Title
Enhancing the Electromagnetic Design Process with Explanation Algorithms

Author
Ho, David

Publication Date
2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Enhancing the Electromagnetic Design Process with Explanation Algorithms

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science
in Materials Science and Engineering

by
David Ho

2021

ABSTRACT OF THE THESIS

Enhancing the Electromagnetic Design Process with Explanation Algorithms

by
David Ho
Master of Science in Materials Science and Engineering
University of California, Los Angeles, 2021
Professor Aaswath Pattabhi Raman, Chair

Designing photonic structures and obtaining optimal responses is still a difficult task. A large number of gradient based algorithms and adjoint solvers are used, and the results have been effective in that complex free form geometries of devices are capable of being created. The fact remains, however, that many solvers still tend to be black boxes, and there is very little transparency in how the solver comes to its conclusion. Furthermore, because many of these solvers are gradient based, additional knowledge of the device is needed, otherwise the solver can get trapped in a local minimum and not reach the true optimal geometry. To this end, an inverse design framework that combines adjoint optimization, automated machine learning (AutoML), and explainable artificial intelligence (XAI) is presented in order to determine both the relation between device structure and performance and also minimize the effect of local minima trapping. This framework is used in the inverse design of waveguides and achieves an average of 43% increase in performance. Consequently, this study portrays how to extend beyond traditional inverse design solvers in terms of both performance and ability to elucidate the solvers' decisions.

The thesis of David Ho is approved.

Amartya Sankar Banerjee

Benjamin S Williams

Aaswath Pattabhi Raman, Committee Chair

University of California, Los Angeles

December 2021

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Professor Aaswath Raman, for his guidance, immense knowledge, and support that dated back to when I was an eager undergraduate student at UCLA who wanted to learn more about the various ongoing research projects that he was overseeing when I knew nothing about photonics and had a rudimentary understanding of programming, let alone any sort of machine learning knowledge. I cannot imagine having a different advisor. Besides my advisor, I would also like to thank Professor Banerjee and Professor Williams for also being a part of my thesis committee.

Next, I would like to sincerely thank my research mentor, Christopher Yeung, for these past years, and for reaching out to me during my last year of undergrad and asking if I would like to aid his project which would then lead me down the current path that I am on as a researcher. Chris, I thank you not just for your patience as you sat down with me and took the time to thoroughly guide me, but I also just want to say thank you for being a good friend for the past few years.

Last, but certainly not least, I would like to thank my parents and my two sisters, Tien and An, for the support that they have provided me these last few years as I wrote this thesis. To my mother and father, Chau and Bong, thank you for always seeing the best in me and believing in me even when I could not. To Tien, thank you for all of your advice in the life of graduate school and for being a steadfast part of my life. To An, thank you for being so supportive of me, and for being a friend I can talk to about anything that came up in life. You might be 1000 miles away, but you have always made sure to never be distant. And also, I want to thank my family for all the support they have provided me my entire life. Without them, I most definitely would not be where I am today.

TABLE OF CONTENTS

List of Tables vii

List of Figures viii

List of Equations x

Chapter 1: Introduction to Photonic Devices 1

1.1 Background 1

1.2 Basic Principles 1

1.3 FDTD Method 2

1.4 FDTD Simplifications 6

1.5 FDTD Strengths and Limitations 7

Chapter 2: Optimization Algorithms 8

2.1 Motivations 8

2.2 Background 8

2.2.1 Evolutionary Algorithms 10

2.2.2 Trajectory Algorithms 10

2.2.3 Adjoint-Based Optimization 11

2.3 Machine Learning Algorithms 11

2.3.1 Convolutional Neural Networks 14

Chapter 3: CNN Training of Adjoint Based Optimization Structures 16

3.1 Creation of Training Data 16
3.2 CNN Problem Definition 19
3.3 Autokeras 20
3.4 Verifying CNN Results 21

Chapter 4: Explanation Algorithm Implementation 23
4.1 Introduction to Explainable Artificial Intelligence 23
4.2 Shapley Additive Explanations 23
4.3 SHAP Redesigns 24
4.4 Simulating Explanation-Optimized Structures 27

Chapter 5: Conclusions 30

Appendix A: FDTD Simulations 32

Appendix B: Device Structure Geometry Evolution 35

Appendix C: Autokeras Implementation 36

Appendix D: Boundary Finder Algorithm 38

Appendix E: Device Random Perturbations 39

Appendix F: Silicon Nitride Waveguide Optimization 41

References 46

LIST OF TABLES

3.1 Final Figure of Merit Values For Each Target Wavelength 19

3.2 CNN Prediction Vector vs FDTD Ground Truth Vector 21

4.1 Comparison of Original and New FOM Values 28

LIST OF FIGURES

1.1 Yee Cell Geometry with the Field Vector Distributions 4

1.2 Perfectly Matched Layer Boundary 6

2.1 Breakdown of Optimization Algorithms 9

2.2 Vanilla Artificial Neural Network Architecture 13

2.3 Convolutional Neural Network Architecture 14

3.1 Initial Y-Splitter Geometry 16

3.2 FDTD Optimization Report 17

3.3 Final Device Geometries for Each Target Wavelength 18

3.4 Figure of Merit vs Design Iteration Across All Optimization Runs 19

3.5 Training and Validation Loss of AutoML Optimized Architecture 20

3.6 CNN Prediction Compared Against FDTD Simulation 21

4.1 SHAP Values Across Various Designs 25

4.2 Workflow for Structure Redesign 26

4.3 Explanation Optimized Structures for Each Target Wavelength 26

4.4 Two Stage Figure of Merit Optimization for Waveguide Geometry 27

A.1 1D FDTD Parameters 32

A.2 E-Field Source Parameters 33

A.3 Perfectly Matched Layer Parameters 33

A.4 FDTD Function 34

A.5 1D FDTD E-Field Exaxmple 34

B.1 Adjoint Optimization Structure Evolution Target Wavelength = 1.3 μm 35

C.1 Training and Validation Losses with respect to Autokeras Trials 36

C.2 Autokeras Image Block/Model Architecture Evolution 37

E.1 Comparison of Random Perturbation vs. SHAP 40

F.1 Silicon Nitride Two Stage Optimization 41

LIST OF EQUATIONS

1.1 Gauss's Law for Electricity 2

1.2 Gauss's Law for Magnetism 2

1.3 Maxwell-Faraday Equation/Faraday's Law of Induction 2

1.4 Ampere-Maxwell Equation/Ampere's Circuital Law 2

1.5 Taylor Expansion of a Function with Positive Offset 3

1.6 Taylor Expansion of a Function with Negative Offset 3

1.7 Higher Derivative Terms Central Difference 3

1.8 Central Difference Approximation for Small Deltas 3

1.9 1D Example of Faraday's Law 5

1.10 1D Example of Ampere's Law 5

1.11 Finite Difference Faraday's Law 5

1.12 Magnetic Field Update Equation 6

2.1 Adjoint-Method Equation 11

2.2 ReLU Activation Function 13

2.3 Sigmoid Activation Function 13

2.4 Hyperbolic Tangent Activation Function 13

3.1 Y-Splitter Figure of Merit 17

3.2 Mean Squared Error 22

4.1 Shapley Additive Explanation Values 23

4.2 Binarization Step Function 25

Chapter 1

INTRODUCTION TO PHOTONIC DEVICES

1.1 Background

The understanding of the manner in which light interacts with matter still has deep potential for advancement. The study of photonics involves creating devices that can manipulate, detect, or create light [1]. This field has paved the way for numerous inventions that have vastly improved quality of life, such as the laser, which helped enable technologies such as photolithography for semiconductor manufacturing and laser surgery, the light emitting diode, which is used in everyday lighting and digital displays, the photovoltaic cell, which gave way to the solar cell and the potential for increased renewable energy, and much more [2][3][4].

Photonics has helped shape the world so far, and it has the potential to continue doing so. More recent advancements in the field of photonics involve photonic integrated circuits (PICs) that are used for fiber-optic communication, and metasurfaces, artificial sheet-like materials with sub-wavelength thicknesses that can modulate the behavior of electromagnetic waves through a desired boundary condition set by the designer [1][5]. In fact, sub-wavelength silicon-on-insulator (SOI) PICs have the potential to make CMOS electronics have reductions in size, cost, and energy consumption [6][7]. As a result, there is a great amount of potential waiting to be unlocked in this field if these devices can be easily designed.

1.2 Basic Principles

There are multiple approaches in which photonic devices have been designed, and one of the first ways to do so is through solving Maxwell's Equations, and in order to do so, many computationally expensive and tedious numerical simulations must be performed due to the complexity of

the device.

$$\nabla \cdot \boldsymbol{E} = \frac{\rho}{\epsilon_0} \tag{1.1}$$

$$\nabla \cdot H = 0 \tag{1.2}$$

$$\nabla \times \boldsymbol{E} = -\frac{\partial H}{\partial t} \tag{1.3}$$

$$\nabla \times H = \mu_0 (J + \epsilon_0 \frac{\partial \boldsymbol{E}}{\partial t}) \tag{1.4}$$

Equations 1.1 through 1.4 are the differential equations that make up Maxwell's Equations, and they relate the electric field vector (E), the magnetic field vector (H), permittivity of free space ϵ_0, permeability of free space μ_0, electric charge density ρ, and electric current density J to one another. Equation 1.1 is commonly known as Gauss's Law, which relates the gradient of the electric field vector to the charge density, and this can then be used to derive other laws such as Coulomb's Law. Equation 1.2 is a special form of Gauss's Law for magnetic fields, stating that the gradient of the magnetic field is equal to zero, which dictates this magnetic vector field as being solenoidal. Equation 1.3 is known as the Maxwell-Faraday law, which dictates that a time varying magnetic field results in a spatially varying electric field. Equation 1.4 is the Ampere-Maxwell law which relates a time varying electric field to a spatially varying magnetic field.

1.3 FDTD Method

Maxwell's equations are central to photonic device design, but a key roadblock is in the lack of analytical solutions to these equations. A common method that is used is known as the Finite

Difference Time Domain (FDTD) method, which is also known as Yee's Method [8]. The FDTD method is predicated on the observation that any time change in electric field is related to the spatial change in the magnetic field, and vice versa. The method is similar to other numerical methods in that it uses finite differences to approximate the spatial and temporal derivatives seen in Ampere's law and Faraday's law [ref]. The finite differences work in the following manner: consider the Taylor Expansion of $f(x)$ expanded about x_0 with an offset of $\pm\delta/2$ [9]:

$$f(x+\frac{\delta}{2}) = f(x_0) + \frac{\delta}{2}f'(x_0) + \frac{1}{2!}(\frac{\delta}{2})^2 f''(x_0) + \frac{1}{3!}(\frac{\delta}{2})^3 f'''(x_0) + \dots \tag{1.5}$$

$$f(x-\frac{\delta}{2}) = f(x_0) - \frac{\delta}{2}f'(x_0) + \frac{1}{2!}(\frac{\delta}{2})^2 f''(x_0) - \frac{1}{3!}(\frac{\delta}{2})^3 f'''(x_0) + \dots \tag{1.6}$$

These two equations differ from one another in the even terms (second and fourth term of Equation 1.5 is positive whereas the second and fourth term of Equation 1.6 is negative). By subtracting Equation 1.5 from Equation 1.6 and then dividing by δ, the following equation comes to light:

$$\frac{f(x+\frac{\delta}{2}) - f(x-\frac{\delta}{2})}{\delta} = f'(x_0) + \frac{1}{3!}\frac{\delta^2}{2^2}f'''(x_0) \tag{1.7}$$

Equation 1.7 is very familiar looking, in that it is the equation for the derivative of the function f(x) alongside a higher order term that includes a δ^2 value as well. There are many other higher order derivatives included in this equation(fifth, seventh, etc.), but they end up including terms of δ^4, δ^6, and so on, which results in these terms being negligible. If δ becomes sufficiently small, then the following equation is formed:

$$\lim_{\delta\to 0}\frac{f(x+\frac{\delta}{2}) - f(x-\frac{\delta}{2})}{\delta} \approx f'(x_0) \tag{1.8}$$

This is known as the central difference approximation. Thus, equation 1.8 allows for an ap-

proximation of $f'(x)$ at x = x_0 even though the function is never sampled there, but rather at the nearby points x + $\delta/2$ and x - $\delta/2$. If δ becomes exactly zero, then the approximation becomes the exact definition of the derivative of $f(x)$ at $x = x_0$.

The FDTD/Yee Method uses the central difference approximation to execute the following: Replace the various spatial and temporal derivatives and Ampere's and Faraday's laws with the central difference approximation, discretize space and time in order for the electric and magnetic fields to be staggered both spatially and temporally, solve the difference equations to obtain equations that express future fields in terms of past fields, evaluate both of the fields one time step into the future which then provides the spatial component as well, and continue to do so until the fields have been solved for the desired time [9].

