EYE DIAGRAM MODELING OF HIGH-SPEED CHANNELS USING ARTIFICIAL NEURAL NETWORKS WITH AN IMPROVED ADAPTIVE SAMPLING ALGORITHM

GOAY CHAN HONG

UNIVERSITI SAINS MALAYSIA

2019

EYE DIAGRAM MODELING OF HIGH-SPEED CHANNELS USING ARTIFICIAL NEURAL NETWORKS WITH AN IMPROVED ADAPTIVE SAMPLING ALGORITHM

by

GOAY CHAN HONG

Thesis submitted in fulfilment of the requirements for the Degree of Master of Science

February 2019

ACKNOWLEDGMENT

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Patrick Goh Kuan Lye for his patience, motivation, advice. His guidance helped me in all the time of research and writing of this thesis. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. Other than that, I would like to thank my co-supervisor, Dr. Nur Syazreen Ahmad for her encouragement.

My sincere thanks to Mr. Tan Chee Sheng for his help in the field of image processing. He provided the background knowledge about image processing and suggestions for implementation. Furthermore, I must thank Dr.-Ing Volker Mühlhaus for his guidance in RF software.

Finally, I would like to thank my family and friends, especially my parents for supporting me throughout my life.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AMG	Automated Model Generation
ANN	Artificial Neural Network
BER	Bit Rate Error
CIFE	Conditional Infomax Feature Extraction
CMIM	Conditional Mutual Info Maximization
ConDred	Conditional Redundancy
DISR	Double Input Symmetrical Relevance
DoE	Design of Experiment
EM	Electromagnetic
FCBF	Fast Correlation-Based Filter
ICAP	Interaction Capping
JMI	Joint Mutual Information
LVDS	Low Voltage Differential Signaling
MESFET	Metal-Semiconductor Field-Effect Transistor
MIFS	Mutual Information Feature Selection
MIM	Mutual Information Maximization
MLP	Multilayer Perceptron
MRMR	Max-Relevance Min-Redundancy
PKI-ANN	Prior Knowledge Input ANN
PKID-ANN	Prior Knowledge Input Difference ANN
RAM	Random-Access Memory
RF	Radio Frequency
RTS	Reduced Training Set
SI	Signal Integrity
SOM	Self-Organizing Map
UI	Unit Interval

LIST OF SYMBOLS

$BEH(t)$	Binned eye heights as a function of time, t.
$B_{i,(2j-1)}$	The minimum value of the jth design parameters in R_i.
$B_{i,2j}$	The maximum value of the jth design parameters in R_i.
C	Capacitance.
$ceil(.)$	Rounding towards positive infinity.
d	Number of dimension of the modeled problem.
D	Constant term.
dH	Step size for hidden neurons searching.
e	Vector of network errors.
$EH(t)$	Eye heights as a function of time, t.
E_r	Region error of a region, r.
$f(s)$	A rational function.
$f_c(x_f, \varphi)$	Coarse model responses.
$f_f(x_f, \varphi)$	Fine model responses.
g	Gradient.
h	Proportional term.
H	Number of hidden neurons.
$H(X)$	The entropy of distribution X.
$H(X\|Y)$	The conditional entropy of X given Y.
H_e	Hessian matrix.
H_h	Upper search boundary for fine searching process.
h_i	The ith hidden neuron.
H_l	Lower search boundary for fine searching process.
H_{max}	Upper search boundary for hidden neuron searching process.
H_{min}	Lower search boundary for hidden neuron searching process.
H_{opt}	Optimal value of H.
h_{sub}	Substrate height.
H_{ub}	Upper bound of H.
I	Identity matrix.
$I(X;Y)$	Mutual information between X and Y.
$I(X;Y\|Z)$	Conditional mutual information.

J	Jacobian matrix.
$J_{cife}(X_k)$	CIFE criterion of a feature X_k.
$J_{cmim}(X_k)$	CMIM criterion of a feature X_k.
$J_{condred}(X_k)$	ConDred criterion of a feature X_k.
$J_{disr}(X_k)$	DISR criterion of a feature X_k.
$J_{icap}(X_k)$	ICAP criterion of a feature X_k.
$J_{jmi}(X_k)$	JMI criterion of a feature X_k.
$J_{mifs}(X_k)$	MIFS criterion of a feature X_k.
$J_{mim}(X_k)$	MIM criterion of a feature X_k.
$J_{mrmr}(X_k)$	MRMR criterion of a feature X_k.
k	A positive value where $H = 2^k$.
K	Number of frequency points.
k_{max}	The maximum value of k.
k_{opt}	Value of k when $H = H_{opt}$.
L	Number of layers of an ANN.
$length(.)$	Length of a vector.
L_{Tline}	Length of the transmission line.
m	Number of output neurons.
M	Number of S-parameters used as inputs.
$mse(.)$	Mean squared error.
n	Number of input neurons.
N	Number of ports of the circuit.
N_{horiz}	Total number of pixels in the horizontal axis.
N_l	Number of neurons at the lth layer of a feedforward ANN.
n_r	Weight assigned to a region, r based on its volume.
N_S	Number of symbols.
N_T	Number of time points.
N_{trials}	Number of ANNs created in each iteration of hidden neuron searching.
N_{trn}	Length of training data.
N_{trneq}	Number of training equations.
N_{vert}	Total number of pixels in the vertical axis.
N_w	Number of unknown weights of the ANN.

$p(x)$	Marginal probability density function.
$p(x\|y)$	Conditional probability density function.
p_q	The *q*th pole.
Q	Order of approximation using the vector fitting method.
R	List of regions.
R_i	The *i*th region in R.
R_{new}	The new regions created by splitting the worst performing region.
r_q	The *q*th residue.
R_{term}	Termination resistance.
S	Set of currently selected features.
S_{DoE}	Sample size for DoE.
S_{RTS}	Sample size for RTS.
S_{Tline}	Separation between two lines of a differential pair.
t	Time.
$TanD$	Loss tangent.
t_{max}	The maximum time.
t_{min}	The minimum time.
t_N	Normalized desired ANN outputs.
$t_{N,r}$	The normalized desired ANN outputs within a region, *r*.
$var(.)$	Variance of a vector.
V_{max}	The maximum voltage.
V_{min}	The minimum voltage.
w_{ij}^l	Weight assigned to the connection between the *i*th neuron at the *l*th and the *j*th neuron at (*l*-1)th layer.
WCE_i	The worst case error of the *i*th normalized output.
w_k	Weight vector in the *k*th iteration.
W_{Tline}	Width of the transmission line.
x_c	Center of a region.
x_f	Design parameters.
x_i	The *i*th external input of an ANN.
$x_{max,i}$	The maximum value of the *i*th design parameters.
$x_{min,i}$	The minimum value of the *i*th design parameters.
x_{nz}	Design parameters that do not affect the Z_{diff}.

x_{trn}	Training samples.
x_{val}	Validation samples.
x_z	Design parameters that affect the Z_{diff}.
y_{ANN1}	Outputs of ANN1.
y_{ANN2}	Outputs of ANN2.
y_{ANN3}	Outputs of ANN3.
y_D	Outputs of the difference ANN.
y_i	The *i*th ANN output.
y_N	Normalized ANN outputs.
$y_{N,r}$	Normalized ANN outputs within a region, *r*.
y_{PKI}	Outputs of the PKI-ANN.
y_{PKID}	Outputs of the PKID-ANN.
y_{sim}	Simulator outputs.
z_i^l	Output of the *i*th neuron at the *l*th layer.
Z_{diff}	Differential impedance.
β	Inter-feature dependency penalization factor.
γ	Conditional penalization factor.
γ_i^l	Weighted sum of the *i*th neuron at the *l*th layer.
ε_r	Dielectric constant.
μ	Combination coefficient.
$\sigma(.)$	Neuron activation function.
φ	Independent variables.

