Efficient Python-Based Transfer Matrix Method for Multilayer Optics: Analytical and Computational Optimizations

Ananthanarayanan, Aniruth
Texas Academy of Math and Science

Acknowledgment of Major Assistance

I would like to acknowledge Dr. Guru Khalsa for teaching me his research methodologies, introducing me to optics, and reviewing part of my manuscript.

1. Introduction

The field of electromagnetics optics aims to effectively model how light interacts with various media by approximating a photon as a wave composed of electrical and magnetic fields. Classical electromagnetics centers around Maxwell's equations, a set of coupled partial differential equations (PDE) that describes light by defining the electric and magnetic fields in terms of each other [1]. By analytically solving Maxwell's equations, physicists can retrieve field solutions for the electric and magnetic fields to describe the propagation of waves through media. From here we can derive various estimations for linear homogeneous isotropic (LHI) media, diagonally anisotropic media, etc [2]. But analytical methods begin to fall short as the introduction of the 2nd and 3rd dimensions along with anisotropy requires solving several multiple complex eigenvalue problems, which are not feasible by hand. In addition, the complex geometries of real-world devices make analytical solutions difficult or impossible.

Computational electromagnetics (CEM) uses numerical approximations, semi-analytical reformulations of Maxwell's equations, and high throughput computing techniques to evaluate relationships between boundary conditions, allowing us to better understand the optical properties of materials. In this context, the transfer matrix method (TMM) emerged as a powerful computational tool by enabling physicists to model photon propagation through multilayer devices, facilitating the calculation of reflection and transmission coefficients.

In this paper, we hypothesize that the transfer matrix method (TMM) can be optimized by combining theoretical analytical reformulations of the four equations used to relate the longitudinal fields in a one-dimensional TMM with various computational optimization techniques. To test this, a TMM algorithm was developed and benchmarked against other implementations, showing an increase in speed of at least an order of magnitude. This algorithm opens up new possibilities because of its significantly decreased runtime and its pure Python implementation makes it easy to integrate with advanced reinforced learning techniques and neural networks to select more optimal materials for thin film coatings.

Additionally, we propose a potential supervised learning approach to optimize multilayer anti-reflective coatings in solar panels, to reduce energy loss [3]. One of the key challenges in developing more efficient solar panels is the reflection of sunlight, which reduces the amount of energy the wiring within the solar panel can absorb. Anti-reflective coatings are applied on solar panels to reduce the amount of energy that is lost. Still, their effectiveness and practicality are dependent on composition, thickness, and the price of the materials that make the coating.

2. Methodology

2.1 Materials

All code was written in Python, and tested on a Lenovo ThinkPad laptop with a CORE i7 vPro 8th Gen CPU on max performance mode.

2.2 Simple Transfer Matrix Method

The simple TMM can be broken into a chain of different steps that takes the parameters of a multilayer device as an input and outputs the reflectance and transmission, as shown in Figure 1. For the sake of simplicity, this paper will only cover a one-dimensional TMM.

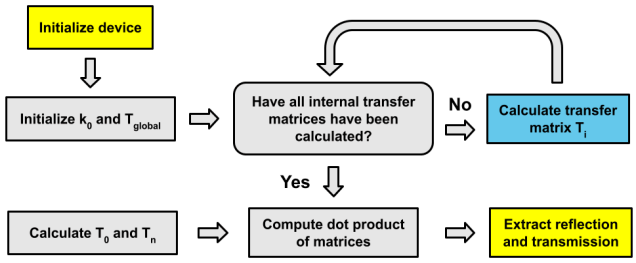

Figure 1: Breaking down the TMM algorithm into multiple steps.

2.2.1 Initializing Device Parameters

The first step in the TMM is initializing the device parameters. This involves defining the number of layers, along with the thickness and refractive index of each layer. The refractive index, n, is a complex number where n_r represents the real part (related to the phase velocity of light in the medium) and κ represents the imaginary part (denoting the material's absorptivity, or extinction coefficient). The refractive index typically varies with the wavelength of light, a factor that plays a critical role in determining the optical properties of the device.

2.2.2 Initializing k_0 and the Global Transfer Matrix

When simulating the propagation of an electromagnetic wave, the k vector describes the wave's direction and magnitude. It is essential in solving wave equations and calculating reflection and transmission coefficients. Specifically, we can calculate the scalar value k_0, which scales how fast a wave propagates through the media using the following formula

$$k_0 = \frac{2\pi}{\lambda}$$

where λ represents the wavelength.