In 3D Cartesian coordinates, this method is supplemented by the Yee Cell, which illustrates the distribution of electric and magnetic field components as the whole area being worked with is now discretized into small cells. In Figure 1.1, a cell with side lengths Δx, Δy, and Δz is formed

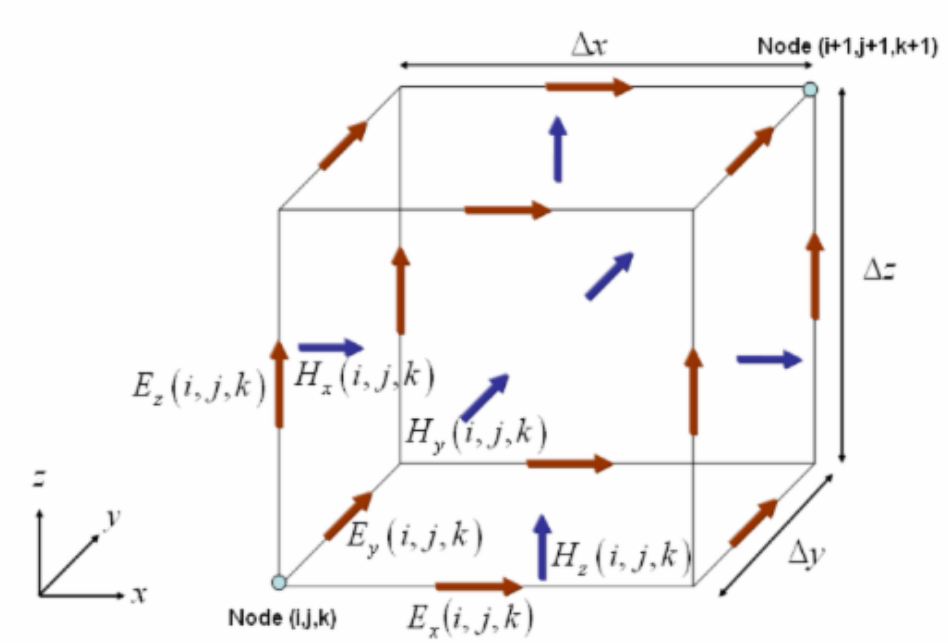

Figure 1.1: Yee Cell Geometry with the Field Vector Distributions

with an electric field vector being placed at an edge midpoint, whereas the magnetic field vector is placed at a face midpoint, and this cell demonstrates what it means to stagger the electric and magnetic fields by both half a spatial step and half a time step.

As stated previously, the next step would be to solve the difference equations in order to obtain

equations. By shifting the vectors by half a step both spatially and temporally, the derivatives can now be approximated with the central difference. This can first be shown in a 1D case of Faraday's Law for simplicity's sake. If there are only variations in the z direction, and the electric field only exists in the x direction, then this law can be written as follows:

$$\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ 0 & 0 & \partial/\partial z \\ E_x & 0 & 0 \end{vmatrix} = \frac{\partial E_x}{\partial z}\hat{j} = -\frac{\partial H}{\partial t} \tag{1.9}$$

Ampere's Law can now also be written in the 1D case as a result of what is seen in Equation 1.9, and it is evident that the only time varying magnetic field component is in the y-direction.

$$\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ 0 & 0 & \partial/\partial z \\ 0 & H_y & 0 \end{vmatrix} = \frac{\partial H_y}{\partial z}\hat{i} = -\mu_0 (J + \epsilon_0 \frac{\partial E_x}{\partial t}) \tag{1.10}$$

Next, the derivatives in Equation 1.9 and Equation 1.10 get replaced with finite differences derived earlier as time and space become discretized now, as seen in Figure 1.1. An example of Faraday's Law written with finite differences will be as follows with the δ variable now being changed to Δx, Δy, or Δz, or Δt depending on whether a spatial or temporal derivative is being taken:

$$\frac{E_x(z + \Delta z, t) - E_x(z - \Delta z, t)}{\Delta z} = \frac{H_y(y, t + \Delta t) - H_y(y, t - \Delta t)}{\Delta t} \tag{1.11}$$

This method results in something known as the leapfrog method [8]. The temporal and spatial offsets that arise mean that in order to approximate Maxwell's Equations, one must calculate the electric field values, and then the magnetic field values where if all of the electric field values are calculated at time $n\Delta t$, where n is the time step in the algorithm (1,2,3 etc.), then the magnetic field values will be calculated at time $n - (1/2)\Delta t$ and $n + (1/2)\Delta t$. Thus, one can solve for a future magnetic field value that contains a past magnetic field value and the current electric field value.

This is known as an update equation:

$$H^{(n+1/2)}(y) = H^{(n-1/2)}(y) + \Delta t\left(\frac{E^n(z+1/2) - E^n(z-1/2)}{\Delta z}\right) \tag{1.12}$$

The update equation for the electric field is found in a similar fashion, but instead starts with Ampere's Law. The combination of these two equations now allows many more E&M problems to be solvable.

1.4 FDTD Simplifications

There are also ways to simplify the FDTD calculations. One way to do so is to introduce the Perfectly Matched Layer (PML) as a boundary condition. When using the FDTD method, Maxwell's equations must be solved in a discretized space, but there is a work-around to do so for a theoretically boundless space. The PML is an artificial absorbing layer that perfectly absorbs outgoing waves and does not reflect any of them back into the interior of the design space [10]. Nonethless, the PML has its own set of shortcomings. One large problem comes from the fact that

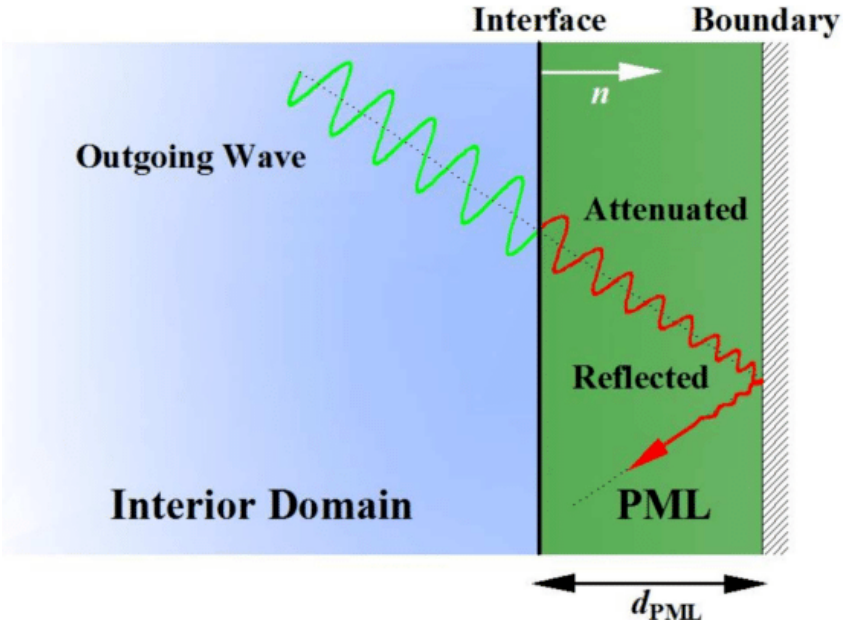

Figure 1.2: Perfectly Matched Layer Boundary

the medium must be invariant in the orthogonal direction to the boundary, thus this approach cannot be applied to photonic crystals [11]. Another issue arises in exotic materials that display negative

refractive index, and as a result exponential growth occurs rather than exponential decay, which is the exact opposite of what the PML intends to do [12].

1.5 FDTD Strengths and Limitations

As we can see, there are obvious strengths to the FDTD Method, but there also weaknesses. Some strengths involve its intuitiveness where users can understand the method and know what to expect from a given model, it is a time domain method, the user can specify the material at all given points in the domain and a large number of both linear and nonlinear dielectric and magnetic materials can be modeled, and the FDTD Method directly uses electric and magnetic fields, therefore requiring no additional conversions following the simulation [13]. Weaknesses of the FDTD approach involve problems such as the entire computational domain being placed into a grid, and thus all the discrete steps in the method need to be small enough to resolve the smallest feature size and the smallest wavelength in the model, and this results in very large computational domains with incredibly long solution run times with an example of a difficult to model system being a long thin wire, and the FDTD approach must satisfy the Courant-Friedrichs-Lewy Condition for numerical solutions to partial differential equations [14][15].

The challenges that occur upon using the FDTD Method are significant, because the 1D case shown is simply not practical for any development in technologies, and as a result further investigation into optimization algorithms have been explored for photonics design problems.

Chapter 2

OPTIMIZATION ALGORITHMS

2.1 Motivations

The first advancement towards creating photonic devices was to find a practical solution via numerical algorithms that could approximate solutions to Maxwell's Equations, and the natural progression from there would be to then optimize these algorithms in such a way for the run time and computational complexity to decrease. This desire for increased efficiency extends far beyond that of the photonics realm. Optimization problems began as simply as the calculus of variations' brachistochrone problem which was solved by numerous mathematicians such as Newton and Bernoulli, or the classical mechanics' principle of least action which was proposed by Maupertuis and then Euler [16][17]. More recent photonic/nanophotonic optimization involves algorithms that perform structural optimization of photonic devices, such as determine the width of a SiON core for a SiO_2 on SiON waveguide, or optimize the bandgap of a structure through determining the proper composition of dielectric material in the system [18][19]. These attempts to create more efficient algorithms stem from the following: existing design methods depend on intuition, which means that non-intuitive solutions are unlikely to be considered even if the result can exceed that of what has already been achieved, device requirements are becoming more complex, such as nonlinear capabilities which coincides with a new lack of intuition for these devices, and brute force methods are not only computationally expensive, but can also miss global optima [20].

2.2 Background

Optimization techniques are beneficial in that any new technique can automatically determine the best design. These algorithms can either be classified as heuristic or meta-heuristic. Heuristic algorithms are approaches that are "problem designed" so to speak in that the algorithm would only be applicable to one specific problem. Heuristic algorithms are fast and efficient, but besides

the narrowness of their application, they also sacrifice optimality, accuracy and precision [21]. A famous example of a heuristic algorithm is the Traveling Salesman Problem, in which the problem asks: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" [22]. The nearest neighbor heuristic approach helps turn what was once a O(n!) problem into a O(n^2) problem. In this paper, meta-heuristic algorithms will be more thoroughly discussed, because they are designed to be general solvers that can be applied and adapted to various problems. Certain types of meta-heuristic algorithms include ones that are swarm-based, trajectory-based, or evolutionary [21].

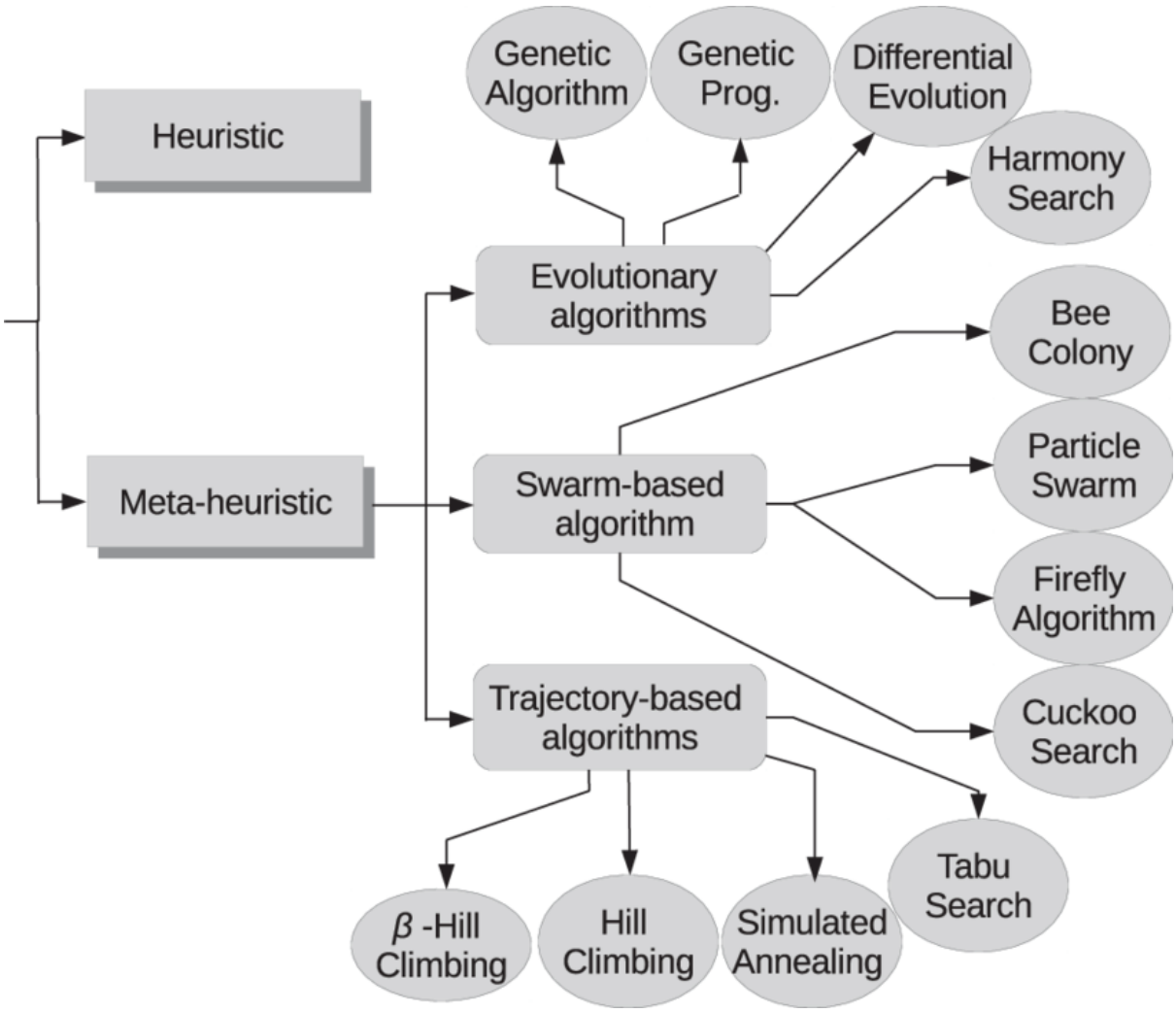

Figure 2.1: Breakdown of Optimization Algorithms

2.2.1 Evolutionary Algorithms

Evolutionary algorithms are a subsection of meta-heuristic algorithms inspired by real life biological processes, and they mimic the process of natural evolution where an individual in a certain population is affected by both the other individuals in the population and the environment. The overall fitness of the individual determines the chance that this individual will have offspring that will also inherit increased levels of fitness [23]. Over each generation, the result is a recombination of preferable characteristics that make up the entire population. Evolutionary algorithms have been applied to photonic design, with an example being two-dimensional photonic crystals with large bandgaps that saw a 12.5% increase in comparison to what humans were designing at the time [24]. Evolutionary algorithms were suitable for this case, because of the large and open ended design space that lacked smooth gradients [24].

