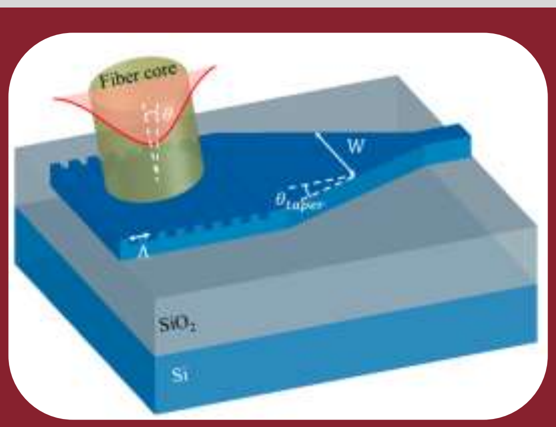

Optimizing Grating Couplers

Archimedes Li, archimedes.li@outlook.com
Westview High School, Class of 2022
USC Viterbi Department of Electrical Engineering, Prof. Sideris, SHINE 2021

Introduction

In order to transfer light from free space to a waveguide, a photonic device such as a grating coupler is necessary (Fig 1). A grating coupler consists of a long silicon waveguide with multiple teeth of varying lengths. By changing the width of each tooth, we can change how the light wave refracts. Our lab works to optimize these grate widths such that it outputs the most light onto the waveguide.

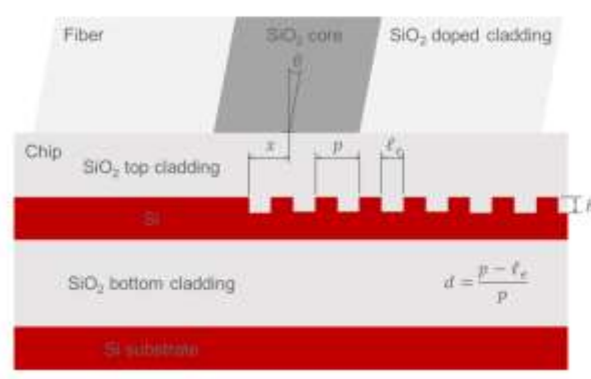

Fig 1. Grating Coupler Diagram

Simulating the Grating Couplers

In order to calculate the output power of the couplers, we used a simulation program called Lumerical, which utilizes the finite difference method. The program (Fig 2) allows the user to create structures that can interact with simulated light waves and measure their power and intensity at various locations. We used materials and measurements from previous tests and research as a starting point.

The grating coupler consisted of a long waveguide with 19 teeth with customizable lengths, resulting in 39 total parameters. When the simulation was run, a beam of light would be projected down onto the middle of the coupler and the path and energy of the waves would be predicted. The shape of the electric field around the grating could result in a straight beam along the waveguide (Fig 3.1) or random noise (Fig 3.2); hence why it was important to find the right combination of widths.

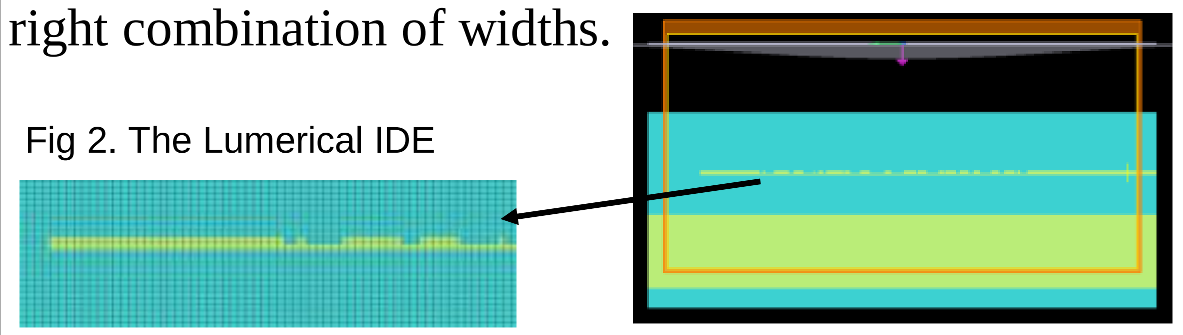

Fig 2. The Lumerical IDE

Gathering the Data

In order to gather data efficiently, we needed to automate the virtual construction of the grating coupler and the running of the simulation. This was accomplished with a script, which executed the following:

1. Delete the existing coupler.
2. Randomly generate widths for the 39 new grate segments.
3. Create a coupler with said lengths in the simulation area.
4. Run the simulation and save the output power.

We ran this script on a loop over 70,000 times throughout the course of SHINE.

In Fig 4, we can see the distribution of the power is clearly skewed; most of the random configurations resulted in an output of less than 2%, while only a few gave a higher output. This revealed the difficulty and scarcity of finding a special configuration that effectively transfers the energy.