We then initialize the global transfer matrix by calculating the transfer matrices for edge media. We can do this by solving the wave equation for the boundary conditions z=0 and z=L in a device of length L.

$$M_{z=0} = \begin{bmatrix} 1 & 1 \\ n_0 \cos\theta & -n_0 \cos\theta \end{bmatrix}$$

$$M_{z=L} = \begin{bmatrix} 1 & 1 \\ n_L \cos\left(\arcsin\frac{n_0 \sin\theta}{n_L}\right) & -n_L \cos\left(\arcsin\frac{n_0 \sin(\theta)}{n_L}\right) \end{bmatrix}$$

Simplifying the second equation for the $M_{z=L}$ matrix like so results in an expression that is faster to compute:

$$M_{z=L} = \begin{bmatrix} 1 & 1 \\ n_L\sqrt{(1-(\frac{n_0 \sin\theta}{n_L}))(1+(\frac{n_0 \sin\theta}{n_L}))} & -n_L\sqrt{(1-(\frac{n_0 \sin\theta}{n_L}))(1+(\frac{n_0 \sin\theta}{n_L}))} \end{bmatrix}$$

$$M_{z=L} = \begin{bmatrix} 1 & 1 \\ n_L\sqrt{1-(\frac{n_0 \sin\theta}{n_L})^2} & -n_L\sqrt{1-(\frac{n_0 \sin\theta}{n_L})^2} \end{bmatrix}$$

$$M_{z=L} = \begin{bmatrix} 1 & 1 \\ n_L\sqrt{\frac{n_L^2}{n_L^2} - \frac{n_0^2 \sin^2\theta}{n_L^2}} & -n_L\sqrt{\frac{n_L^2}{n_L^2} - \frac{n_0^2 \sin^2\theta}{n_L^2}} \end{bmatrix}$$

$$M_{z=L} = \begin{bmatrix} 1 & 1 \\ n_L\sqrt{\frac{1}{n_L^2} \cdot n_L^2 - n_0^2 \sin^2\theta} & -n_L\sqrt{\frac{1}{n_L^2} \cdot n_L^2 - n_0^2 \sin^2\theta} \end{bmatrix}$$

$$M_{z=L} = \begin{bmatrix} 1 & 1 \\ \sqrt{n_L^2 - n_0^2 \sin^2\theta} & -\sqrt{n_L^2 - n_0^2 \sin^2\theta} \end{bmatrix}$$

We then have all the components necessary to formulate the global transfer matrix from which we will extract the reflection and transmission coefficients. The global transfer matrix is the dot product of the transfer matrices of each layer and the boundary matrices.

$$M_{global} = M_{z=0} M_{internal} M_{z=L}$$
$$M_{internal} = M_1 M_1 \ldots M_{L-1}$$

2.2.3 Calculating the Simple 2x2 Transfer Matrix for a Single Layer

Solving the wave equation at the boundary conditions within the system of equations as follows, we can derive the 2x2 transfer matrix for a single layer as a function of theta and the refractive index of that layer. For the sake of brevity, I have included a slide from one of EMPossible's lectures that describes the fundamental matrix equations that govern the TMM simulation [4]. These are derived from Maxwell's equations, which have been simplified to one dimension.

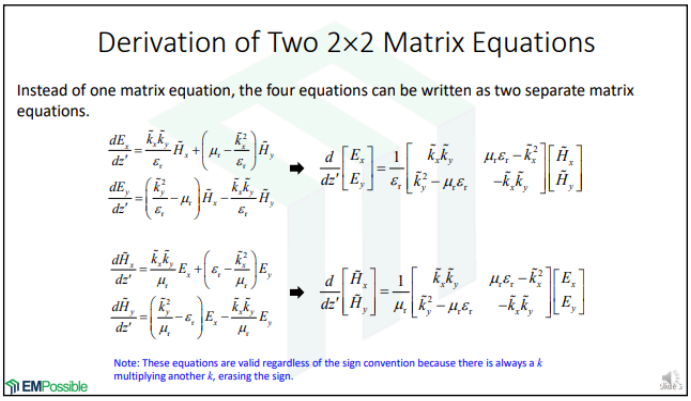

Figure 2: Derivation of the 2x2 matrix equations from Maxwell's equations (Rumpf *Computational Electromagnetics*)

From here we can obtain a solution to the following wave differential equation.

$$\frac{d\Psi}{dz'} - \Psi(z') = 0$$
$$\Psi(z') = We^{\Omega z'}W^{-1}$$

Then, solving the equation that defines the transfer matrix gives us the definition for the simple transfer matrix in terms of the refractive index.