PERMODELAN GAMBAR RAJAH MATA BAGI SALURAN BERKELAJUAN TINGGI MENGGUNAKAN RANGKAIAN NEURAL BUATAN DENGAN ALGORITMA PERSAMPELAN SUAI YANG DIPERBAIKI

ABSTRAK

Apabila kadar data meningkai kepada julat gigabit dan seterusnya, analisis integriti isyarat (SI) menjadi semakin sukar dan lambat. Oleh itu, banyak penyelidik telah mula mencari rangkaian neural tiruan (ANN) sebagai alternatif kepada alat permodelan SI tradisional kerana ANN mudah digunakan dan cepat. Walau bagaimanapun, sejumlah besar sampel perlu dijanakan untuk proses latihan ANN untuk permodelan reka bentuk yang kompleks, dan ini mengakibatkan kos pembinaan model rangkaian neural yang tinggi. Teknik pensampelan suai digunakan untuk penjanaan data kerana fleksibilitinya di mana ia menjana sampel mengikut ketidaklinearan kawasan-kawasan di ruang reka bentuk. Kerja ini mencadangkan penambahbaikan kepada algoritma persampelan suai asal dan menggunakannya sebagai kaedah persampelan untuk pemodelan rajah mata. Ini mengurangkan bilangan sampel latihan sebanyak 16.1%, sampel pengesahan sebanyak 14.7% dan masa pembinaan rangkaian neural sebanyak 23%. Disamping itu, penggunaan rangkaian neural input pengetahuan sebelumnya (PKI-ANN) dan rangkaian neural input pengetahuan sebelumnya perbezaan (PKID-ANN) untuk permodelan masalah SI dimensi tinggi dicadangkan. Kesalahan kes terburuk yang dinormalkan untuk PKI-ANN hanya 6.66% dan untuk PKID-ANN hanya 6.32% berbanding dengan ANN konvensional yang sebanyak 11.44%. Akhir sekali, teknik rangkaian neural bagi permodelan keseluruhan kontur kadar ralat bit (BER) dicadangkan untuk memberikan lebih banyak maklumat kepada jurutera, seperti bentuk penuh mata bukannya hanya ketinggian dan lebar mata. Prestasi pengujian purata $R^2 = 0.983$ dicapai untuk teknik permodelan neural kontur BER.

EYE DIAGRAM MODELING OF HIGH-SPEED CHANNELS USING ARTIFICIAL NEURAL NETWORKS WITH AN IMPROVED ADAPTIVE SAMPLING ALGORITHM

ABSTRACT

As data rates increase to the gigabit range and beyond, signal integrity (SI) analysis becomes increasingly difficult and time consuming process. Thus, many researchers have started to look out for artificial neural networks (ANNs) as an alternative to traditional SI modeling tool because ANNs are easy to use and fast. However, large amount of samples need to be generated for the training process of the ANN for the modeling of a complex design, resulting in a high neural model development cost. The adaptive sampling technique is used for the data generation due to its flexibility where it generates samples according to the non-linearity of the regions in the design space. This work proposes an improvement to the original adaptive sampling algorithm and uses it as the sampling method for eye diagram modeling. This reduces the number of training samples by 16.1%, validation samples by 14.7% and neural model development time by 23%. Besides that, the use of the prior knowledge input neural network (PKI-ANN) and the prior knowledge input difference neural network (PKID-ANN) for the modeling of high dimensional SI problem is proposed. The normalized worst-case error for the PKI-ANN is only 6.66% and for the PKID-ANN is only 6.32% as compared to that of the conventional ANN which is 11.44%. Finally, the neural network technique for the modeling of entire bit rate error (BER) contours is proposed which provides engineers with more information, such as the full shape of eye instead rather than just the height and width of the eye. An Average testing performance of R^2=0.983 is achieved for the BER contour neural modeling technique.

CHAPTER ONE

INTRODUCTION

1.1 Background Overview

As high-speed signal rates increase to the multi-gigahertz range, signal integrity (SI) becomes a very significant factor in a circuit design. At low data rates, a simple conductor can be used to transmit signals over short distances without causing severe signal degradation issues. However, it becomes more difficult to maintain the characteristics of the transmitted signal waveform as the signal speed increases. This is because effects such as ringing, crosstalk, reflections, and ground bounce start to become significant at high data rates even for short lines. Consequently, the design engineers of high-speed circuits need to take these effects into consideration, resulting in even more complex electronic designs. In addition, the engineers will also need to consider for variations of design parameters due to manufacturing process limitations. This can also affect the electrical properties of high-speed circuits and cause further unwanted problems.

An example high-speed interconnect topology of a PCI Express 2.0 system is shown in Fig. 1.1. Usually, SI analysis involves two simulation tools, electromagnetic (EM) field solvers and circuit simulators [1]. An EM field solver is used to extract the frequency response of the high-speed interconnect structure. The circuit simulator is then used to carry out time domain simulations to obtain the corresponding output waveforms. EM field solvers are accurate, but they are also slow. Even though circuit simulators are generally fast, when the time domain responses involve very long bit sequences, it can still take up a considerable amount of time. Traditionally, engineers need to perform multiple EM and time domain simulations during the circuit design stage to obtain the desired design which is costly in terms of both computational power and time. Thus, there is an ever-increasing demand for faster and more efficient strategies for high-speed circuit modeling and analysis.