One type of evolutionary algorithm is known as a swarm algorithm. Swarm-based algorithms differentiate themselves from the rest of the meta-heuristic algorithms by mimicking the behavior of a group of animals that are searching for food. Each iteration of this algorithm, solutions are formed, based on historical information that was obtained by prior generations. This makes swarm-based algorithms a type of genetic algorithm. Particle swarm has seen photonics applications in that it optimized 2x2 power splitters in a silicon photonic platform [25].

2.2.2 Trajectory Algorithms

Trajectory algorithms make up another subsection, and these algorithms are very commonly spread in not only photonics, but also in classical mechanics, robotics, and other forms of artificial intelligence [26]. The essence of a trajectory algorithm is that there is an initial solution, and the solution is adjusted by moving it towards a neighboring solution within this design space. An example of a trajectory algorithm is gradient descent, which is the backbone of machine learning algorithms. In photonics, trajectory algorithms have seen usage to describe the dynamics of quantum systems such as photonic crystals [27]. This optimization technique is very susceptible to being caught in a local minimum and not be able to find the global minimum, because oftentimes not all optimizable functions can be perfectly convex.

2.2.3 Adjoint-Based Optimization

Recently, adjoint optimization has seen a surge in usage due to its superior generality, degrees of design freedom, and computational efficiency, especially when compared to prior optimization methods [20]. The adjoint method computes the gradient of an objective function with respect to all degrees of design freedom using simply two full-field simulations [28]. The equation of the adjoint method is as follows:

$$\left(\frac{\delta \boldsymbol{M}}{\delta \psi(x)}\right)^{\dagger} \lambda(x) = \frac{\delta \boldsymbol{F}}{\delta \psi(x)} \tag{2.1}$$

M represents an optimization-specific constraint equation, ψ is a field that F is optimized relative to, † is the adjoint operator, and F is an objective functional. At this point, determining $\lambda(x)$ simply consists of one single solve, because the right hand side of the equation is just a number for each instance of $\psi(x)$. Inverting M can be tedious, however, if the constraint equations can be linear in $\psi(x)$, then this makes determining $\lambda(x)$ much easier.

The adjoint method has proven its usefulness to the point of it being integrated into a commercial FDTD solver that has been widely implemented for designs such as silicon-on-insulator (SOI) Y-splitters [28]. The design of Y-splitters as a result of adjoint-based optimization will be the primary focus of this study, as this method will be extended through the implementation of machine learning.

2.3 Machine Learning Algorithms

In recent years, machine learning (ML) has shown promise in solving inverse design problems alongside the aforementioned metaheuristic algorithms. For instance, ML methods have been used in recent years to design the cross section of silicon dioxide/titanium dioxide multi-layered core shell nanoparticles, and convolutional neural networks (CNNs) have been used to map physical geometries to spectral properties [29][30]. ML has been extremely useful in that it can provide solutions that lack intuition to most people, but the tradeoff is in its lack of transparency. ML has traditionally been seen as a ”black box” of sorts [31]. In a nutshell, the way that machine learning models work is in that they receive input data and output data and attempt to find the underlying relation between the input and output. For instance, one of the simplest machine learning methods

is known as the perceptron algorithm, which takes inspiration from biological neurons [32]. The perceptron algorithm is a simple supervised learning binary classifier in which there is a separable labelled dataset, and attempts to find the decision boundary hyperplane that perfectly classifies the two types of labelled data. The perceptron algorithm performs as follows:

Algorithm 1: Perceptron Algorithm

Data: $x = x_1, x_2 ... x_n \in R^n$, and where $y_i \in$ -1,1
$w_0 \leftarrow 0 \in R^n$;
for *each example in dataset* **do**
  if $y_{pred} = sgn(w^T x) \neq y_i$ **then**
    $w_{t+1} = w_t + r(y_i, x_i)$ where r is the learning rate ;
    go back to the 0th index example of the dataset (restart process);
  else
    if $y_{pred} = sgn(w^T x) = y_i$ **then**
      continue;
    end
  end
end
return *Final weight vector when the entire dataset is properly classified*

This algorithm is a fundamental building block for current day machine learning, because the neural network architectures that are used are also known as multi-layered perceptrons. Rather than just having a single perceptron working, there are many perceptrons that are organized into three different layers that act in conjunction with one another: the input layer, the hidden layer, and the output layer. A simple neural network architecture looks as follows: The input layer contains nodes that correspond to key features in a dataset, or the x vector that was mentioned previously. The hidden layer nodes help provide the proper weight for the vectors, and the output layer nodes ultimately perform the classification task. Unlike the single perception mentioned at the start, the artificial neural network typically does not use the heaviside step function as the activation function. Instead, functions such as the Rectified Linear Unit (ReLU), sigmoid (logistic curve), and tanh (hyperbolic tangent). The forms of all these activation functions are as follows:

$$f(x) = max(0, x) \tag{2.2}$$

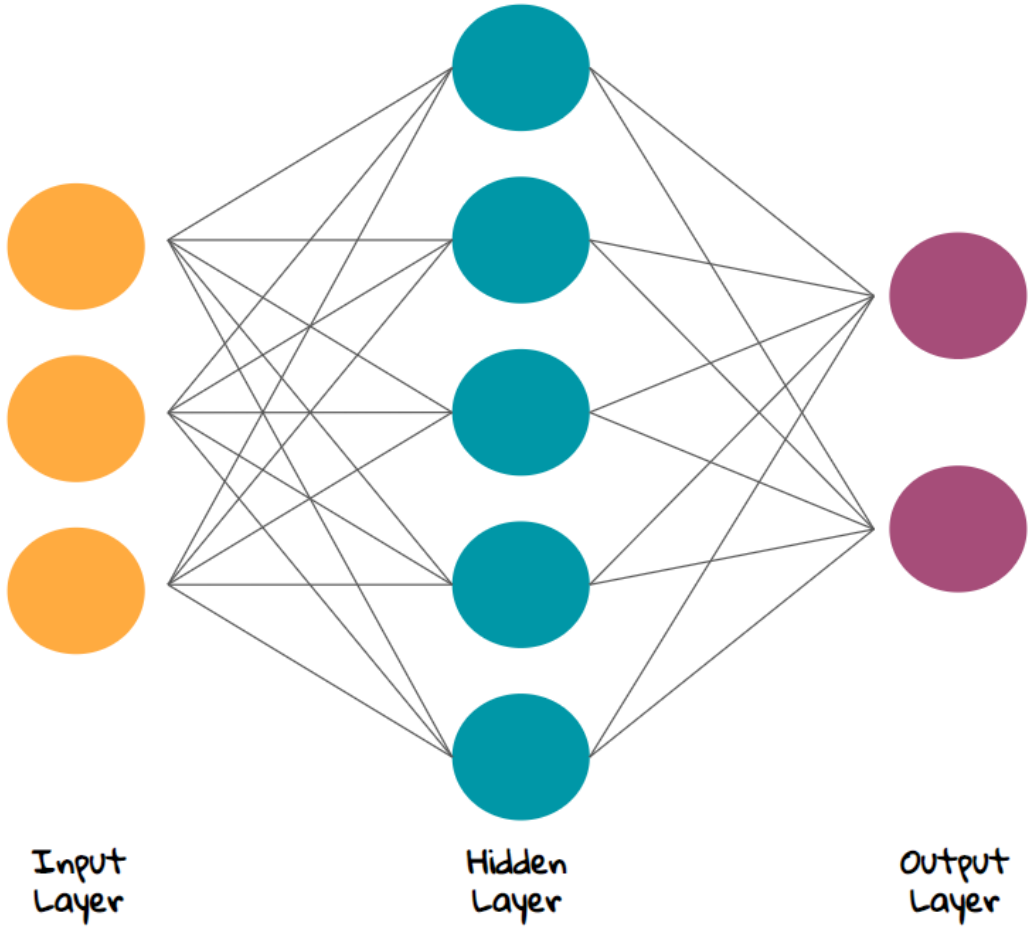

Figure 2.2: Vanilla Artificial Neural Network Architecture

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.4}$$

This artificial neural network architecture is very useful, however, this is not suitable for complex materials design, because that would require image recognition.

2.3.1 Convolutional Neural Networks

Within the umbrella of artificial neural networks, there is a class known as the convolutional neural network (CNN). CNNs have been proven to be extremely useful in analyzing images, and the name of this architecture derives from the convolution operation that is done rather than the usual matrix multiplication in one of the layers [33]. An example of a CNN architecture for an image from the MNIST (Modified National Institute of Standards and Technology) dataset is shown in the following: In this figure, the image of the digit zero is the input, and it gets passed in as a tensor

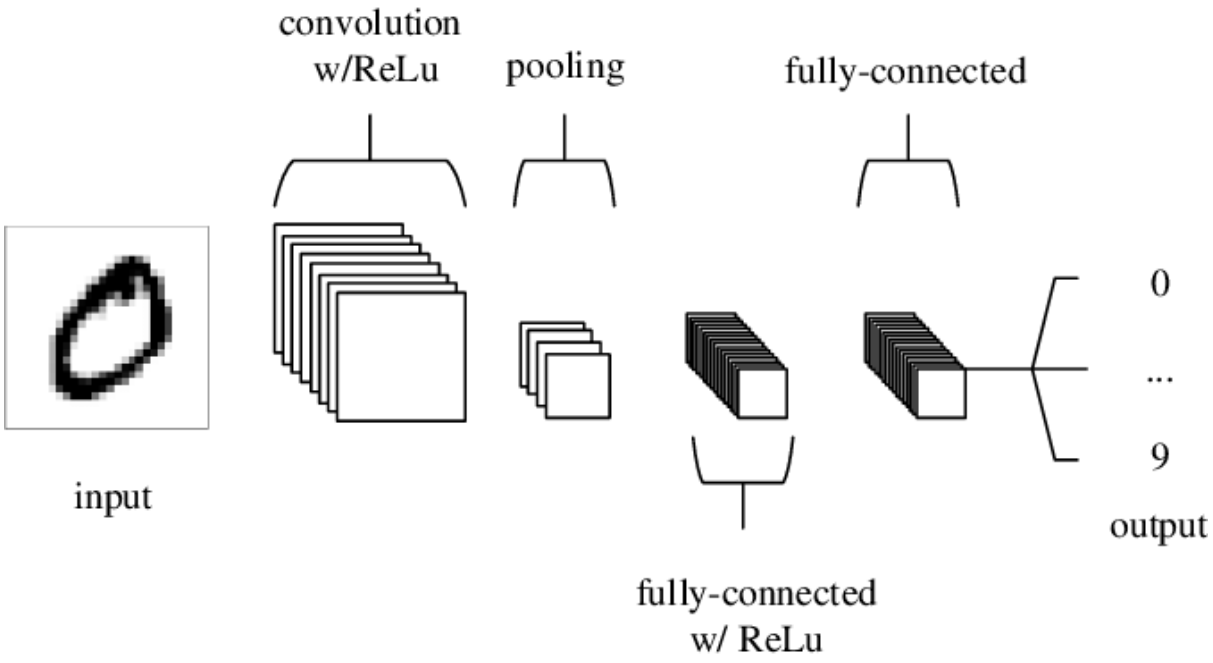

Figure 2.3: Convolutional Neural Network Architecture

with a shape that corresponds to: how many pixels tall, how many pixels wide, number of inputs, and number of channels.