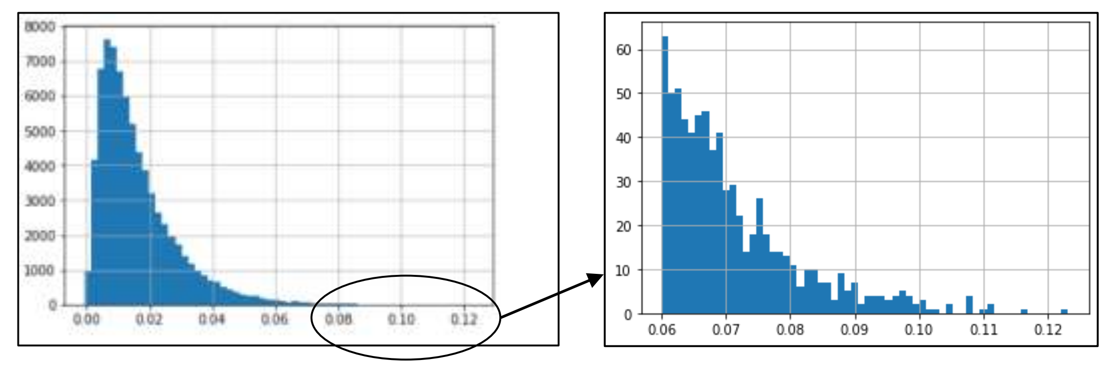

Fig 4. Power Distribution Graph

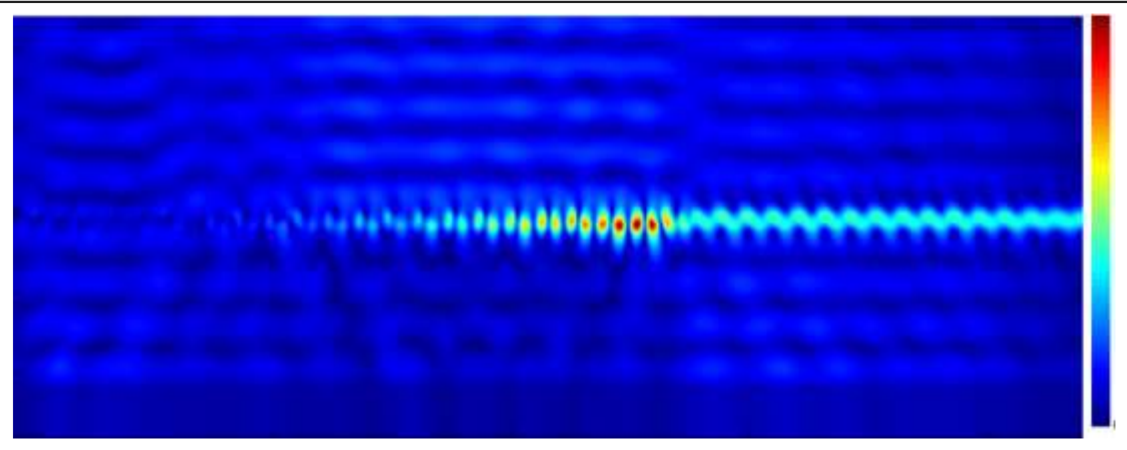

Fig 3.1. Electric Field of a High Power Grating Coupler

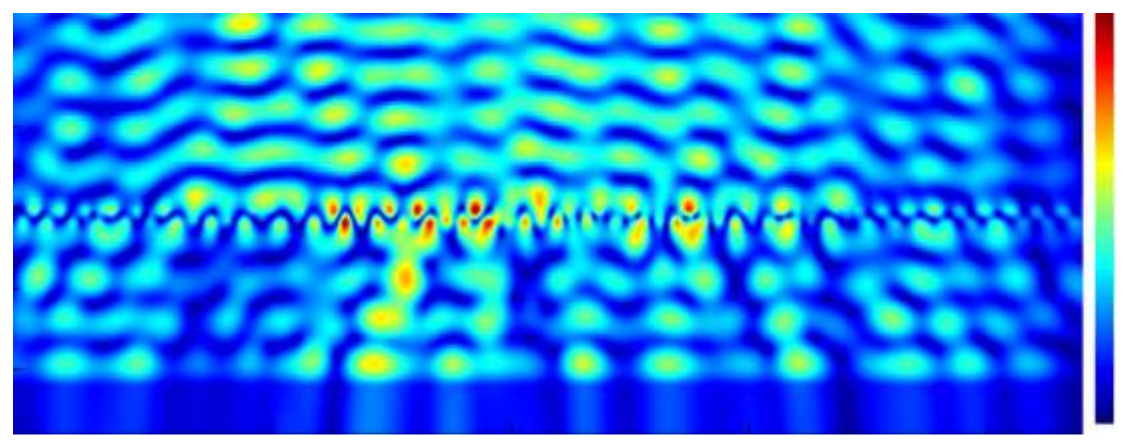

Fig 3.2. Electric Field of a Low Power Grating Coupler

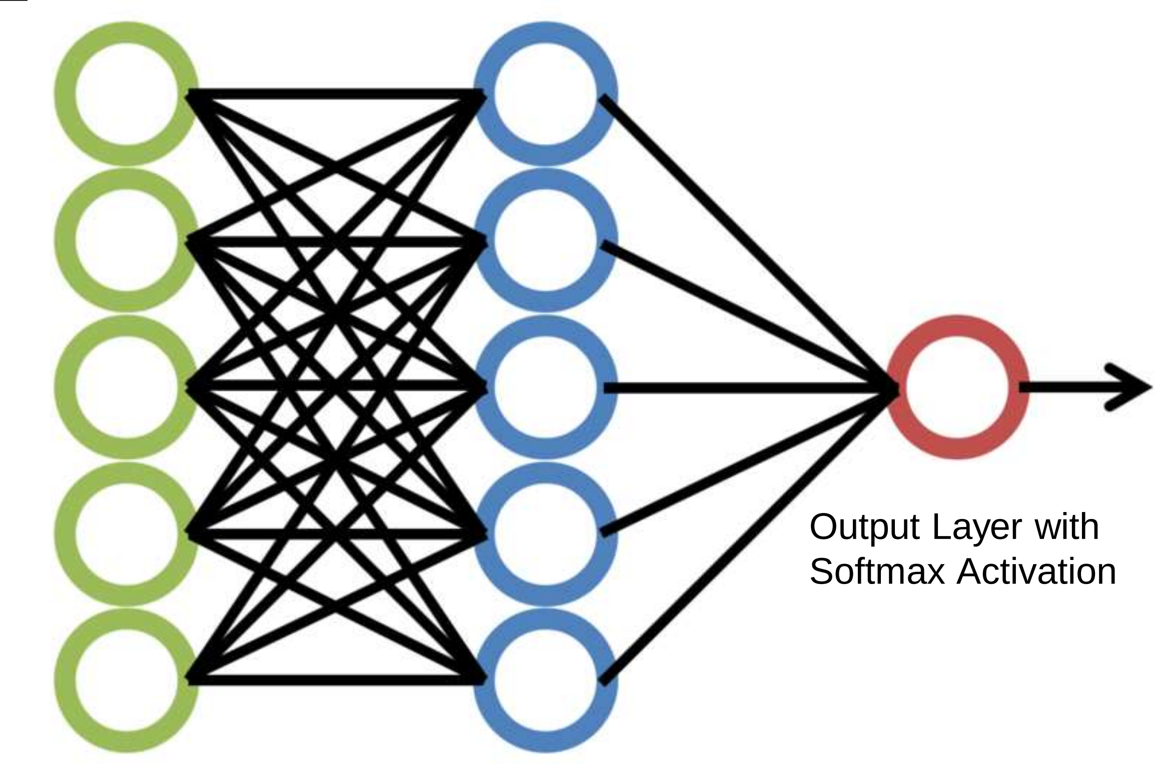

Fig 5. Neural Network Diagram

Machine Learning Model

The process of optimizing the grating widths would require an astronomical amount of direct simulations on Lumerical. With each simulation taking almost a minute to run, it can be impractical to optimize it directly. A remedy to this is to create a machine learning (ML) model to mimic the simulation.

We initially tried three different machine learning models: a random forest, linear regression, and a neural network. We used Python modules Tensorflow and Scikit Learn. Unfortunately, these models all had a very low accuracy, due to the fact that there were 39 different features over a complex power function. We decided to stick with a neural network because of its good reputation for dealing with complex problems like this (Fig 5).