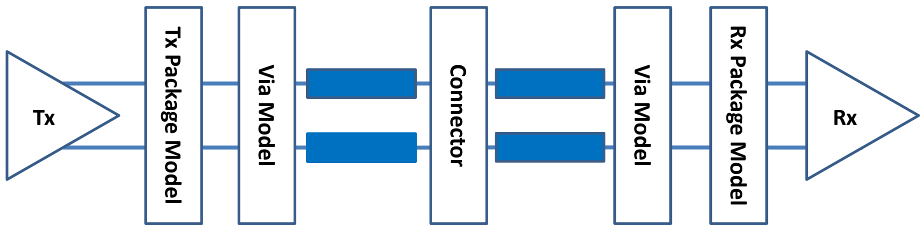
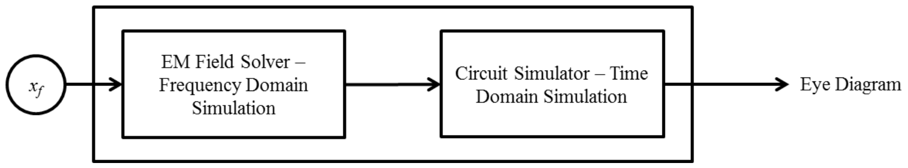


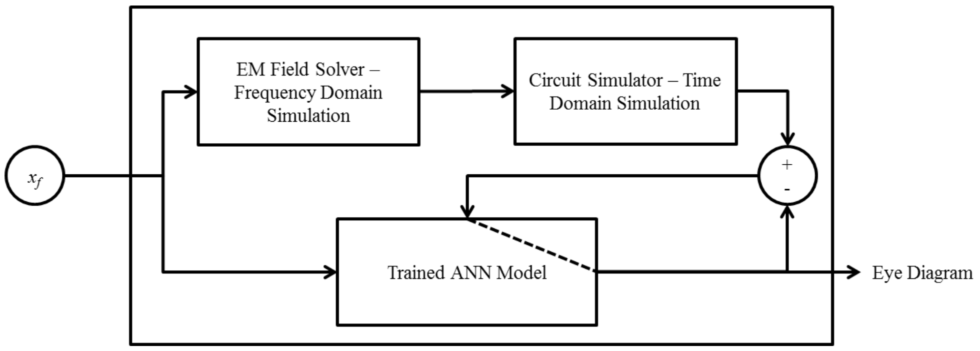
Figure 1.1: PCI Express 2.0 topology from the transmitter to the receiver.

Artificial neural networks (ANNs) have been widely applied in radio frequency (RF) and microwave circuit modeling problems [2-6]. An ANN is an information processing system, with its design inspired by the neuronal structure of mammalian brains. A neural network can learn the relationship between the design parameters and the electrical properties of electronic designs. Then, the well trained neural network can be used in the design process, thereby partially or completely replacing either the EM field solver or the circuit simulator or even both of them. This will speed up the design process because a well-trained ANN is many times faster than both the EM field solver and the circuit simulator. Fig. 1.2 shows the differences between the traditional and neural networks based technique for SI modeling where x_f are the design parameters.



x_f
EM Field Solver – Frequency Domain Simulation
Circuit Simulator – Time Domain Simulation
Eye Diagram

(a)



x_f
EM Field Solver – Frequency Domain Simulation
Circuit Simulator – Time Domain Simulation
+
-
Trained ANN Model
Eye Diagram

(b)

Figure 1.2: SI modeling using a) traditional and b) neural networks technique.

1.2 Problem Statement

Simulation tools such as 3D electromagnetic field solvers can be accurate, but slow, whereas faster models such as design equations and equivalent circuit models lack accuracy. Therefore, there is a demand for more effective modeling strategies, with a high level of accuracy and faster speed. Recently, ANNs have gained popularity in the RF and microwave circuit modeling community as a new modeling tool. The ANN can learn from the simulation data of a modeled problem, then perform prediction based on the design parameters (inputs) to electrical properties relationship (outputs) it learned. Once an ANN is trained, it is many times faster than the EM field solver. In [6], the ANN is compared to EM simulators in terms of speed. It is presented that full-wave EM simulation of a side-coupled circular waveguide dual-mode filter using a mode-matching-based EM simulator takes about 6 minutes, and the simulation takes about 45 minutes to complete when using a finite-element-based EM simulator. On the other hand, the ANN method only requires 0.006 second for each evaluation. The popularity of ANNs in RF and microwave circuit modeling has inspired the use of ANNs in the field of signal integrity. Usually, the analysis of a high-speed channel requires at least two simulations, a frequency domain, and a time domain simulation. Most of the time, the 3D EM field solver is used for the frequency domain simulation to extract the S-parameters of the design. S-parameters (also called scattering parameters) describe the input-output relationship between ports or terminals in an electrical system. For example, if there are two ports (Port 1 and Port 2) in a system, then S_{12} represents the power transferred from Port 2 to Port 1. Then, a time domain simulation is performed to obtain the output voltage waveform. Finally, the output voltage waveform is used for the construction of an eye diagram which is commonly used by the designers for signal integrity analysis. ANNs can be applied to the field of signal integrity to speed up the simulation process by either replacing one or both simulations.

Traditionally, the sampling methods used in neural circuit modeling are one-shot design approaches such as the design of experiment (DoE) and uniform grid sampling. Usually, it is very hard for the designers to know the information about the level of non-linearity across the design space before the sample generation process. Additionally, the level of non-linearity is not uniform across the entire design space most of the time. It is very easy for one-shot design approaches to cause undersampling in non-linear regions and oversampling in highly linear regions especially when a uniform grid sampling approach is used. If the sampling method fails to provide the desired exploration of the design space after the sampling process, then the designers may have to reset the grid or even repeat the sampling process. Additionally, it is very difficult to explore the whole design space manually as the design space is often too large and each simulation can take a very long time to complete. Therefore, another sampling strategy called adaptive sampling is used where it generates data points iteratively and intelligently at locations with high non-linearity to have the greatest information gain [7]. Adaptive sampling usually starts by generating a small number of samples, and then adds more samples in highly non-linear regions in each iteration based on how the neural networks perform in each region. This can prevent the generation of excessive samples and then lead to a better exploration of the design space, which then enables the construction of better neural models with fewer data points. However, the adaptive sampling strategy has a weakness where it often focuses too hard on a few small, highly non-linear regions, causing it to get stuck in these regions. This usually happens when these highly non-linear regions also contain erroneous data. Sometimes, this can also prevent the neural networks from converging. Thus, this work proposes a modification to be done on the original adaptive sampling algorithm, allowing it to escape from these regions.

1.3 Research Objectives

The objectives of this research are:

1. To apply artificial neural networks and knowledge embedded neural networks for eye diagram modeling of electrical systems for signal integrity analysis.
2. To improve the adaptive sampling algorithm and use it for neural network modeling of bit error rate contours.

1.4 Research Scope

This work focuses on the eye diagram modeling using ANNs, with the adaptive sampling as the data generation strategy. ANNs are created to map the design parameters to the SI metrics such as the eye height and eye width.

Two types of circuits are modeled, the single-ended and differential microstrip transmission lines. The single-ended microstrip transmission lines have data rate of 1 Gbps, and they are created and simulated using MATLAB RF toolbox. The differential microstrip transmission lines have low-voltage differential signaling (LVDS) standard with the data rate of 2.5 Gbps. The LVDS lines are created and simulated using Advanced Design System (ADS) if EM capability is not needed, and using SonnetLite if EM capability is required. Also, ADS is used for its channel simulator, ChannelSim to construct the eye diagrams. Multilayer perceptron is the only class of neural networks used in this work. The neural networks are constructed using the MATLAB Neural Network Toolbox and trained using Levenberg-Marquardt backpropagation algorithm.

There are several constraints in this research. Firstly, all the modeled eye diagrams are horizontally symmetrical. Secondly, no noise is fed into the system at the transmitter end. Thirdly, equalization method is not use at both the transmitter and receiver end.

