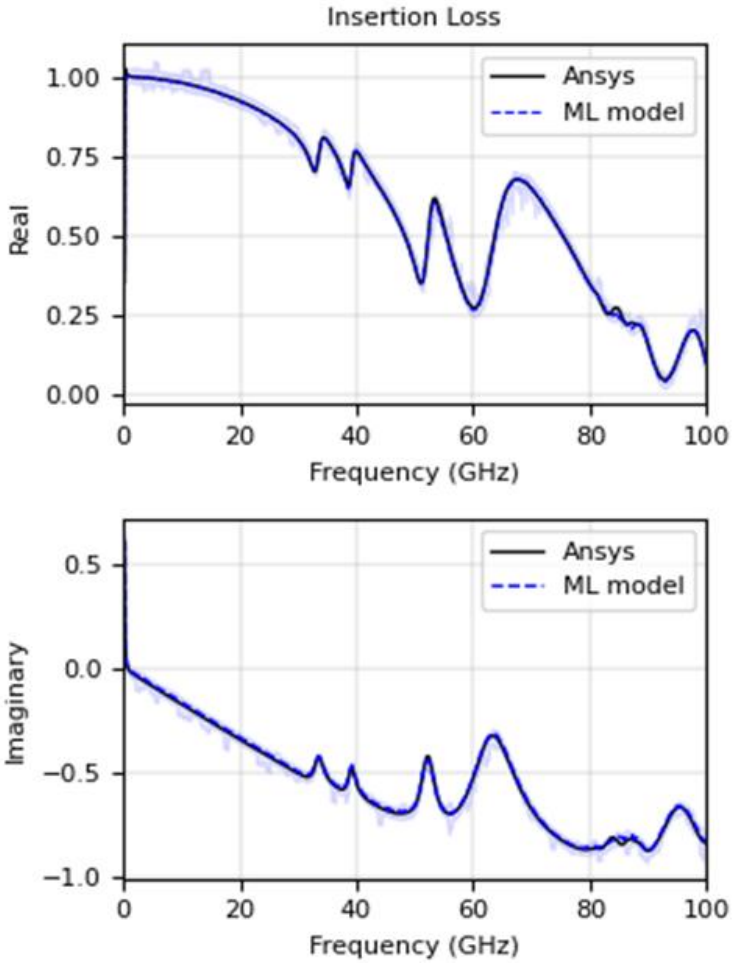
TABLE 4.2 TEMPERATURE COMPARISON BETWEEN ANSYS AND ML MODEL

| S. No | ***Simulated Temperature (°C)*** | ***Predicted Temperature (°C)*** | ***RMSE*** |
|---|---|---|---|
| 1 | 20.8723 | 20.6627 | 0.2096 |
| 2 | 20.8628 | 20.8157 | 0.0471 |
| 3 | 20.8648 | 20.838 | 0.0268 |