Within the convolutional layer, the image, which is now represented as a tensor that is a certain number of pixels tall and wide, then has its pixels convolved. Filters are used, and they are applied across the data in a sliding window. For every sliding action, the process involves an element-wise multiplication and then summing these values. In short, the input is the image itself, and the result is a convolved feature, which is a typically a matrix of a lower dimensionality.

Next is the activation layer, which in this case uses ReLu. ReLu is used because it mitigates the 'vanishing gradient' problem because it has a constant gradient for all of its positive inputs [33].

The next layer is called the pooling layer. Pooling layers summarize the presence of features in pieces on the feature map that was already created as a result of the convolutional filter. Convolutional filter feature maps run into the issue where small movements in the position of features in the input image create a vastly different feature map. The pooling layer down samples, which essentially uses a lower resolution of the image and thus summarizes the presence of features. The pooling layer is typically 2x2 pixels with a stride of 2 pixels. What this means is that this layer will reduce the size of a feature map by 2, and thus the number of total pixels is quartered. There are two types of pooling: max and average. Max pooling takes the 2x2 and outputs the maximum value out of the 4 values, whereas average pooling aptly takes the average of the 4 values.

Lastly, there is the fully connected layer. This flattens the matrix that is created following the pooling layer, and then the values get summed up accordingly depending upon the weights of all the values. Another activation function is used, and then the CNN can create some sort of output, and in our MNIST case, the CNN can determine from a choice of 10 outputs what the handwritten input is.

This idea of using a CNN is pertinent to this study, because it is somewhat similar to the idea of a handwritten digit having a number 0 through 9 affiliated with it. In this study's case, the input image will be a device geometry, and then the affiliated value with the device geometry will be a numerical value that characterizes the device performance. Ultimately, CNNs will greatly enhance this study.

Chapter 3

CNN TRAINING OF ADJOINT BASED OPTIMIZATION STRUCTURES

3.1 Creation of Training Data

In order to use a CNN, one must first obtain a training dataset of images. This was done by performing adjoint based optimizations on a starting Y-splitter design. A widely implemented method of this adjoint optimization that is integrated into a common FDTD solver was used [28].

Figure 3.1: Initial Y-Splitter Geometry

The investigated system is defined as a silicon waveguide with silicon dioxide cladding, 220 nm thick, the black and white pixels represent the respective permittivities of silicon and silicon dioxide, and the optimizable geometry is the area between the input and output ports while keeping the port sizes themselves to be fixed. Each optimization was run at a different wavelength, varying from 1.3 μm to 1.8 μm in increments of 0.1 μm. The result of all of these optimizations is a large amount of device geometries with gradual performance increases with 6 different target wavelengths.

The performance of the device was shown via a specific figure of merit (FOM), and improvements correlated with a decreasing FOM. Each design consisted of simulations that were done in the forward direction (typical forward design) and in the backwards direction (adjoint/inverse design). The simulation would halt once the device geometry obtains an optimal FOM, and this would differ

for all of the target wavelengths (i.e. the simulation at 1.3 μm is not guaranteed to run for as many iterations as at 1.4 μm). This application of the adjoint simulations uses an FOM that represents the power coupling of the guided modes, and the equation is as follows:

$$FOM = \frac{1}{P(\lambda)} \int |T_0(\lambda) - T(\lambda)| d\lambda \tag{3.1}$$

In this equation, P is the source of power, T_0 is the ideal power transmission, T is the realized power transmission, and λ is the wavelength at which this simulation occurs. In essence, an optimal FOM is one that minimizes the difference of the power transmission of the input and output.

The result of the simulation once the FDTD solver determines the optimal geometry for a minimized FOM can look as follows:

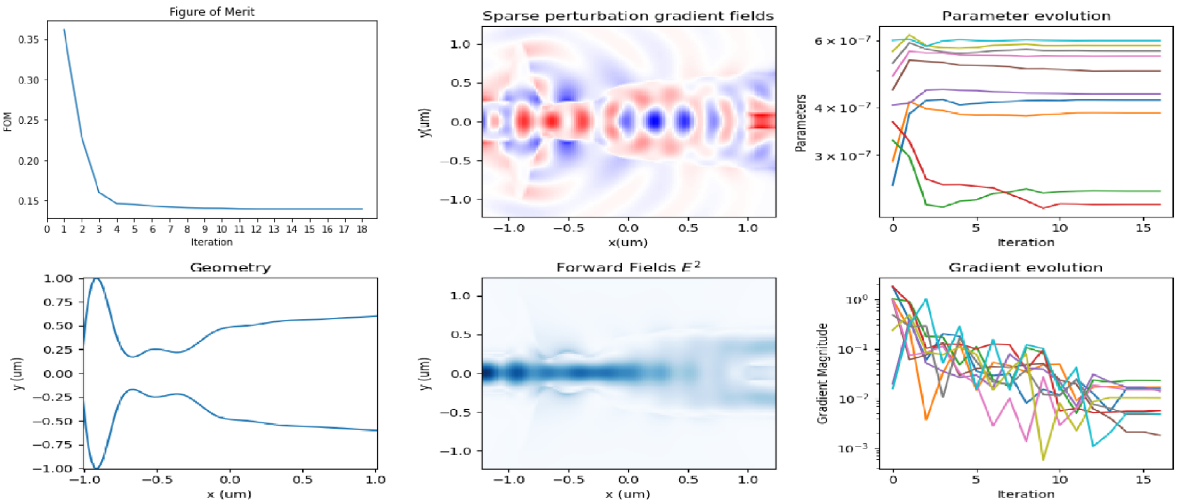

Figure 3.2: FDTD Optimization Report

This figure details the final device geometry, which is in this case the 18th, the sparse pertubation gradient fields, and the forward fields, along with the evolution of the FOM, the evolution of the parameters, and the evolution of the gradient. The initial FOM was approximately 0.36, and the final iteration was approximately 0.14. The simulation is programmed to halt once the percent difference between the final iteration and the one prior is less than 1E-5.

Unique geometries are achieved at each target wavelength, and as a result, the FOM also differs. The results of all the simulations performed at each target wavelength are as follows:

In Figure 3.3, the starting design is shown to be constant for all of the simulations, and even

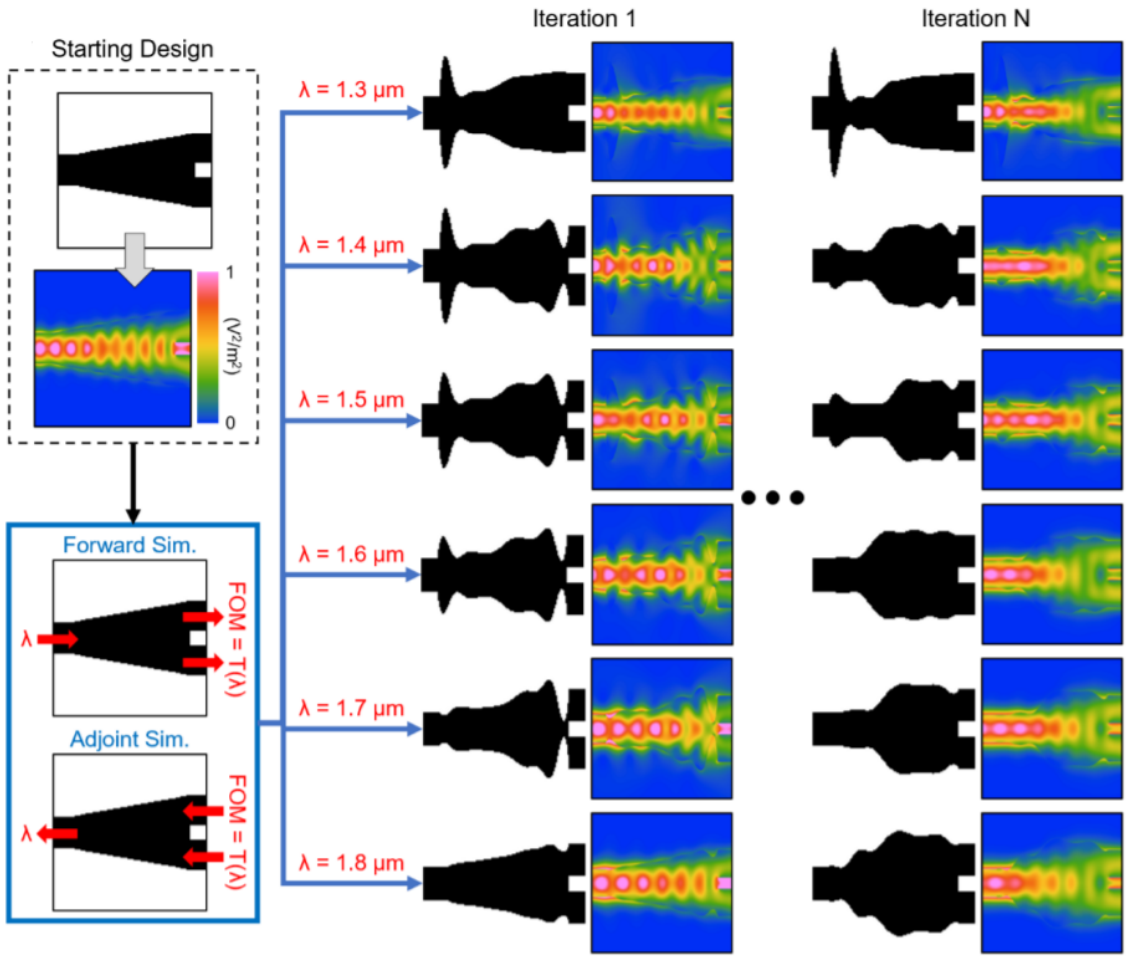

Figure 3.3: Final Device Geometries for Each Target Wavelength

after the first simulation the geometries are already distinct from one another. After N iterations, where N differs for each of the six geometries, the final geometries differ wildly, with the simulation for target wavelength 1.3 μm being thin and having spikes near the input, whereas the simulation for target wavelength 1.8 μm is thicker and does not have such sharp features.

In Figure 3.4, there is a noticeable trend where the higher performing device geometries (i.e. lower FOM values) also iterated longer for its respective simulation. Target wavelength = 1.3 μm, for instance, plateaus much earlier than any of the other 5 optimization runs. It is possible that for this target wavelength, the simulation ran into the issue that most gradient based algorithms have: local minima trapping due to a non ideal starting value during the optimization process. As a result, additional techniques may need to be used in order to further lower this figure of merit to the order of magnitude that target wavelength = 1.8 μm has.

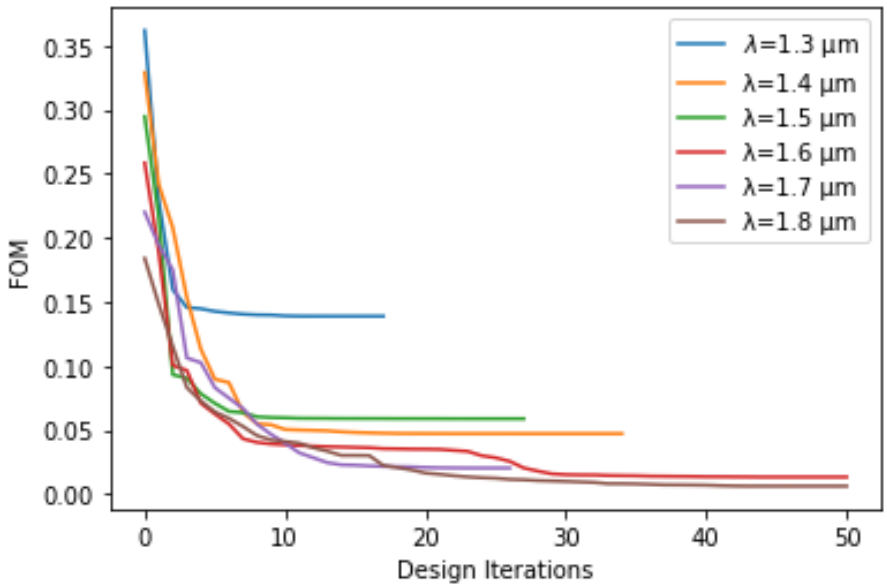

Figure 3.4: Figure of Merit vs Design Iteration Across All Optimization Runs

Target Wavelength (μm)	Figure of Merit	Number of Iterations
1.3	0.13889	17
1.4	0.047326	34
1.5	0.058932	27
1.6	0.013458	50
1.7	0.020481	26
1.8	0.0063360	50

Table 3.1: Final Figure of Merit Values For Each Target Wavelength

Table 3.1 shows the exact final FOM values alongside the exact number of iterations each simulation ran for. Target wavelength = 1.3 μm ran the least amount of times (17), and ended with an FOM = 0.13889, whereas Target wavelength = 1.8 μm ran the most amount of times (50) and ended with an FOM = 0.0063360.