1.5 Thesis Outline

The thesis begins with Chapter 1, which discusses the background, problem statement, project objectives, project scope, and thesis outline.

The background of ANNs and previous works on applications of ANNs in the field of signal integrity is reviewed in Chapter 2. Other than that, details about the adaptive sampling algorithm are discussed. Several types of sampling strategy are reviewed and compared. The knowledge embedded neural modeling technique is also explained. This technique allows designers to make use of existing knowledge in combination with the ANNs to further improve the prediction performance.

Chapter 3 discusses the important processes in neural network development, such as data generation, determination of neural network structure, training of neural network, and evaluation of neural network. In the data generation process, the workflow of frequency and time domain simulation are presented. Besides that, two types of eye diagram modeling approaches are discussed: 1) to model an eye diagram as an eye height and an eye width with one sampling point, and 2) to model an eye diagram as the whole eye contour which is proposed in this work. The improvement made to the original adaptive sampling is discussed. Finally, the feature selection technique is discussed. Feature selection is used in the second eye modeling strategy

The results in Chapter 4 are presented in several sections. Section 4.2 compares the performances of the original and improved adaptive sampling algorithm for the modeling problem of a single-ended microstrip transmission line structure. Section 4.3 compares the performances of conventional ANN, prior knowledge input ANN (PKI-ANN), and prior knowledge input difference ANN (PKID-ANN) on the modeling of a circuit with a low voltage differential signaling (LVDS) standard. Section 4.4 shows the results for the modeling of the entire eye contour using ANN.

Chapter 5 concludes the whole project with some remarks and suggestions for future work or research improvements.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

This chapter presents the review of prior works in the field of SI modeling using ANN. Section 2.2 discusses the background about ANN in terms of its structure and computation of its outputs. Section 2.3 reviews the previous works in SI neuro-modeling as well as the sampling techniques used by them. Eye diagram modeling from design parameters and S-parameters are compared in terms of their strengths and weaknesses. Section 2.4 discusses about knowledge embedded neural networks which are ANNs with incorporated with existing knowledge in their structure. Lastly, Section 2.5 discusses the background of various filter-based feature selection methods.

2.2 Artificial Neural Networks

ANN is a simplified mathematical model of a biological neural network that consists of interconnected neurons. The multilayer perceptron (MLP) is one of the most widely used artificial neural networks. Its neurons are arranged in L layers, where layer one is the input layer, layer L is the output layer, and layers two to L-1 are hidden layers. A neural network with one hidden layer is often considered a shallow neural network while those with multiple hidden layers are considered deep neural networks. The general structure of an L-layer perceptron with n input neurons and m output neurons is shown in Fig. 2.1. During training, ANN learns by adjusting its weights as to minimize the training errors, which are defined as the differences between the desired outputs from the training samples and the modeled outputs by the ANN. The training goal is to achieve generalization. A neural model with good generalization can provide accurate answers even when it is tested with inputs that it has never encountered before in the training process. Normally, a set of unused data called test set is required to evaluate the generalization capability of a neural network. The test set cannot be used for training and validation purposes, or even to select one network from a group of candidate networks. Early stopping and regularization are often used to improve the generalization capability of a neural network and avoid overfitting. The readers are referred to reference [8] for more background about ANN.

While back-propagation remains as one of the popular training algorithms, it has been reported that the quasi-Newton, Levenberg-Marquardt and conjugate gradient methods can outperform primitive back-propagation in terms of speed and accuracy in the microwave modeling problems [9]. In [10], a comparative study between several training algorithms such as adaptive backpropagation, conjugate gradient, quasi-Newton, and Levenberg-Marquardt is performed. These algorithms are tested based on their performance on two circuit modeling problems, i.e., 3-conductor microstrip line and physics-based MESFET. A 3-layer MLP with 28 hidden neurons is used for the microstrip line example, whereas a 3-layer MLP with 60 hidden neurons is used for the metal-semiconductor field-effect transistor (MESFET) example. The training results for microstrip line and MESFET examples are presented in Table 2.1 and Table 2.2 respectively. The results show that for a neural network with smaller size, Levenberg-Marquardt is the fastest. However, Levenberg-Marquardt becomes slow compared to quasi-Newton for a large network.

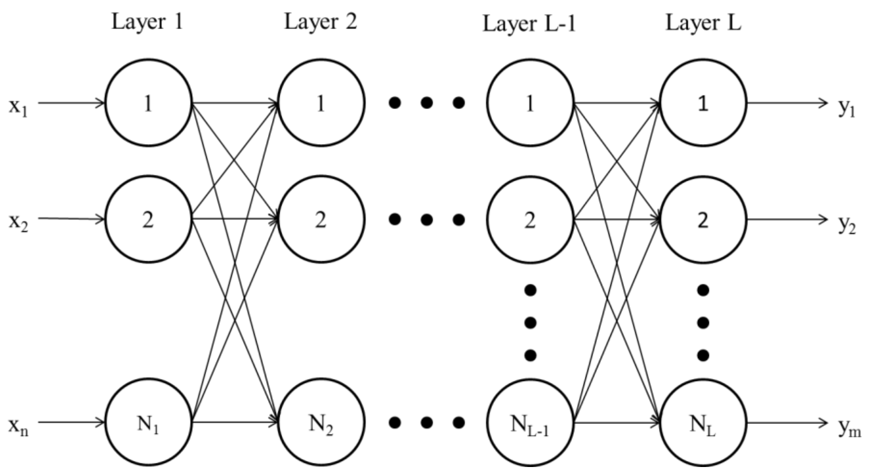


Figure 2.1: Generic MLP structure [10].

Table 2.1: Comparison of various training algorithms for the microstrip line example [10]

Training Algorithm	**No of Epochs**	**Training Error (%)**	**Avg. Test Error (%)**	**CPU(s)**
Adaptive backpropagation	10,755	0.224	0.252	13,724
Conjugate gradient	2169	0.415	0.473	5511
Quasi-Newton	1007	0.227	0.242	2034
Levenberg-Marquardt	20	0.276	0.294	1453

Table 2.2: Comparison of various training algorithms for the MESFET example [10]

Training Algorithm	**No. of Epochs**	**Training Error (%)**	**Avg. Test Error (%)**	**CPU(s)**
Adaptive backpropagation	15,319	0.98	1.04	11,245
Conjugate gradient	1605	0.99	1.04	4391
Quasi-Newton	570	0.88	0.89	1574
Levenberg-Marquardt	12	0.97	1.03	4322