$$E(d) = M(d)E(0)$$
$$M(d) = \begin{bmatrix} \cos k_z\theta & \frac{i\sin k_z d}{n_i \cos\theta} \\ in_i \sin(k_z d)\cos(\theta) & \cos k_z\theta \end{bmatrix}$$

2.2.4 Extracting Reflection and Transmission

Extracting the reflection and transmission coefficients is as easy as dividing and multiplying elements of the global transfer matrix once it is completely calculated. The reflection and transmission coefficients are defined as follows:

$$r = \frac{M_{total_{21}}}{M_{total_{11}}} \qquad t = \frac{1}{M_{total_{11}}}$$
$$R = |r|^2 \qquad T = |t|^2 \frac{n_i cos(\theta_i)}{n_1 cos(\theta))}$$

2.3 Scattering Matrix TMM

In certain cases, transfer matrices become unstable. This is a result of the eigenmodes (the columns of the eigenvector W in this case) being unsorted. The only way around this is by constructing the Poynting vector and rearranging the columns. However, we can avoid this, and other edge-cases, all altogether by working with scattering matrices.

Scattering matrices function similarly to transfer matrices in that they relate the input and output boundary conditions, but they do so differently. The elements of scattering matrices directly correlate with a physical phenomenon, a direct result of their definition. However, this also means that we cannot simply use the dot product to compile the various scattering matrices. Instead, we must use the Redheffer star product, a function designed specifically for multiplying two scattering matrices against each other while retaining their connection to physical phenomena [4]. Though this may seem like it results in overhead, the Redheffer star product is easily parallelized and as a result, the function does not add too much runtime. In practice, this is a reasonable tradeoff for the stability of scattering matrices.

2.4 Half-Infinite Gap Media Layer Isolation Optimization

Because they share boundary conditions, the scattering matrices/transfer matrices of each layer are dependent on the respective matrices of the layers before and after. Because of this, we have to calculate each layer irrespective of whether or not the refractive index and thickness are repeated. This is very inefficient as most multilayer devices are periodic. We can take advantage of this periodicity by isolating each layer with what is known as half-infinite gap media (which I will refer to as HGMLI) [4]. As illustrated in Figure 3, this isolates each layer and allows us to reuse transfer matrices, resulting in a decrease in redundant calculations.

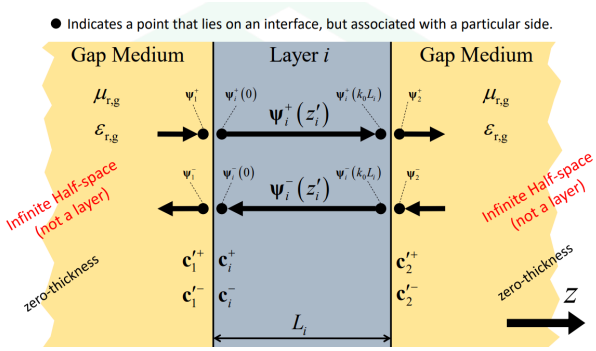

Figure 3: An example of half-infinite gap media isolating the layer i
(Rumpf *Computational Electromagnetics*)

There is another benefit to isolating each layer: parallelization. Since each layer is now independent, we can queue up multiple threads at a time to simultaneously perform the necessary

calculations. Then, we can perform a series of dot products or Redheffer star products to calculate the final global matrix. This yields a significant runtime improvement on multicore CPUs. Another significant side-effect of the HGMLI optimization is that we can store previously calculated matrices in a hashmap to avoid having to calculate them again. This, in combination with parallelization, almost eliminates all redundant calculations. However, a significant runtime decrease is not observed until you reach structures with a large amount of internal layers.

3. Results and Discussion

3.1 STMM and rcwa TMMSolver Comparison

The scattering matrix transfer matrix (STMM) method takes the simple transfer matrix's computational capability further by introducing gap media optimizations and the PQ method. These introductions increase stability and decrease the computations necessary to calculate the global transfer matrix. In combination with standard programming optimization techniques such as memoization, this results in a significant performance boost. To quantify this performance difference, the program was contrasted with the rcwa library's TMMSolver class. Our optimized TMM method performed significantly better than the rcwa library's TMMSolver. The performance improvement was at least an order of magnitude for each data point. Each data point in Figure 4 is the average runtime of the algorithm for 100 calculations with the number of internal layers. This was done to decrease the amount of noise from background processes running on the computer.