3.2 CNN Problem Definition

Having all of the optimized structures means that there is now sufficient training data for a CNN to use in order to obtain a relation between the structure and the FOM. Because the initial geometry for each simulation is the same, those were left out of the training dataset in order to have strictly unique images. As mentioned previously, the target wavelength and FOM are coupled with one another with respect to the device geometry, thus one neural network was designed for an input

of one Y-Splitter geometry (128 x 64 pixels or 2.5 x 1.25 μm^2), and the output will be an FOM value (between 0 and 1) and a target wavelength value. This problem is simultaneously a regression (FOM) and classification (target wavelength) problem. This design space includes 192 input and output pairs created for the neural network. In order for the neural network to easily classify the target wavelength, they were all converted into categorical labels as a 6 x 1 vector. For instance, if the target wavelength was 1.3 μm, then its associated vector was $T_{1.3} = [1, 0, 0, 0, 0, 0]$. An additional way to state the target wavelength is simply $argmax(T_\lambda)$.

3.3 Autokeras

90% of the data was used for training, whereas 10 % of the data was used for validation. Rather than manually tuning hyperparameters through trial and error, a neural network architecture searcher known as Autokeras was used in this study. Autokeras uses image blocks that automate the deep learning process through testing differing models across various trials [34]. The optimal architecture was found following 17 trials, and the validation loss was $8.6 \times 10^{-}5$, and the training and validation losses both strongly converged towards one another, displaying neither overfitting nor underfitting. As a result, the neural architecture has likely found the relation between structure and device performance.

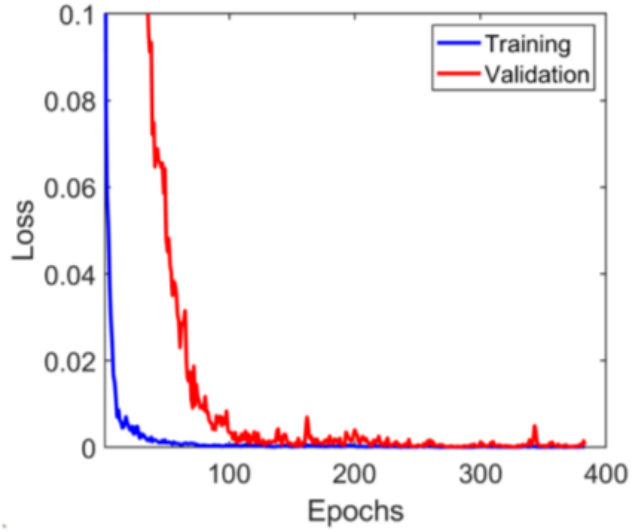

Figure 3.5: Training and Validation Loss of AutoML Optimized Architecture

3.4 Verifying CNN Results

The next step is in verifying the efficacy of the model in predicting target wavelength and FOM when given an image of a device geometry, which has its own set of 'ground truth' values that came from the FDTD solver. An example of the level of efficacy of this model is as follows:

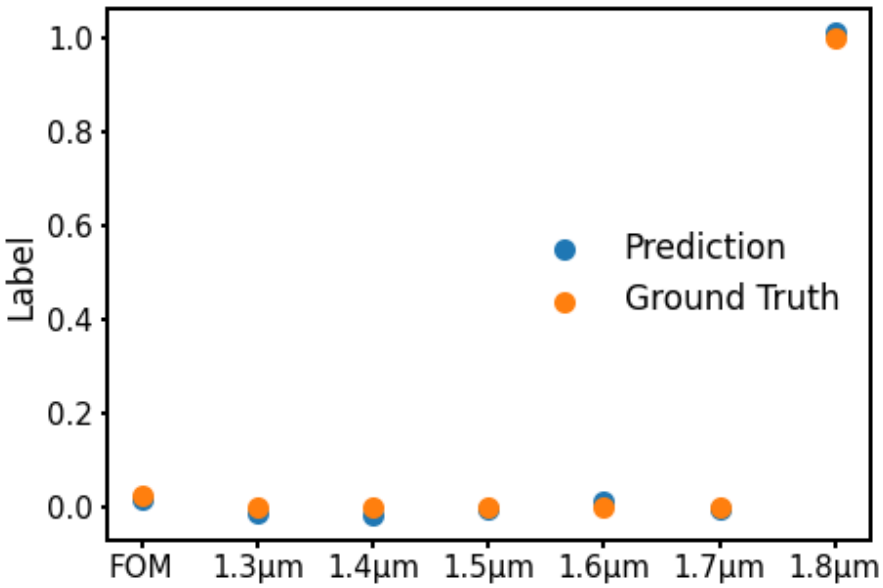

Figure 3.6: CNN Prediction Compared Against FDTD Simulation

Figure 3.6 simultaneously compares the regression (FOM) and classification (Target Wavelength) aspects of the model. For this one instance, it is clear that the CNN knows that it is looking at a device geometry optimized for 1.8 μm while also predicting an FOM that is very near the ground truth.

CNN Prediction	FDTD Simulation Ground Truth
0.01875806	0.0261579
-0.00986289	0
-0.01770346	0
-0.00149331	0
0.01118828	0
-0.00134203	0
1.0101404	1

Table 3.2: CNN Prediction Vector vs FDTD Ground Truth Vector

In knowing the exact values that the CNN outputs, it is then possible to calculate the mean

squared error (MSE). MSE is defined as follows:

$$MSE = \frac{1}{N}\sum_{i}^{N}(Y - \hat{Y})^2 \tag{3.2}$$

Where N is the number of sample data points (in this case N = 7), Y is the observed value (FDTD Simulation) and $\hat{Y}$ is the predicted value (CNN). For this particular example, the MSE = 9.96E-5. Thus, the CNN has likely learned the relationship between structural geometry and high performance.

Chapter 4

EXPLANATION ALGORITHM IMPLEMENTATION

4.1 Introduction to Explainable Artificial Intelligence

CNNs have proven to be incredibly useful, and in this instance seem to be capable of determining the relation between structure and performance, but at the end of the day, CNNs tend to be a 'black box' of sorts. That is to say, there is very little understanding as to how a CNN makes its predictions. It is incredibly difficult to fully grasp what a CNN is doing due to the fact that the deep learning process is an optimization of thousands to millions of weights and biases that derive from both the decided neural architecture, and also the training data itself [35]. This is a key limitation in ML, because it can be difficult to trust a model that lacks transparency. As a result, explainable artificial intelligence (XAI) has become a growing field that attempts to elucidate what is occurring in said black box. XAI is used in this study in order to unveil what the CNN has actually discovered as the relation between structure and performance, and use this knowledge to potentially further increase device performance.

4.2 Shapley Additive Explanations

In this study, the XAI of interest is known as Shapley Additive Explanations (SHAP). SHAP is a post-hoc explanation method that derives from game theory, and it attempts to explain features through a heat map in order to highlight the most relevant contributions [36]. Each feature in an image will have an associated SHAP value, and these SHAP values are calculated via the following equation:

$$\Phi_i = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'|! - 1)}{M!} (f_x(z') - f_x(z' \backslash i) \tag{4.1}$$

Φ_i is the SHAP value, $|z'|$ is the number of nonzero entries in z', x' is a simplified unit of the

original input x, M is the amount of simplified features, $f_x(z')$ is a model trained with that specific feature present, and $f_x(z' \backslash i)$ is a model trained without that specific feature present. Essentially, the SHAP algorithm withholds a specific feature, and then iterates throughout all the possible subsets for the image. By removing any specific feature, a change in the output is then calculated, which in turn allows for the algorithm to determine whether that specific feature contributes positively or negatively to the output.

In this implementation of SHAP, the heatmap uses red pixels to highlight features that contribute negatively towards device performance and blue pixels to highlight features that contribute positively towards device performance. These values range from -1 to 1. While the structures are predominantly blue, there are significant pockets of red that still exist in the structure that supposedly negatively impact the device performance. The device structure for target wavelength = $1.3\mu m$ seemingly has the most amount of red pixels, but this does make sense due to it also having the worst FOM of the 6 structures. While pockets of red pixels do exist within the center of the structures, the only pixels that will be manipulated will be those along any of the SOI boundaries due to the nature of the original FDTD simulations only manipulating the boundaries of the structures.

After having defined the blue pixels as a a positive contribution and red pixels as a negative contribution, a boundary extraction algorithm was created in order to adjust the shape of the device. The overall spirit of the algorithm was inspired specifically by the device with target wavelength = $1.3\mu m$. SHAP is seemingly suggesting that if the large spikes are removed from the device, then the structure should perform far better than what it is already capable of.

4.3 SHAP Redesigns

The next step is to use these explanations, and manipulate the images to decrease their red pixel content. An initial filtering process is implemented upon the SHAP images, and this helps idenfity the transition points from red to blue along the boundaries of the images. Next, a binarization function is used to convert the image to either show existing or non-existing elements, and the

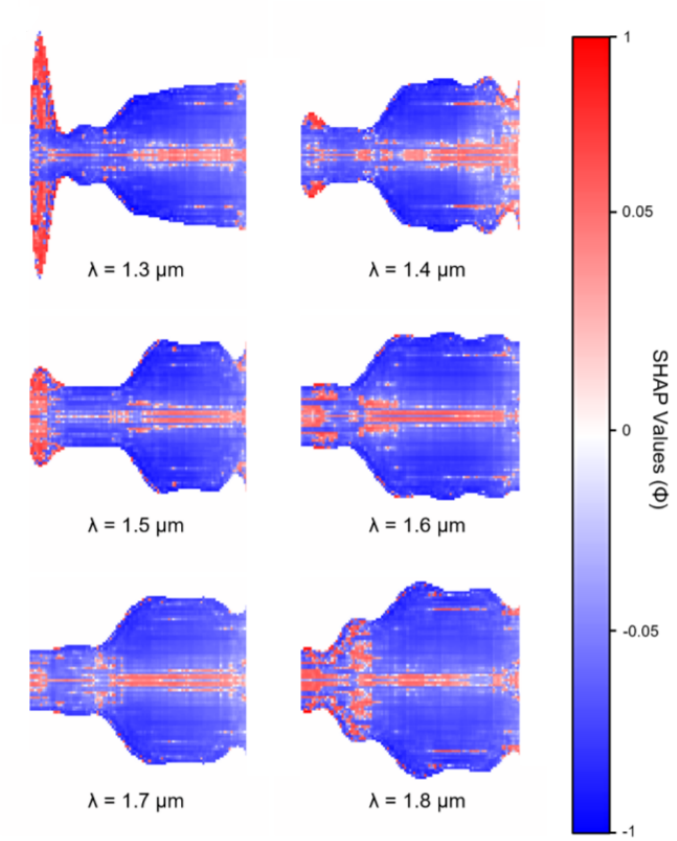

Figure 4.1: SHAP Values Across Various Designs

function is represented by a step function:

$$\rho(x,y) = \begin{cases} 1 & \text{for } \Phi(x,y) \leq 0 \\ 0 & \text{for } \Phi(x,y) > 0 \end{cases} \tag{4.2}$$

Whenever the function is equal to 1, this is representative of an existing element, whereas the function equalling 0 represents non-existing elements. Next, a boundary is drawn around the existing elements, and is represented by $\eta(x,y) = [X_i, Y_i]$. $X_i = [x_1, x_2, x_3, \ldots x_i]$ and $Y_i = [y_1, y_2, y_3, \ldots y_i]$ are both vectors of length i. Ultimately, i determines the resolution of the shape, and in order to keep this design to be feasibly fabricable, i is set to be 20. This spaces the points by 100 nm, which is within typical CMOS lithography resolution [37]. The manner in which the y-coordinates are determined will be described as follows (further details in Appendix).

The image was rastered from top to bottom across all the values in X_i, and then the algorithm

found the first instance of an existing element where $\rho(x,y) = 1$, and this is marked as a point to be added to Y_i. Furthermore, a hyperparameter denoted as α is implemented in the algorithm in order to improve robustness of the image, and mitigate the amount of sharp edges in the structure.

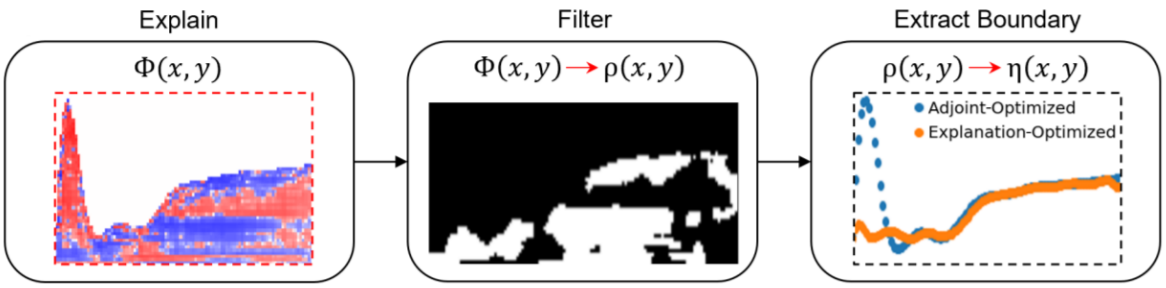

Figure 4.2: Workflow for Structure Redesign

Figure 4.2 is a schematic of the workflow where the SHAP values are first determined for an image in order to create a heatmap, next the image is filtered between existing and non-existing elements, and then the new explanation-optimized boundary is drawn and compared with the adjoint-optimized boundary.