The details about feedforward computation of MLP are presented in [11]. The feedforward computation is used to obtain neural network outputs, y_i from external inputs, x_i. The feedforward computation from the input to the output layer is as follows:

$$\text{Input Layer: } z_i^1 = x_i,\ \ i = 1, 2, \ldots N_1,\ \ n = N_1 \tag{2.1}$$

$$\text{Hidden Layer: } z_i^l = \sigma\left(\gamma_i^l\right),\ i = 1, 2, ..., N_l,\ l = 2, 3, ..., L \tag{2.2}$$

$$\text{Output Layer: } y_i = z_i^L,\ i = 1, 2, ..., N_L,\ m = N_L \tag{2.3}$$

where N_1 is the number of input neurons, N_L is the number of output neurons, and N_l is the number of hidden neurons at the lth layer. Fig. 2.2 shows the structure of the ith hidden neuron at the lth layer. Each neuron is connected to all outputs of the previous layer and each connection between ith neuron at lth and jth neuron at (l-1)th layer is assigned a weight, w_{ij}^l. Other than that, the neuron has an additional weight, w_{i0}^l named bias. Firstly, the weighted sum of the inputs, γ_i^l is computed. Then, the output of the neuron, z_i^l is obtained by applying the activation function, σ to γ_i^l. Usually, the activation function is the sigmoid function

$$\sigma\left(\gamma_i^l\right) = \frac{1}{1 + e^{-\gamma_i^l}}, \tag{2.4}$$

or the tansig function

$$\sigma\left(\gamma_i^l\right) = \frac{2}{1 + e^{-\gamma_i^l}} - 1, \tag{2.5}$$

where

$$\gamma_i^l = \sum_{j=0}^{N_{l-1}} w_{ij}^l z_j^{l-1}. \tag{2.6}$$

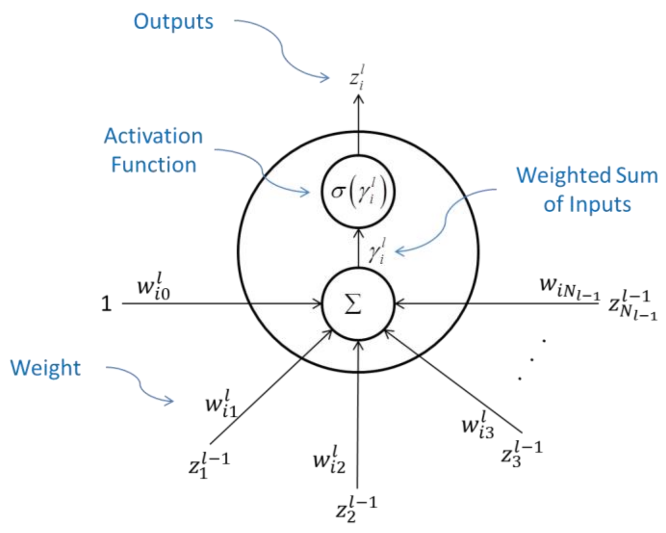


Figure 2.2: The structure of the *i*th hidden neuron of the second layer [11].

2.3 Signal Integrity Modeling with Artificial Neural Networks

Recent works show that ANNs have also been used in fast SI modeling applications [12-18]. Most of the works focus on eye diagram modeling or prediction of SI metrics such as crosstalk or jitter. The eye diagram is a graphical metric that is commonly used to evaluate the performance of high-speed systems. Fig. 2.3 shows an eye diagram with its height, width and timing jitter labeled. The eye diagram is constructed by slicing the time-domain signal waveform into sections that are a small number of symbols in length and overlaying them on top of each other. Ideally, the eye-opening should be as wide as possible so that the design will have enough margins for voltage and timing requirements at the receiver. An eye diagram can also be used to estimate the bit error rate (BER) of a system, which is the rate at which error occurs in digital data transmissions.

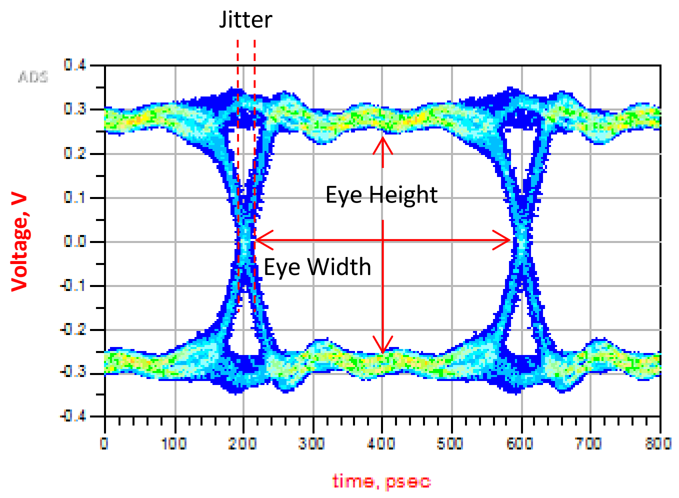


Figure 2.3: Example of a received eye diagram of a 2.5 Gbps system.

2.3.1 Sampling Techniques

Neural networks can only be used for circuit modeling after it is trained. Thus, the first step in neural modeling is to generate input-target pairs of the problem to be learned. Often, the generated input-target pairs are divided into three groups called the training set, a validation set, and a testing set. The training set is used to adjust the weights and biases during the training process of the neural model. The validation set is used to determine the stopping criteria of the training process. Finally, the testing set is used for an unbiased performance estimation of the neural model. Design of experiments (DoE) is a very popular sampling method in various neural modeling applications. One of its variants called the Taguchi design of experiments, which use orthogonal arrays for efficient design space exploration, has been used for the eye height and timing jitter neural modeling of high-speed interconnects [12]. Although DoE has been used in many applications and proven to be an effective sampling strategy, DoE sampling is a one-shot approach which means that the sampling process and training process are carried out separately. If the engineers do not have the full understanding of the input-target mapping function, it is very difficult to decide the sample size to be generated. Also, the degree of linearity may not be consistent throughout the whole design space. These problems can cause undersampling or oversampling to occur.

In order to solve this problem, adaptive sampling is proposed [7]. Instead of generating all of the samples at once, adaptive sampling only generates a small number of samples at first, and then adds further samples iteratively. The idea is to add more samples in a highly non-linear region of the design space compared to the linear regions. At the start of the algorithm, the whole design space is divided into 2^d equal volume regions where d is the number of input variables of the problem. Sometimes, the algorithm can also start with just a single region. Then, training and validation samples are generated for each region and a neural model is created using the newly generated samples. All regions are given performance scores based on the validation errors, which are the errors between the neural network predictions and the targets of the validation samples. The region with the worst performance (biggest validation errors) is then further split into 2^d equal volume regions. The process continues until the neural network's performance meets the minimum user-defined accuracy. The user-defined accuracy can be any type of performance matrices such as mean-square error and R^2. Fig. 2.4 illustrates the splitting of the worst performing region during the adaptive sampling process for a 2-dimensional modeling problem. In this case, the training points are generated at the corners of the regions, whereas the validation points are generated at the center of the regions. Once the splitting is completed, the validation point of the worst performing region becomes a training point at the next iteration. The pseudo-code of the adaptive sampling algorithm is presented in Algorithm 2.1.