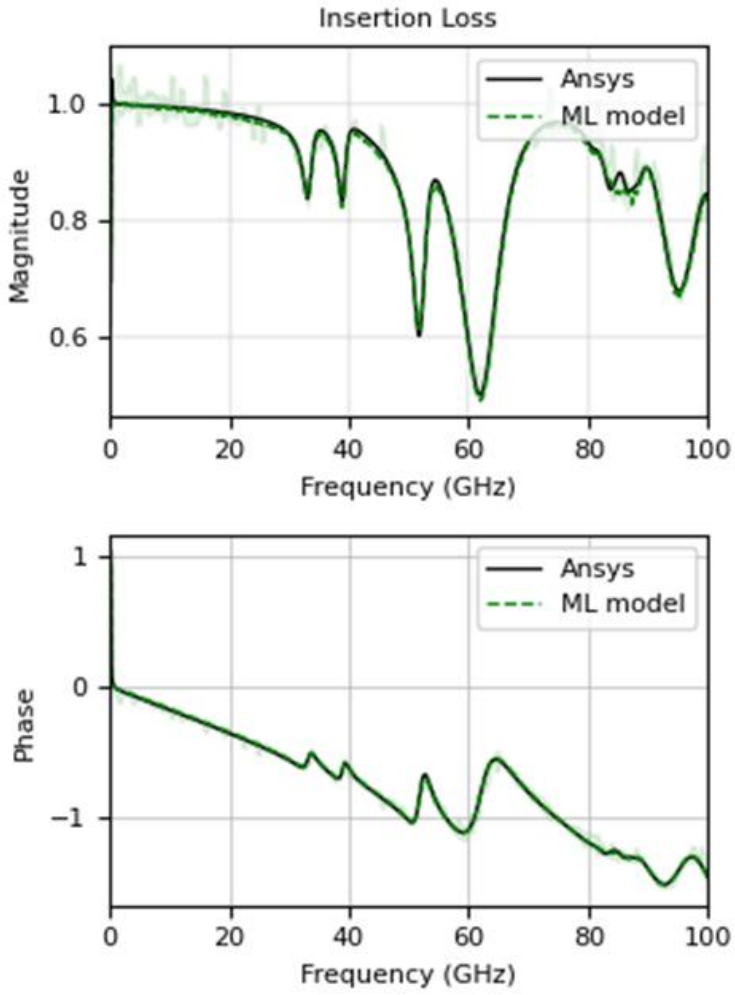
TABLE 4.3 RUN TIME COMPARISON BETWEEN ANSYS AND ML MODEL

| | ***Time for 256 simulations (seconds)*** | ***Time for 1 simulation (seconds)*** |
|---|---|---|
| HFSS | 313*60 | 73 |
| Icepak | 445*60 | 104 |
| ML model | 1700 | 0.804 |
| Speedup → 220X | | |

One of the significant advantages of creating an ML model for electro-thermal co-simulation over the two-way coupling simulations in Ansys, is the significant reduction in the computational time. For any set of design parameters within the design space used for training, the ML model can predict the S-parameter and temperature data within a fraction of seconds. The run times of Ansys simulations and ML model predictions are summarized in Table 4.3. The ML model showed a speedup of 220X in terms of run-time when compared to Ansys simulations.




(a)




(b)

Figure 4.6: S-parameters comparison between Ansys HFSS and ML model a) Real-Imaginary plots, b) Magnitude-Phase plots

## 4.5 Summary

The demand for advanced electro-thermal modeling tools is growing in response to the increasing complexity of semiconductor devices. Multi-physics co-simulation techniques, coupled with machine learning-driven optimization, provide a promising path for ensuring high-performance, reliable, and efficient systems.

The ML model built in this paper shows that electro-thermal co-simulation of integrated systems can be significantly faster by using a machine learning framework. The co-simulation scope can be increased to analyze more thermal parameters. Multi-objective optimization can be performed, that can help resolve conflicts between thermal properties like the temperature verses signal integrity effects while designing a system. The accuracy of the temperature prediction can be improved by developing a more comprehensive ML framework. AI-based predictive models like the one discussed in this paper are crucial for addressing the modern-day design challenges.

# CHAPTER 5

# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

As high-speed digital systems continue to scale in performance and complexity, accurate and efficient signal integrity (SI) analysis has become critical in modern electronic design. Traditional full-wave electromagnetic (EM) solvers and circuit-level simulators, while accurate, are computationally expensive and slow for iterative design tasks. In this context, Spectral Transposed Convolutional Neural Networks (STCNNs) offer a powerful alternative by leveraging deep learning to learn frequency-domain behaviors such as S-parameters directly from data. Below are the key advantages of using STCNNs in SI analysis.

1. High-speed inference

One of the most significant advantages of STCNNs is their ability to produce instantaneous predictions once trained. Traditional EM solvers may take several minutes to hours to compute S-parameters for complex interconnect structures, especially over a wide frequency range. In contrast, STCNNs can generate the full spectral response in milliseconds using a single forward pass through the network. This computational speed makes STCNNs ideal for design space exploration, optimization loops, what-if analysis and real-time signal integrity checks during PCB layout.