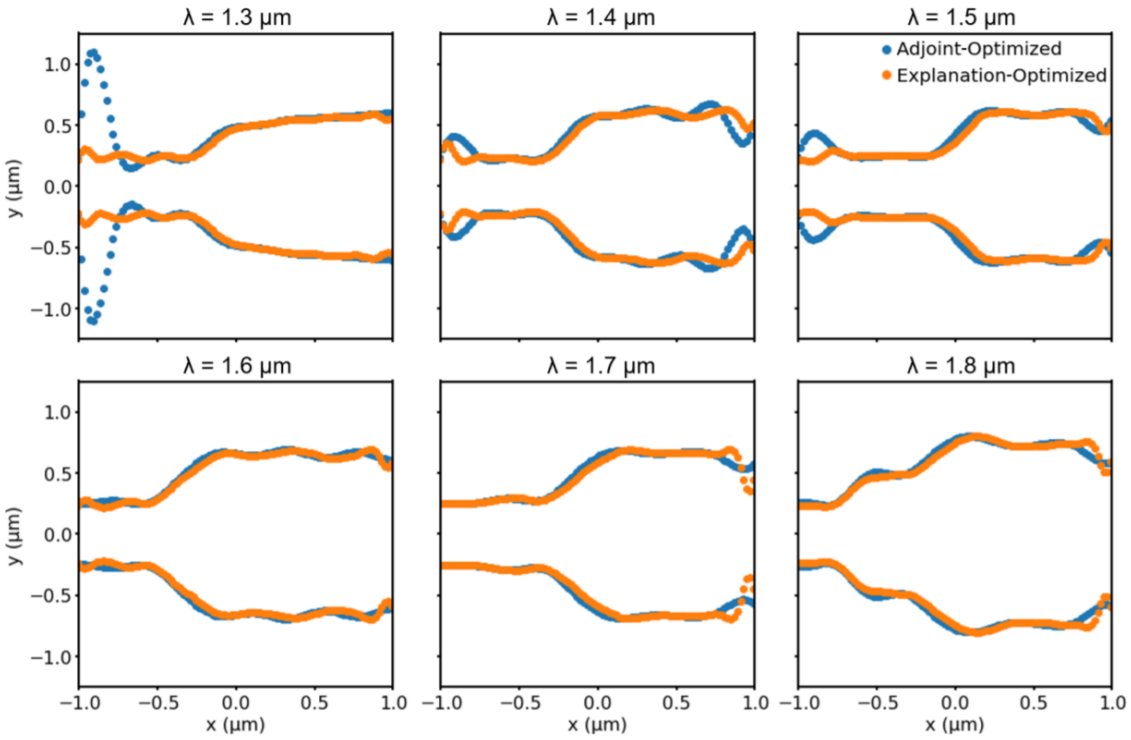

Figure 4.3: Explanation Optimized Structures for Each Target Wavelength

Figure 4.3 highlights all of the structures that are reoptimized due to the SHAP explanations.

There are subtle improvements for all the devices that had high performance, but it can be seen that for $\lambda = 1.3\mu m$, the large spike that was highlighted red by SHAP no longer exists following the explanation-optimized boundary finder. In fact, any sort of large extrusion near the input port is reduced down as a result of SHAP.

4.4 Simulating Explanation-Optimized Structures

After having obtained new device geometries for all of the target wavelengths, the natural progression would be to use these images as a starting point for another round of adjoint optimization in the FDTD solver. Depending on whether the removal of specific features results in a noticeable change in FOM, then it can be stated whether the CNN truly has learned the structure-performance relation and whether SHAP also pointed towards the correct features to remove or not.

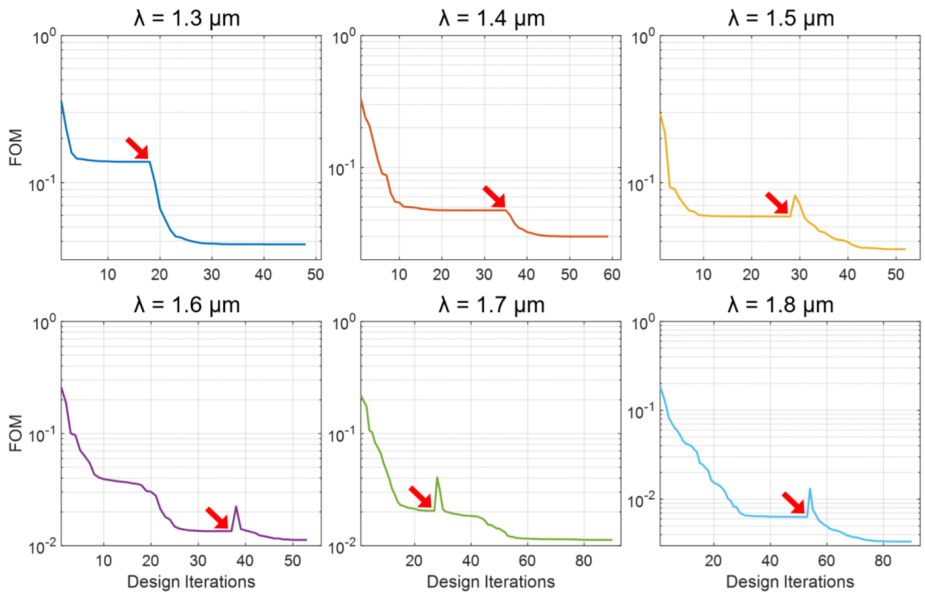

Figure 4.4: Two Stage Figure of Merit Optimization for Waveguide Geometry

Figure 4.4 displays the two stage optimization, with the red arrows in each optimization run being where the optimizations with the SHAP-explained images start. It can be seen that improvements occurred for all 6 of the target wavelengths, and the final FOM improved drastically for $\lambda = 1.3\mu m$.

Every optimization improved as a result of the initial starting geometry coming from the SHAP explanations. All of the FOM values improved by at least 15%, and for $\lambda = 1.3\mu m$, the FOM

Target Wavelength (μm)	Original Final FOM	New Final FOM	Percent Change
1.3	0.13889	0.038114	72.6
1.4	0.047326	0.029936	36.7
1.5	0.058932	0.035341	40.0
1.6	0.013458	0.011252	16.4
1.7	0.020481	0.011259	45.0
1.8	0.0063360	0.0033349	47.4

Table 4.1: Comparison of Original and New FOM Values

improved by nearly 73%. The fact that the first optimization improved by such a drastic amount suggests that the initial FDTD simulation was caught in a local minimum, and by changing the initial starting geometry, one can get out of this local minimum and improve the FOM. It is noted that this process only requires two optimization runs per target wavelength: the initial run and the SHAP explained run. This is far superior to other methods, which typically include repeatedly running optimizations with random starting points in hope of finding the optimal starting point. These methods can definitely take more than two optimization runs in order to discover the optimal starting point, whereas SHAP seemingly immediately uncovers this more optimal starting point.

Furthermore, these changes to the device geometry are targeted due to feature relevance (or lack thereof), and are not simply stochastic perturbations. In order to prove this, all of the first stage structures were all randomly modified (further explained in Appendix E) and the second stage optimization was then performed using these new structures. 5 random perturbations were made to each device geometry, and this results in 30 new test data points to compare with the SHAP reoptimized structures. None of the new 30 modified structures were able to obtain an FOM superior to that of SHAP's design, and thus the SHAP design is not simply stochastic. If anything, a stochastic perturbation can even drive towards an even worse FOM due to changing the starting location and driving the optimization even further into a local minimum that is even farther away from the true global minimum.

Next, this design process should also prove to be generalizable outside of this specific photonic design problem. Rather than simply study SOI waveguides, the goal is to then extend this problem to other materials. The decision to look into generalizing via materials selection stems from the fact that past XAI endeavors have already been studied for other nanophotonic structures, such as metasurfaces or cross resonators and were shown to be capable of tuning spectral properties

[38]. Thus, one possibility was to change the waveguide material system, and replace the silicon with silicon nitride. Silicon nitride is a material of interest for waveguides due to it being a low loss material that can handle high optical power and be used for both linear and nonlinear optical functions [39]. Thus, a 2 stage optimization was done with this setup where the Y-splitter geometry was kept constant, but instead silicon nitride is used (a number of optical constants will obviously change as a result). Nonetheless, the 2 stage optimization once again provided a final FOM that was superior to just the FDTD simulations (this is detailed further in Appendix F). Consequently, the shown design process is generalizable to other material classes for electromagnetic design using adjoint optimization.

Chapter 5

CONCLUSIONS

In conclusion, this study presents an inverse design framework that extends beyond the current capabilities of gradient and topology optimization in that it provides a succinct manner in which the effects of local minima trapping can be mitigated. This study combines adjoint optimization, convolutional neural networks, neural network architecture searchers, and explainable artifical intelligence in order to enhance the performance of Y-splitter devices power transmittance by an average of 43%. When using solely adjoint optimization, the algorithm does plateau, but SHAP explanations elucidate the features of the device geometry that contribute positively and negatively towards an optimal FOM. By altering the device according to the SHAP explanations, the adjoint optimization algorithm was able to break out of the local minimum it found, and approached a minimum much closer to that of the global minimum due to the improved FOM. This method in total only required two adjoint optimization simulations, which is potentially more efficient than other techniques that may randomly search for whatever the optimal starting point is for this sort of gradient-based algorithm. Hopefully, combining conventional optimization solvers with data driven approaches such as SHAP will enhance additional inverse design problems and lead to other discoveries about the relationship between structure and performance in photonics.

Appendices

Appendix A

FDTD SIMULATIONS

As mentioned in Chapter 1, the FDTD method is critical in determining electromagnetic behavior. The FDTD algorithm is cumbersome to display mathematically, but writing a script to perform the algorithm is instead quite feasible, and open source FDTD algorithms can easily be found to give simple working examples [40].

First, several constants need to be defined for the problem setup, such as Planck's constant, the speed of light, or even just the time lapse of the simulation.

```
%--------------------------------------------------------------------%
h=0.6582119514 ; % Planck's constant

dt=0.13; % time step in femto seconds

tmax=100; % maximum time limit

t=-20:dt:tmax; % time array

c=299.792458; % speed of light in nm/fs

S=1; % courant factor

dz=c*dt/S;
mu0=2.013354451e-4; % permeability of free space in (V fs^2/e nm)
ep0=55.26349597e-3; % permittivity of free space in (e / V nm)
```

Figure A.1: 1D FDTD Parameters

Other important values include free space permeability and permittivity, the speed of light, and the Courant factor, which is an important stability factor when solving partial differential equations numerically.

Next, a source should be defined, and in this case it is a Gaussian envelope, which is the product of a sinusoid and a decaying exponential function.

```
source=exp(-((t).^2)/(delta^2)).*cos(wl.*t); % E-field Gaussian envelope

%--------------------------------------


%---Gaussian envelope source----------
Zs=200; % position index of source

M=500; % no. of spatial grid points
Z=(0:M-1).*dz; % space axis
ep(1:M)=ep0; % permitivity array
mu(1:M)=mu0; % pemeability array
```

Figure A.2: E-Field Source Parameters

Constants for the PML boundary condition also need to be defined as shown below. This includes the conductivity array, the width of the layer, the allowed reflectivity, and other important values that represent the attenuation and wave decay.

```
%---- PML absorbing boundary condition----

sigma(1:M)=0; % initialize conductivity array
d=55; % width of PML layer
m=3; % polynomial order for grading sigma array (pp 292, Taflove)
neta=sqrt(mu0/ep0);
R=1e-8; % required reflectivity
sigma_max=-(m+1)*log(R)/(2*neta*d*dz);
Pright=((1:d+1)./d).^m*sigma_max;
sigma(M-d:M)=Pright; % lossy conductivity profile
sigma(1:d+1)=fliplr(Pright);

sigma_star(1:M)=sigma.*mu0./ep0; % Eq 7.8 Taflove, pp 275
%------------- PML constants ---------------------------------------%

A=((mu-0.5*dt*sigma_star)./(mu+0.5*dt*sigma_star));
B=(dt/dz)./(mu+0.5*dt*sigma_star);
C=((ep-0.5*dt*sigma)./(ep+0.5*dt*sigma));
D=(dt/dz)./(ep+0.5*dt*sigma);
```

Figure A.3: Perfectly Matched Layer Parameters

With this in mind, the actual FDTD algorithm can be built, and it can be seen that the E-fields and H-fields depend on one another as they are solved simultaneously in the leapfrog manner (the two fields are offset from another by one) throughout the duration of the defined time.

```
for n=1:length(t)
        Ex(Zs)=source(n); % insert source in certain space grid
        Hy(1:M-1)=A(1:M-1).*Hy(1:M-1)-B(1:M-1).*(Ex(2:M)-Ex(1:M-1));
        Ex(2:M-1)=C(2:M-1).*Ex(2:M-1)-D(2:M-1).*(Hy(2:M-1)-Hy(1:M-2));
        Ex(M)=Ex(M-1);
        figure(1)
        plot(Z,Ex)
        xlabel('Z in nm','fontSize',14);
        ylabel('E_x','fontSize',14);
        title('1D FDTD with PML','fontSize',14);
        axis([0 M*dz -1 1]);
        getframe();
end
```