Algorithm 2.1 Adaptive Sampling

```
Initialize design space as region $R$;
while($netPerformance$ < $desiredPerformance$)
{
    Generate non-existing training and validation samples within $R$;
    Create and train an intermediate neural model using training samples;
    Compute the network's performance using validation samples;
    if($netPerformance$ < $desiredPerformance$)
    {
        Identify worst performing region as $R^*$;
        Split $R^*$ into equal volume regions $R_1, R_2, \dots, R_k$, where $k = 2^d$;
        Delete $R^*$ from $R$;
        Add regions $R_1, R_2, \dots, R_k$ into $R$ where $R = [R, [R_1, R_2, \dots, R_k]]$;
    }
    End
}
End
```

The adaptive sampling also allows easier integration between data generation and neural model creation. Several papers on automated model generation (AMG) use the adaptive sampling as part of the AMG algorithm [19-22], and these works focus on the modeling of S-parameters. In this work, adaptive sampling method is used for data generation for the modeling of eye diagram from design parameters.

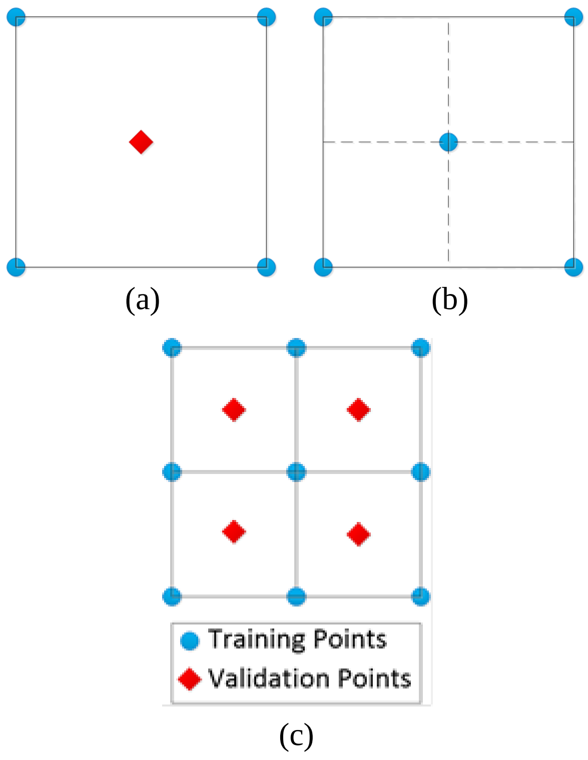


Figure 2.4: 2D visualization of the splitting process. a) The worst performing region. b) The region is split into 2*n* new regions, four new regions in this case; also the validation point at the center of the original region is changed to a training point. c) Generate new training and validation points for the worst performing region.

2.3.2 Input Data Selection/Preprocessing

Some researchers use the frequency responses of the circuits, such as the S-parameters, in place of the design parameters as inputs of their neural models [13-16]. It is a common practice to describe a complex design with its port responses instead of its actual design parameters in order to protect any proprietary information about its structure. This method is useful when the time domain simulation is much slower than the frequency domain simulation. If that is the case, it is beneficial to perform frequency domain simulation on large number of designs, and then only select some relevant designs which are selected as training samples for time domain simulation. Thus, the total number of time domain simulation can be reduced. However, this technique has a few weaknesses compared to the conventional approach of using design parameters. Firstly, the neural model generated is probably not suitable for use in optimization routines since it is very difficult to know the structures of the designs just from the S-parameters alone. Secondly, and perhaps more importantly, the neural model, in this case, does not replace the EM field solver, which is usually the most time-consuming part of the design simulation. Third and finally, the neural model will have a large number of inputs, which may slow down the training process. This is because S-parameters are frequency dependent and are normally generated for at least a few hundred points across the frequency range of interest. In addition, for an *N*-port network, the S-parameter matrix has N^2 elements. For example, the S-parameter matrix of a 2-port network has four elements, S_{11}, S_{12}, S_{21}, and S_{22}. Therefore, the total amount of inputs will be doubled due to the fact that S-parameters are complex numbers, with real and imaginary parts. Despite this, continuous researches are being carried out in this area to improve the accuracy and decrease the model development cost.

In order to reduce the number of simulations, a method called reduced training set (RTS) has been proposed [14]. Initially, the frequency responses of all designs are generated. Then, a certain number of frequency points, *K* are selected. After that, three designs that contribute to the maximum, median, and minimum values of the frequency responses are selected as the training samples for each of the selected frequency. This is visualized in Fig. 2.5. The sample sizes of DoE, S_{DoE}, and RTS, S_{RTS} are given by:

$$S_{DoE} = 2^{d-1} + 2d + 1 \tag{2.7}$$

$$3M \le S_{RTS} \le 3MK \tag{2.8}$$

where *M* is the number of S-parameters used as inputs. For example, if S_{11} and S_{21} are both selected as inputs, *M* is two. Unlike S_{DoE}, S_{RTS} does not grow exponentially with *d*. Thus, this technique can reduce the amount of time domain simulations. RTS technique is highly effective when the time for frequency domain simulations is much shorter than time for time domain simulations.

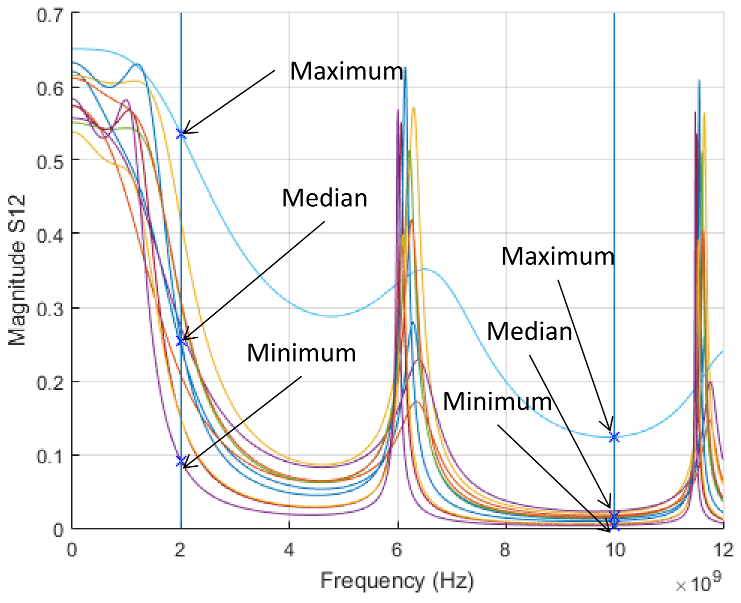


Figure 2.5: Generation of input data for RTS.

Besides that, feature selection techniques can be applied to extract the relevant frequencies from large amounts of uniformly sampled frequencies, which can be up to hundreds or thousands of points. In [16], a feature selection method centered around a fast correlation-based filter (FCBF) is used to identify the relevant S-parameters metrics. This allows the engineers to train the neural model with only relevant inputs, resulting in a more compact and accurate neural model.