In order to improve the accuracy, we tried a couple of different things. First, we converted the features from teeth widths to their absolute position. This allowed for us to remove the last feature which would make the models slightly better.

Furthermore, we decided to switch the model to a classifier, which had a lot more room for error. The classifier model output a probability for the occurrence of each class, which we seeked to optimize instead of directly maximizing the output power.

Tools Used

Gradient Descent Optimization

After obtaining a model for the behavior of the grating coupler, we just needed to optimize it. We applied a gradient descent algorithm to maximize the prediction probabilities of the "high output" class in the ML model. What gradient descent does is it essentially follows the slope of the function that it is optimizing (Fig 6). This allows it to reach a peak and thus a local maximum. The logic behind this was that a grating pattern that the ML model recognized as "high output" was more likely to yield a high output power.

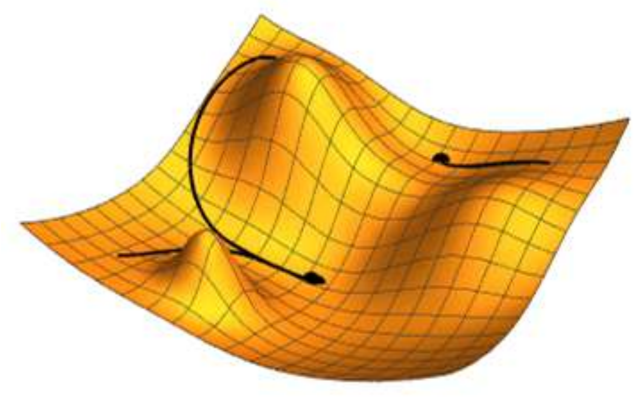

Fig 6. Example of Gradient Descent

Acknowledgements

Thank you to Professor Sideris, my mentor Emmanuel Garza, my center mentor Emily Yamanaka, and the program organizer, Katie Mills.