Figure A.4: FDTD Function

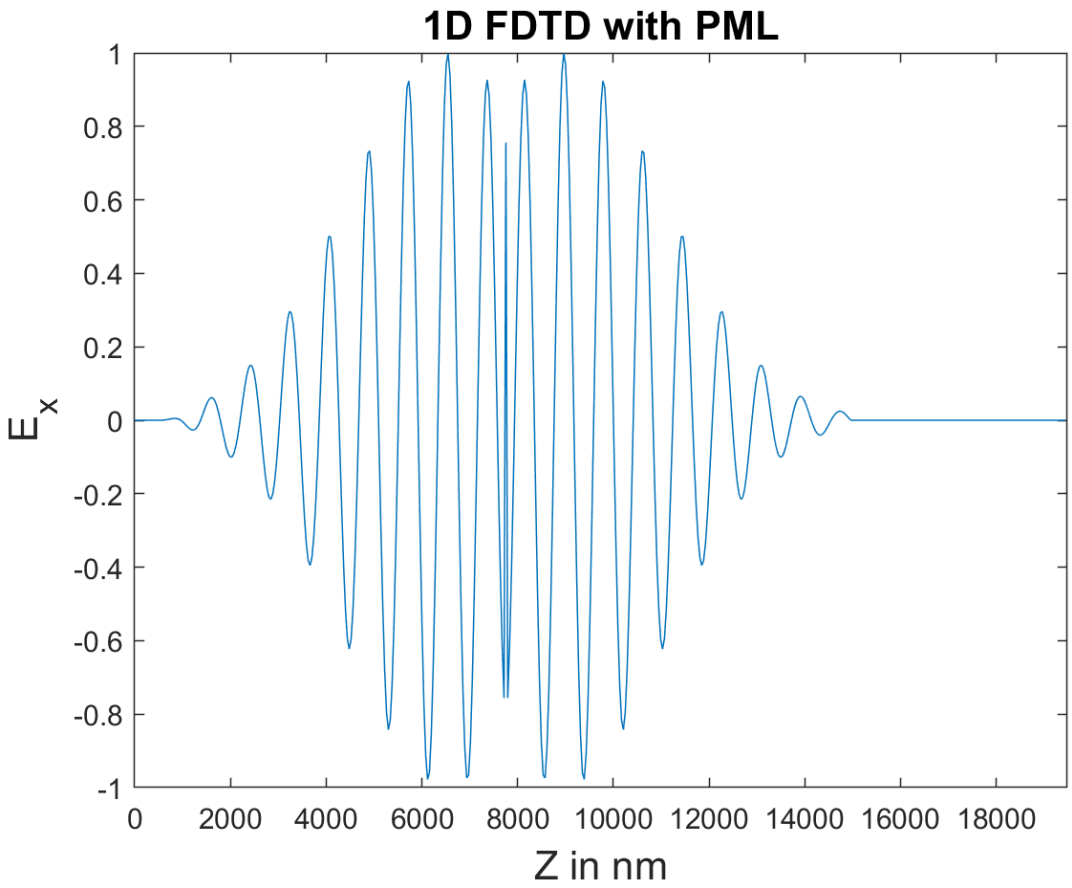

Figure A.5: 1D FDTD E-Field Exaxmple

Appendix B

DEVICE STRUCTURE GEOMETRY EVOLUTION

The following is an example of the entire evolution of device geometry throughout one specific adjoint optimization. The first image is shown to be 'iteration 0' because all of the optimizations start with this initial geometry, and the optimization ends when it is apparent that there is no further improvement to be had. And in showing the entire device evolution, the growing of the spike throughout the iterations can be seen. For this specific optimization, the FDTD solver ran until the 17th iteration.

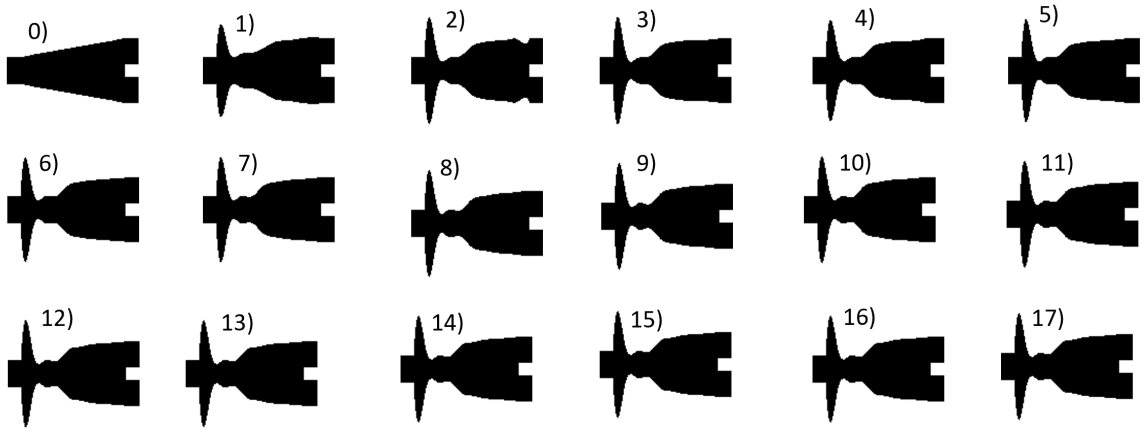

Figure B.1: Adjoint Optimization Structure Evolution Target Wavelength = 1.3 μm

Appendix C

AUTOKERAS IMPLEMENTATION

Rather than tuning hyperparameters in a neural network architecture manually, a neural architecture searcher known as Autokeras was used, and it did so using image blocks in hopes of finding the optimal blocks to output the 7 point vector (6 points associated with classifying the target wavelength and 1 point regression for the FOM). The optimal architecture was determined after 17 trials, which had a validation loss equal to $8.6 \times 10^{-}5$. The two separate losses follow one another relatively well, thus neither underfitting nor overfitting is in play.

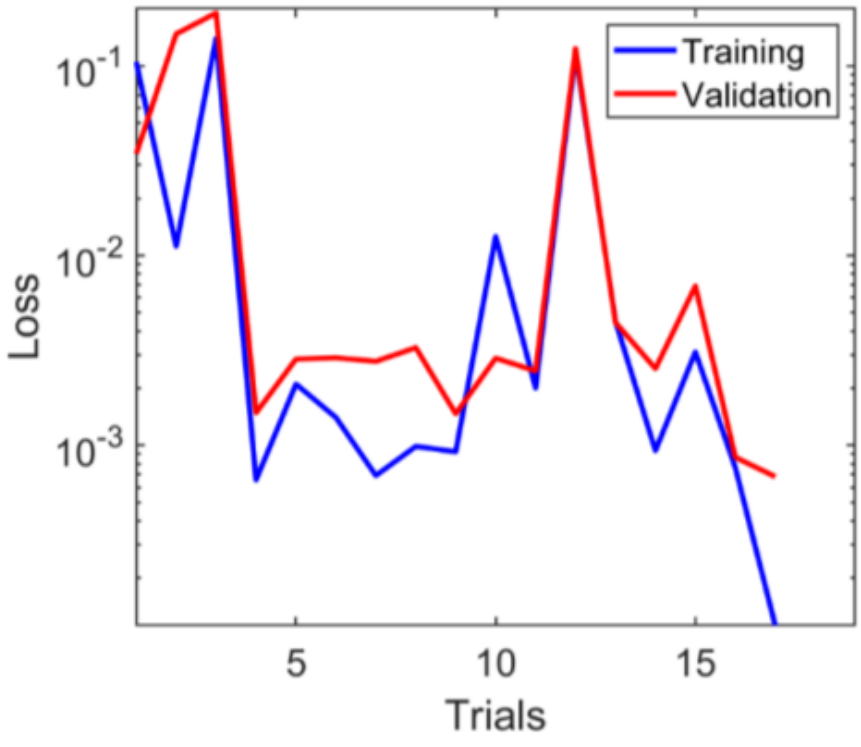

Figure C.1: Training and Validation Losses with respect to Autokeras Trials

The image block progression is also shown, where the very first trial uses two convolutional blocks with 32 and 64 filters and an initial loss of $3.4 \times 10^{-}2$. The final trial improved the loss by 3 orders of magnitude by using 4 convolutional blocks with 512,64,32, and 32 filters. The activation

function was always leaky ReLu, and batch normalization and max pooling layers were kept as constant.

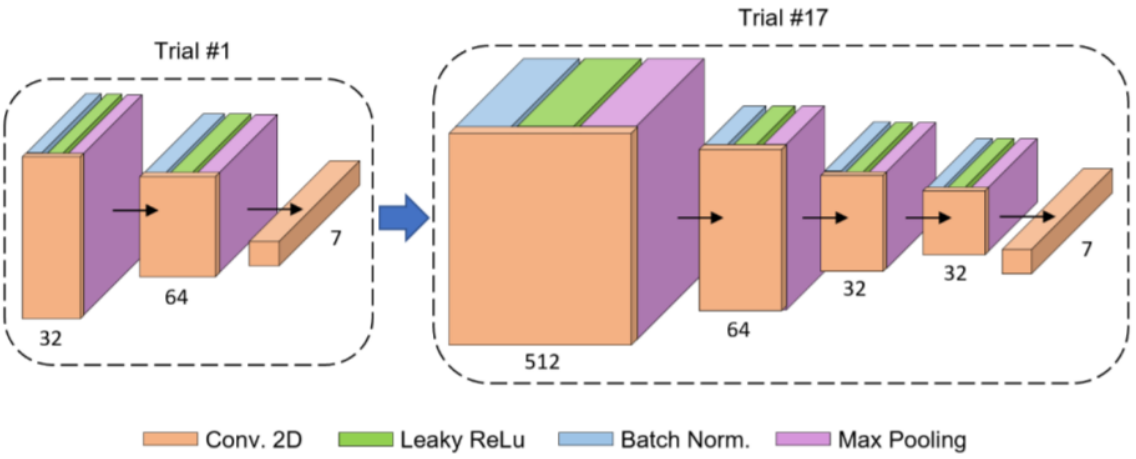

Figure C.2: Autokeras Image Block/Model Architecture Evolution

Appendix D

BOUNDARY FINDER ALGORITHM

The SHAP explanations created a heatmap that determined what features contributed positively to the FOM, as well as the features that contributed negatively. A boundary algorithm was created that took advantage of this fact, and accordingly adjusted the shape of the device. The algorithm performs as follows:

Algorithm 2: Boundary Finder Algorithm

Data: Filtered image array P with pixel values $\rho(x, y)$
for $x \in X_i$ **do**
  for *y = 0:L, where L is the vertical length of the image* **do**
    if $\rho(x, y) = 1$ & $y_i - y_{i-1} < \alpha$, *where* α *is a hyperparameter that enhances image robustness* **then**
      Append(y_i) to list Y_i;
    else
      Append(y_{i-1}) to list Y_i ;
    end
  end
end

This algorithm searches through the image and finds the first instance of an existing element within that column, and adds this to the array of y-point values that are used to form the new boundary of the device geometry. The hyperparameter ensures that there are minimal sharp features in the device.

Appendix E

DEVICE RANDOM PERTURBATIONS

To ensure that SHAP is learning something about the fundamental geometry of the device, and not simply arbitrarily highlighting features as positive or negative, an effort was made to modify the initial device geometries, and then rerun the FDTD solver optimization in order to compare the results of random perturbations and SHAP's explanations. 5 random modifications were done to each of the 6 optimization tests, meaning that there are 30 new data points in total to compare against SHAP. The 5 modifications decision comes from the final array of y-values for the design is of length 10. 2 values get changed at a time for an array of length 10, thus yielding 5 modifications for one design at one specific target wavelength. The values in the y-array are modified by adding 300 nm.

Algorithm 3: Random Perturbations Algorithm

```
Data: Design parameters (Y-values) of all the final design optimizations
for i ∈ Design Parameters do
    New Array = Copy of Design Parameters Array;
    for j = 0:length of design parameters array/2 do
        New Array[2j]= New Array[2j]+3e-7;
        New Array[2j+1]= New Array[2j+1]+3e-7;
        Append(New Array) to list Random Changes;
        New Array = Copy of Design Parameters Array;
    end
end
```

The randomly modified structures always produced FOMs higher than that which SHAP provided. The majority of the new random structures also yielded FOM values that were higher than the initial optimized design, thus showing that randomly perturbing the device geometry can not only be incapable of finding the optimal starting point, but also push towards a far inferior starting point.