Other than that, vector fitting has also been used to reduce the number of inputs into neural networks. Vector fitting is a robust numerical method for rational approximation in the frequency domain with poles and residues. The vector fitting method approximates a rational function *f(s)* as:

$$f(s) = \sum_{q=1}^{Q} \frac{r_q}{s - p_q} + D + sh \tag{2.9}$$

where Q is the order of approximation, r_q are the residues, p_q are the poles, D and h are the constant and proportional terms respectively. If $f(s)$ does not have an asymptotic value, term D and h can be set to zero, reducing Equation (2.9) into:

$$f(s) = \sum_{q=1}^{Q} \frac{r_q}{s - p_q} \tag{2.10}$$

The detailed description about vector fitting can be found in [23-25]. In [26], poles and residues are used in place of S-parameters as inputs to neural networks. The S-parameters of the designs are extracted into poles and residues as in Equation (2.10). A higher order of approximation, Q can lead to better fitting accuracy during the vector fitting process, but it can also increase the number of inputs to neural networks. Thus, it is important to keep the Q as small as possible without compromising the fitting accuracy. The poles and residues, with their corresponding eye heights and eye widths are used for the neural model development process. The comparison between neural networks with S-parameters and poles/residues as inputs in terms of training time is shown in Table 2.3. The results display significant speedup when vector fitting is used, the speedup factor ranges from about 22× (1 hidden neuron) to 1642× (10 hidden neurons). The speedup factor is obtained by dividing the average training time of neural models with S-parameters as inputs by the average training time of neural models with poles and residues as inputs. This is because the vector fitting reduces the total number of inputs to the neural networks from 1002 to just 108, thus reducing the amount of time required for the learning process of neural networks.

Table 2.3: Training speed comparison of the neural models [27]

No. of Hidden Neurons	**Average Training Time of Neural Models with S-parameters as Inputs (s)**	**Average Training Time of Neural Models with Poles and Residues as Inputs (s)**
1	5.16	0.23
2	31.15	0.24
3	77.31	0.29
4	199.86	0.35
5	398.91	0.50
6	455.62	0.63
7	661.28	0.84
8	1069.50	1.04
9	6261.60	1.18
10	6418.70	3.91

2.3.3 Eye Diagram Prediction

Eye diagram prediction is one of the most commonly seen applications of neural networks in the field of signal integrity. Specifically, most of the works focus on modeling the eye-height and eye-width [13-18]. This is because the minimum height and width of signals at the receiver are important metrics for the performance evaluation of a high-speed channel. Sometimes, a neural network is also used to model the timing jitter of the eye [12]. The eye diagram modeling problem is described as follows. Suppose there is a function that maps design parameters (input) to eye metrics (output), and that a neural network is to be used to learn that function. Conventionally, two simulations are carried out to obtain the eye diagram. First, frequency responses such as S-parameters are extracted from the design. Then the S-parameters are used as a representation of the design in a transient simulation to obtain the output waveform and the eye diagram is constructed. Therefore using a trained neural network as a replacement for this mapping, both simulations can be omitted, speeding up the design process significantly. This is especially useful in cases where the eye diagram needs to be generated repeatedly as the design parameters are tweaked, for example during an optimization process. In this work, the neural networks are responsible for mapping the design parameters to eye diagram metrics, which allow us to replace both simulators with the neural models.

2.4 Knowledge Embedded Neural Networks

Sometimes, the performance of the ANN is not satisfactory when the modeled problem has a highly nonlinear input-output relationship. In this case, the existing knowledge can be incorporated into the structure of ANN. Several types of knowledge embedded neural model are discussed such as prior knowledge input ANN (PKI-ANN) and difference neural network.

2.4.1 Prior Knowledge Input ANN

A PKI-ANN is just like a conventional ANN except that the PKI-ANN has one or more additional input, and the inputs are the outputs of a mathematical model. The mathematical model can be an empirical model or another ANN. For example, the output space mapping neural model is a type of PKI-ANN where it models the fine model responses from the coarse model responses and the design parameters. The fine model represents an accurate but slow model such as the 3D EM field solver whereas the coarse model represents a faster but less accurate model such as the empirical model. Let the design parameters be x_f, the independent variables be φ, the coarse model responses be $f_c(x_f, \varphi)$, the fine model responses be $f_f(x_f, \varphi)$, and the PKI-ANN outputs be y_{PKI}. The independent variables can be frequency if the outputs are in the frequency domain, and can be time if the outputs are in the time domain. The main purpose of the output space mapping model is to map the coarse model responses to fine model responses, which gives the speed of the coarse model and accuracy of the fine model. The additional information from the coarse model can reduce the complexity of a modeled problem compared to the modeling from design parameters alone. This is because the problem has been changed to just correcting the mistake made by the coarse model, which is usually easier. Sometimes, another ANN can also be used as the coarse model. For example, a first ANN that is trained to model the S-parameters of a transistor can be used in the construction of a second ANN, which is used to model the S-parameters of another transistor that belongs to the same class [27]. However, the performance of PKI-ANN largely depends on the quality of the coarse model. A good coarse model can improve the accuracy of the PKI-ANN significantly while a bad coarse model may give only minimal improvement. The general structure of a PKI-ANN with the output space mapping approach is shown in Fig. 2.6.

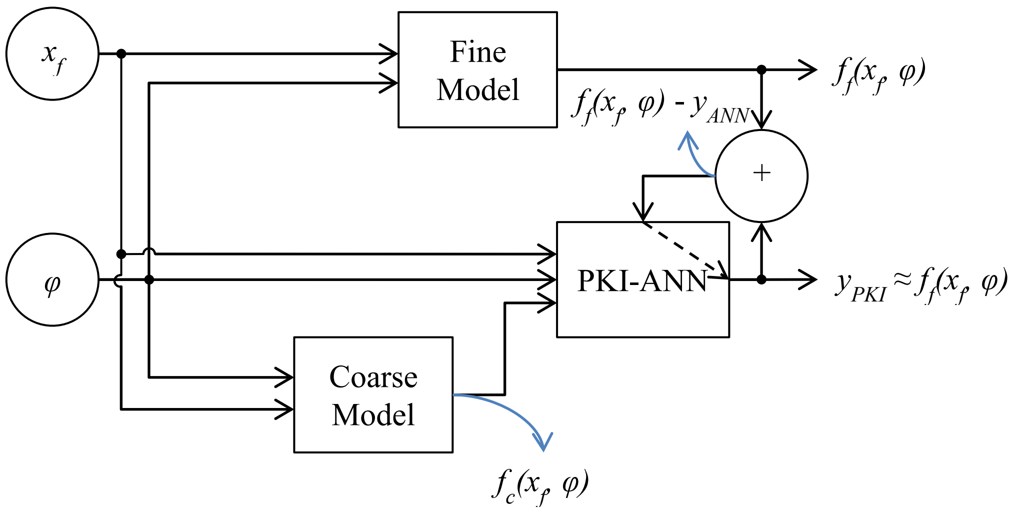
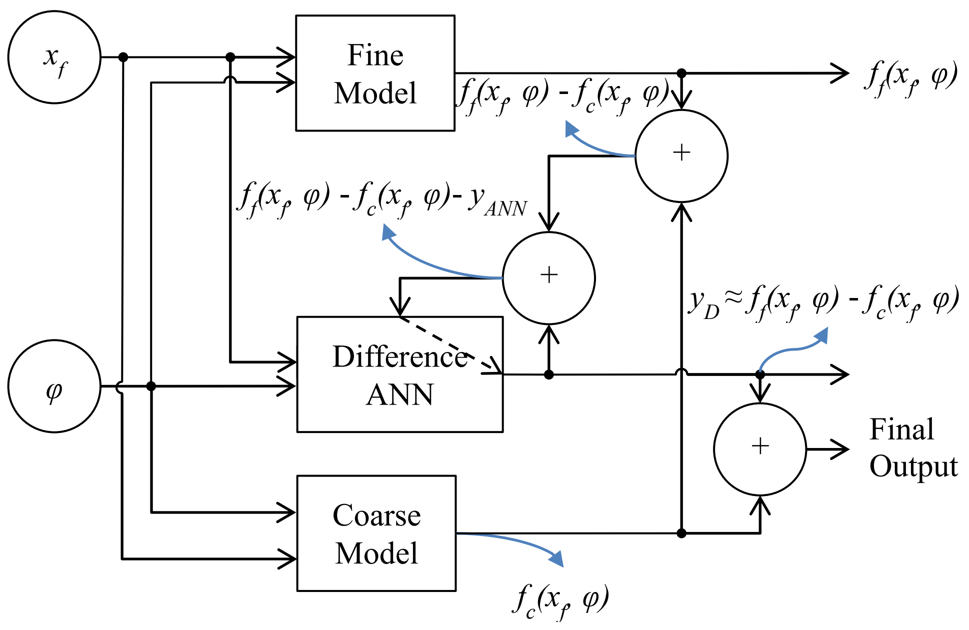