2. Learning non-linear relationships

S-parameter behavior is inherently nonlinear with respect to interconnect geometry, material properties, and signal frequency. STCNNs, built on deep neural networks, are well-suited to learn such complex, nonlinear mappings. The nonlinearity enables STCNNs to capture subtle effects like skin effects and dielectric losses at high frequencies, discontinuity-induced reflections.

3. Data-driven surrogate modeling

STCNNs act as surrogate models trained on data generated by high-fidelity simulators (e.g., HFSS, CST, ADS). Once trained, they eliminate the need to rerun expensive simulations for new geometries that fall within the training distribution. This makes STCNNs highly valuable in parametric sweeps, sensitivity analysis and yield optimization in manufacturing tolerances. By replacing slow simulations with fast inference, design cycles can be accelerated significantly without sacrificing much accuracy

4. Integration with optimization and co-simulation

Once trained, STCNNs can be easily embedded into circuit-level co-simulation flows, automatic PCB layout verification and topology optimization algorithms. This allows for closed-loop optimization where the STCNN model serves as a surrogate solver, significantly reducing computation time in tasks such as trace width tuning for impedance control, stub length optimization for via transitions and crosstalk minimization in dense routing scenarios

5. Adaptability to complex design spaces

STCNNs can be trained to model a wide variety of interconnect structures, including microstrip and stripline transmission lines, vias, coupled differential pairs and multi-layer PCB structures. This adaptability makes them a flexible modeling tool that can generalize across multiple use cases with the right data and architectural tuning. Their modular design allows for easy integration with other ML models or EDA workflows.

## 5.2 Limitations and Future work

While Spectral Transposed Convolutional Neural Networks (STCNNs) offer significant advantages in accelerating S-parameter prediction for high-speed interconnects, they are not without limitations. These limitations arise from both the neural network architecture itself and the complexity of the physical domain in which they are applied. Understanding these challenges is critical for deploying STCNNs responsibly and for motivating further improvements in model architecture and training methodology.

1. Generalization of unseen designs

One of the primary limitations of STCNNs is their limited generalization capability outside the distribution of training data. These models are inherently data-driven, and their accuracy depends heavily on the diversity and quality of the training dataset. If the model is exposed to interconnect designs or frequency ranges not well represented during training, it may produce inaccurate or non-physical predictions. Unlike physics-based simulators that can extrapolate based on Maxwell's equations, neural networks operate based on pattern recognition and interpolation.

This issue is further exacerbated by high-dimensional input spaces, such as parametric variations in trace length, width, dielectric constants, and layer stack-ups. Covering the entire design space with simulated training data is computationally expensive, which limits the practicality of using STCNNs as general-purpose solvers

2. Training complexity and computational cost

Although inference using STCNNs is extremely fast, training the model can be computationally expensive. The need for large, high-fidelity simulation datasets and iterative training processes—

sometimes involving hundreds of thousands of parameters—requires significant GPU/TPU resources and tuning expertise.

Moreover, the design of the network architecture (e.g., number of layers, kernel sizes, upsampling strategies) significantly affects performance, and there is no universally optimal configuration. Thus, model development may involve trial-and-error, cross-validation, and hyperparameter tuning, which can delay deployment.

3. Interpretability and physical insight

STCNNs operate as black-box models and lack interpretability compared to traditional analytical or simulation-based approaches. Designers and SI engineers often need not just predictions but insights into the physical causes of signal degradation, such as reflections, coupling, or dielectric losses. STCNNs do not provide any interpretable internal representation of physical phenomena like wave propagation or impedance mismatch, making them less useful in root-cause analysis or debugging

Despite their speed and learning capacity, STCNNs have important limitations that must be addressed before they can fully replace traditional electromagnetic solvers in SI workflows. As discussed above, these include generalization limitations, potential physical law violations, training complexity, and lack of interpretability. Addressing these challenges involves augmenting STCNNs with physics-aware constraints, hybrid modeling approaches, and domain-specific architecture tuning.