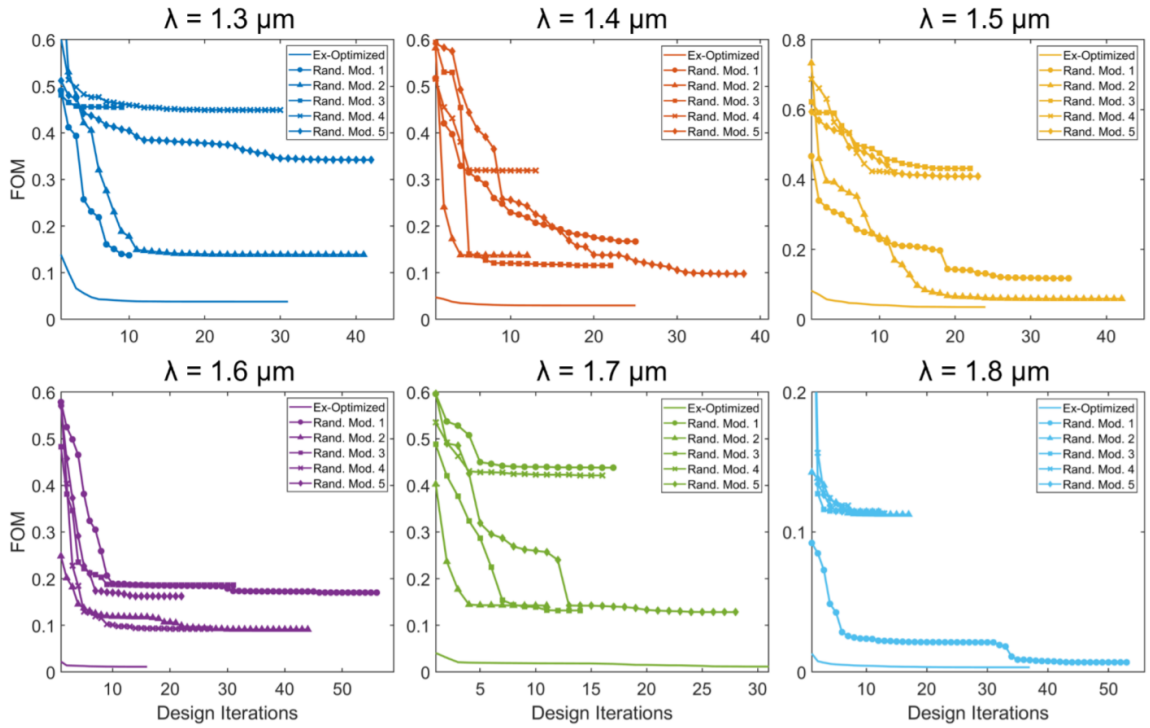

Figure E.1: Comparison of Random Perturbation vs. SHAP

Appendix F

SILICON NITRIDE WAVEGUIDE OPTIMIZATION

The generalizability of this study is also of concern, thus it was expanded towards silicon nitride waveguides as well (the main focus of this research is on SOI) and the Y-splitter geometry was kept constant. The two-step optimization was done, with the red arrow in the figure below marking where the second optimization starts, and the second stage of the optimization following SHAP provided lower FOMs for each target wavelength. Thus, this approach is generalizable for multiple applications that require the adjoint method in optimization.

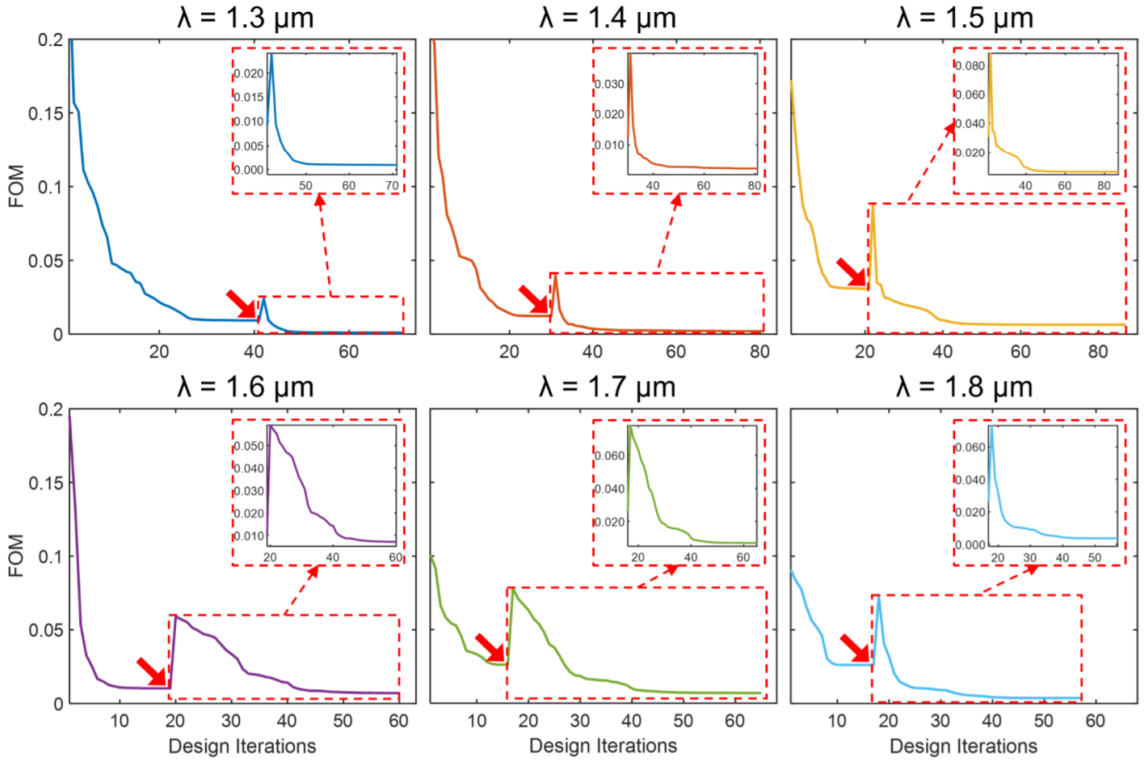

Figure F.1: Silicon Nitride Two Stage Optimization

REFERENCES

[1]I. Amiri, S. Bin Azzhuri, M. Jalil, H. Hairi, J. Ali, B. M., and Y. P., "Introduction to photonics: principles and the most recent applications of microstructures", Micromachines **9**, 492 (2018).

[2]R. Ma, R. Oulton, V. Sorger, and M. Zhang, "Plasmon lasers: coherent light source at molecular scales", Laser & Photonics Review **7**, 1–21 (2012).

[3]S. Li, S. Tan, L. C., E. Waffenschmidt, S. Hui, and T. C., "A survey, classification, and critical review of light-emitting diode drivers", IEEE Transactions on Power Electronics **31**, 1503–1516 (2016).

[4]E. Singh, K. Kim, G. Yeom, and H. Nalwa, "Atomically thin-layered molybdenum disulfide (mos2) for bulk-heterojunction solar cells", ACS Applied Materials % Interfaces **9**, 3223–3245 (2017).

[5]C. Caloz and T. Itoh, *Electromagnetic metamaterials: transmission line theory and microwave applications* (John Wiley & Sons, 2005).

[6]R. Isaac, "The future of cmos technology", IBM Journal of Research and Development **44**, 369–378 (2000).

[7]C. Li, D. Liu, and D. Dai, "Multimode silicon photonics", Nanophotonics **8**, 227–247 (2019).

[8]D. Sullivan, *Electromagnetic simulation using the fdtd method* (John Wiley & Sons, 2013).

[9]J. Schneider, "Understanding the finite-difference time-domain method", School of electrical engineering and computer science Washington State University **28** (2010).

[10]J. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves", Journal of computational physics **114**, 185–200 (1994).

[11]A. Oskooi, L. Zhang, Y. Avniel, and S. Johnson, "The failure of perfectly matched layers, and towards their redemption by adiabatic absorbers", Optics Express **16**, 11376–11392 (2008).

[12]E. Bécache, S. Fauqueux, and P. Joly, "Stability of perfectly matched layers, group velocities and anisotropic waves", Journal of Computational Physics **188**, 399–433 (2003).

[13]C. Varin, R. Emms, G. Bart, T. Fennel, and T. Brabec, "Explicit formulation of second and third order optical nonlinearity in the fdtd framework", Computer physics communications **222**, 70–83 (2018).

[14]D. Hockanson, J. Drewniak, T. Hubing, and T. Van Doren, "Fdtd modeling of thin wires for simulating common-mode radiation from structures with attached cables", in Proceedings of international symposium on electromagnetic compatibility (IEEE, 1995), pp. 168–173.

[15]Z. Chen, W. Fan, and S. Yang, "Towards the wave-equation based explicit fdtd method without numerical instability", in 2016 ieee international conference on computational electromagnetics (iccem) (IEEE, 2016), pp. 265–267.

[16]J. Ball, "The calculus of variations and materials science", Quarterly of Applied Mathematics **56**, 719–740 (1998).

[17]J. Gray, "The calculus of variations", in *Change and variations* (Springer, 2021), pp. 81–93.

[18]B. Offrein, G. Bona, R. Germann, I. Massarek, D. Erni, et al., “A very short planar silica spot-size converter using a nonperiodic segmented waveguide”, Journal of Lightwave Technology **16**, 1680 (1998).

[19]D. Dobson and S. Cox, “Maximizing band gaps in two-dimensional photonic crystals”, SIAM Journal on Applied Mathematics **59**, 2108–2120 (1999).

[20]S. Molesky, Z. Lin, A. Piggott, W. Jin, J. Vucković, and A. Rodriguez, “Inverse design in nanophotonics”, Nature Photonics **12**, 659–670 (2018).

[21]N. Kokash, “An introduction to heuristic algorithms”, Department of Informatics and Telecommunications, 1–8 (2005).

[22]M. Flood, “The traveling-salesman problem”, Operations research **4**, 61–75 (1956).

[23]M. Majig and M. Fukushima, “Adaptive fitness function for evolutionary algorithm and its applications”, in International conference on informatics education and research for knowledge-circulating society (icks 2008) (2008), pp. 119–124.

[24]S. Preble, M. Lipson, and H. Lipson, “Two-dimensional photonic crystals designed by evolutionary algorithms”, Applied Physics Letters **86**, 061111 (2005).

[25]J. Mak, C. Sideris, J. Jeong, A. Hajimiri, and J. Poon, “Binary particle swarm optimized 2× 2 power splitters in a standard foundry silicon photonic platform”, Optics letters **41**, 3868–3871 (2016).

[26]A. Malik, T. Henderson, and R. Prazenica, “Trajectory generation for a multibody robotic system using the product of exponentials formulation”, in Aiaa scitech 2021 forum (2021), p. 2016.

[27]M. Biondi, G. Blatter, H. Türeci, and S. Schmidt, "Nonequilibrium gas-liquid transition in the driven-dissipative photonic lattice", Physical Review A **96**, 043809 (2017).

[28]C. Lalau-Keraly, S. Bhargava, O. Miller, and E. Yablonovitch, "Adjoint shape optimization applied to electromagnetic design", Optics express **21**, 21693–21701 (2013).

[29]J. Peurifoy, Y. Shen, L. Jing, Y. Yang, F. Cano-Renteria, B. DeLacy, J. Joannopoulos, M. Tegmark, and M. Soljačić, "Nanophotonic particle simulation and inverse design using artificial neural networks", Science advances **4**, eaar4206 (2018).

[30]C. Yeung, J. Tsai, B. King, Y. Kawagoe, D. Ho, M. Knight, and A. Raman, "Elucidating the behavior of nanophotonic structures through explainable machine learning algorithms", ACS Photonics **7**, 2309–2318 (2020).

[31]J. Olden and D. Jackson, "Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks", Ecological modelling **154**, 135–150 (2002).

[32]F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.", Psychological review **65**, 386 (1958).

[33]S. Albawi, T. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network", in 2017 international conference on engineering and technology (icet) (Ieee, 2017), pp. 1–6.

[34]H. Jin, Q. Song, and X. Hu, "Auto-keras: efficient neural architecture search with network morphism", arXiv preprint arXiv:1806.10282 **5** (2018).

[35]S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural networks", arXiv preprint arXiv:1506.02626 (2015).

[36]S. Lundberg and S. Lee, "A unified approach to interpreting model predictions", in Proceedings of the 31st international conference on neural information processing systems (2017), pp. 4768–4777.

[37]Y. Taur, D. A. Buchanan, W. Chen, D. J. Frank, K. E. Ismail, S.-H. Lo, G. A. Sai-Halasz, R. G. Viswanathan, H.-J. Wann, S. J. Wind, et al., "Cmos scaling into the nanometer regime", Proceedings of the IEEE **85**, 486–504 (1997).

[38]C. Yeung, J. Tsai, B. King, B. Pham, D. Ho, J. Liang, M. Knight, and A. Raman, "Multiplexed supercell metasurface design and optimization with tandem residual networks", Nanophotonics **10**, 1133–1143 (2021).

[39]R. Baets, A. Subramanian, S. Clemmen, B. Kuyken, P. Bienstman, N. Le Thomas, G. Roelkens, D. Van Thourhout, P. Helin, and S. Severi, "Silicon photonics: silicon nitride versus silicon-on-insulator", in Optical fiber communication conference (Optical Society of America, 2016), Th3J–1.

[40]S. Rao, *1d finite difference time domain simulation (fdtd) with perfectly matched layer (pml)*, (2015) (visited on 09/30/2021).