Figure 2.6: General structure of a PKI-ANN with output space mapping approach, adapted/modified from [27].

2.4.2 Difference ANN

A difference ANN is a type of primitive PKI-ANN. Instead of modeling the fine model responses, the difference ANN models the differences between fine and coarse model responses. Then, the outputs of the difference ANN, y_D is summed up with the corresponding coarse model responses to obtain the final outputs. The general structure of a difference ANN is shown in Fig. 2.7. Usually, the designers assume that it is easier to model the differences than to model the actual fine model responses. However, this is not always the case. Sometimes, the differences can have a more complicated relationship with the inputs than that of the fine model responses. In this case, the improvement may be small or even none.



x_f
Fine Model
$f_f(x_f, \varphi)$
$f_f(x_f, \varphi) - f_c(x_f, \varphi)$
+
$f_f(x_f, \varphi) - f_c(x_f, \varphi) - y_{ANN}$
+
$y_D \approx f_f(x_f, \varphi) - f_c(x_f, \varphi)$
φ
Difference ANN
+
Final Output
Coarse Model
$f_c(x_f, \varphi)$

Figure 2.7: General structure of a difference ANN, adapted/modified from [27].

2.4.3 Prior Knowledge Input Difference ANN

A PKI-ANN and a difference ANN can be used in combination to construct a prior knowledge input difference ANN (PKID-ANN). The general structure of a PKID-ANN is shown in Fig. 2.8. The PKI-ANN makes use of the existing knowledge two times during the forward computation, first for the PKI-ANN and second for the difference ANN. In [28], a 3-step modeling strategy using knowledge-based technique is proposed. The first step is to create a conventional ANN. The second step is to create a PKI-ANN by using the conventional ANN created in the first step as the coarse model. In the third step, a difference ANN is created to model the differences between the fine model and the PKI-ANN created during the second step. The PKID-ANN outputs and the difference ANN outputs are summed to obtain the final outputs, y_{PKID}. It is shown that the 3-step modeling strategy can give lower errors as compared to the conventional neural modeling. Moreover, it takes less time to train as well, provided that the total number of iterations and hidden neurons are the same for both methods. The comparison between 3-steps and conventional ANN techniques for Branin function modeling are tabulated in Table 2.4. Mathematical formulation of Branin function is given as follows:

$$f\left(x_1,x_2\right)=\left(x_2-\frac{5x_1^{\ 2}}{4\pi^2}+\frac{5x_1}{\pi}-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10. \tag{2.11}$$

Both the 3-steps and conventional ANNs are trained using 10000 points. The 3-steps ANN is constructed by combining ANN models developed during step one, two, and three. As can be observed, the 3-steps ANN is able to give lower maximum and mean errors compared to the conventional ANN. Other than that, the 3-steps ANN also requires less time during the training process.

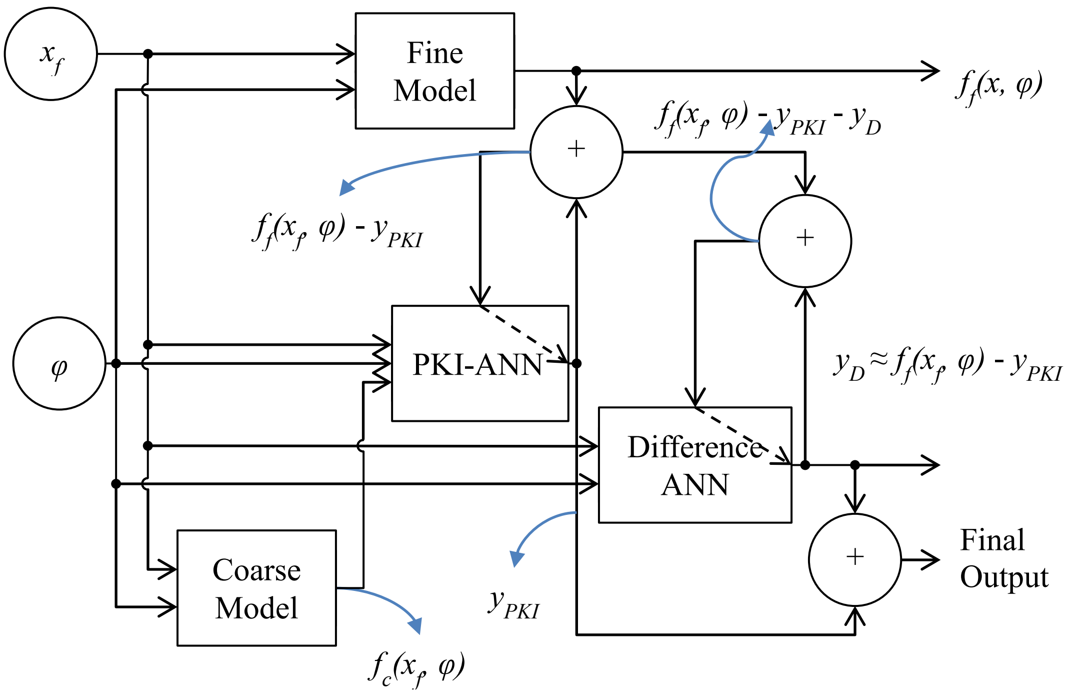


Figure 2.8: General structure of a PKID-ANN, adapted/modified from [28].