## 5.3 Future work

### 5.3.1 Time -domain analysis using ML model

The aim is to develop an efficient way for design and analysis of a channel. We have seen that machine learning can be used to predict the S-parameters for any given geometry or material information. The next most natural step is to use the S-parameters obtained for time-domain analysis using simulators like SPICE, LIM, etc. Feed-forward neural networks, recurrent neural networks, Long Short-term memory (LSTMs), CNNs, can be used to learn the sequential data in the time domain, where temporal relationships are key. With good datasets and proper tuning, an ML model can be obtained to perform time-domain analysis.

### 5.3.2 Multi-objective optimization and material property analysis

As it has been demonstrated that Electro-thermal analysis can be performed using ML methods, this can enable holistic optimization that traditional methods struggle to achieve. Multi-objective optimization can be performed, that can help resolve conflicts between thermal properties like the temperature verses signal integrity effects while designing a system. This can also provide insights into the materials that should be used to reduce SI effects. The accuracy of the temperature prediction can be improved by developing a more comprehensive ML framework.

# REFERENCES


[1] R. R. Tummala and Madhavan Swaminathan, *Introduction to system-on-package (SOP) : miniaturization of the entire system*. New York: Mcgraw-Hill, 2008.

[2] M. Swaminathan and A. Engin, *Power Integrity Modeling and Design for Semiconductors and Systems*, Prentice Hall, 2007.

[3] D. M. Pozar, *Microwave Engineering*, 4th ed., Wiley, 2011.

[4] A. Deutsch et al., "High-Speed Signal Propagation on Lossy Transmission Lines," IBM J. Res. Dev., vol. 34, no. 4, pp. 601–615, Jul. 1990.

[5] M. Swaminathan, H.M. Torun, H. Yu, J. A. Hejase, and W. D. Becker, "Demystifying Machine Learning for Signal and Power Integrity Problems in Packaging," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 10, no. 8, pp. 1276-1295, Aug. 2020.

[6] H. M. Torun, A. C. Durgun, K. Aygun, and M. Swaminathan, "Causal and passive parameterization of s-parameters using neural networks," IEEE Trans. Microw. Theory Techn., 2020, doi: 10.1109/TMTT.2020.3011449.

[7] Bengio, Y.; Goodfellow, I.; Courville, A., *Deep Learning*; MIT Press: Massachusetts, 2017.

[8] H. M. Torun, A. C. Durgun, K. Aygun, and M. Swaminathan, "Enforcing causality and passivity of neural network models of broadband S-parameters," in Proc. IEEE 28th Conf. Electr. Perform. Electron. Packag. Syst. (EPEPS), Oct. 2019, pp. 1–3.

[9] "Heterogeneous Integration Roadmap - IEEE Electronics Packaging Society," *eps.ieee.org*. https://eps.ieee.org/technology/heterogeneous-integration-roadmap.html

[10] K. Zhang et al., "Machine learning-based temperature prediction for runtime thermal management across system components," IEEE Trans. Parallel Distrib. Syst., vol. 29, no. 2, pp. 405-419, Feb. 2018.

[11] X. Cui et al., "A fully coupled multi-physics model for electromigration failure analysis," IEEE Trans. Device Mater. Reliab., 2019.

[12] S. Sukharev et al., "Physics-based electro-thermal modeling of interconnect reliability," Microelectronics Reliability, vol. 72, pp. 1-11, 2017.

[13] “Two-Way Coupling — Lesson 6 | ANSYS Innovation Courses,” ANSYS Innovation Courses | Free, online, physics and engineering courses, Dec. 19, 2022. https://innovationspace.ansys.com/courses/courses/etm-using-ansys-hfss-and-icepak/lessons/two-way-coupling-lesson-6-2/ (accessed Nov. 05, 2024).

[14] Juhitha Konduru, Oleg Mikulchenko, L. Y. Foo, and J. E. Schutt-Ainé, “Signal Integrity Analysis and Design Optimization using Neural Networks,” pp. 924–928, May 2024, doi: https://doi.org/10.1109/ectc51529.2024.00150.

[15] M. Hernandez -Force10 and Networks, “A Definition of FR-4 Joel Goergen -Force10 Networks.” Accessed: Nov. 04, 2024. [Online]. Available: https://ieee802.org/3/ap/public/may04/goergen_01_0504.pdf