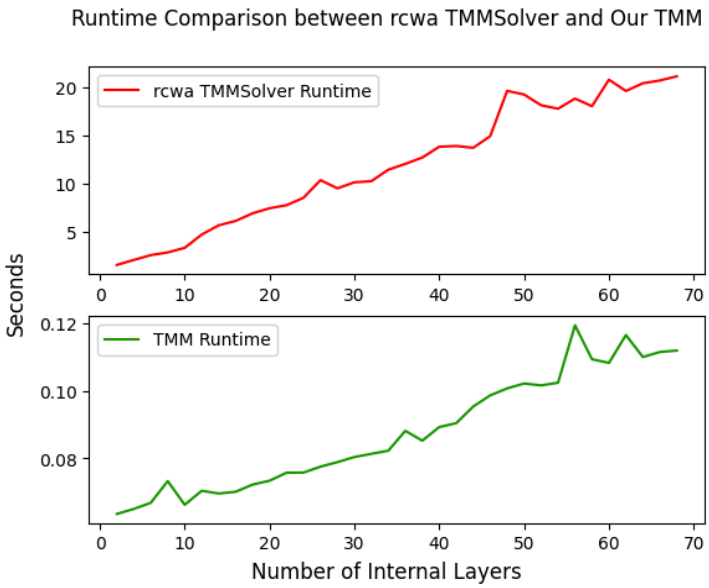

Figure 4: A graph of the rcwa TMMSolver runtime in comparison to our custom TMM implementation

We should also note that the accuracy of the program was 100%. It lined up exactly with the calculations of the rcwa library's method as well as the data in refractive-index.info.

3.2 Refractive Index as a Function of Wavelength

In our previous simulations, we have assumed that our sample materials have constant refractive indices. However, this is an extreme approximation, as almost all materials' refractive indices are a function of wavelength. Integrating non-linearity is essential to making parameter sweeps over wavelength more accurate to real-world data, allowing us to more precisely model real-world data. To do this, rather than passing in a number for the refractive index of a layer within the device, we passed in a function that took wavelength as a parameter and returned a complex refractive index. To ensure we hadn't broken anything, we attempted to replicate graphs from refractive-index.info using the Shkondin database [5]. We were able to replicate the results exactly, as shown in Figure 5.

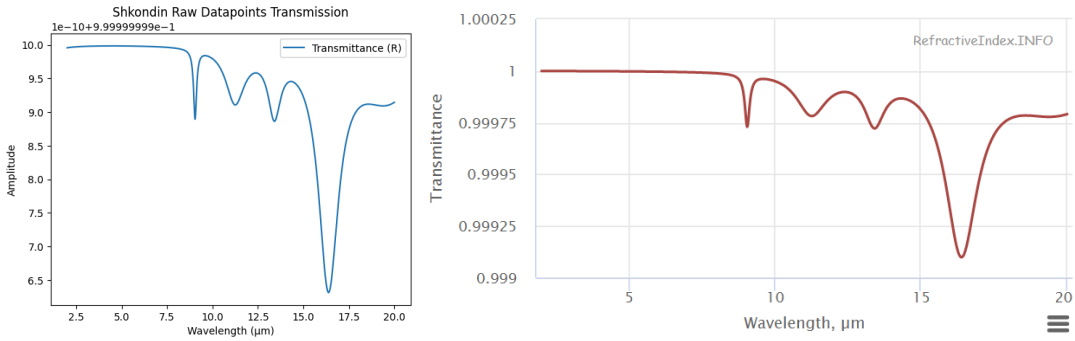

Figure 5: Two graphs of the refractive index of silicon using refractive indices from the Shkondin data set. The peaks are replicated exactly. One subtle thing to note is the distortion of the graph at the top left. When comparing the data points, however, we found the error to be 0 percent, so we chalked this up to be a visual distortion that is the result of the two graphs' different scales.
(Shkondin Dataset *https://refractiveindex.info*)

4. Conclusions

4.1 CEM in Python

The goal of the algorithm addressed in this paper is to fill a void in CEM resources in Python. While CEM has been traditionally dominated by proprietary software or tools implemented in other programming languages such as MATLAB or C++, Python's rise as a

powerful scientific computing platform offers a unique opportunity to democratize access to these advanced simulations. Python's ease of use, extensive libraries (e.g., NumPy, SciPy, and Matplotlib), and large community support make it an ideal choice for researchers, engineers, and students alike.

By providing a Python-based implementation, we aim to leverage its open-source ecosystem to make computational electromagnetics more accessible. The Python-based CEM tools can foster innovation by lowering the barrier to entry, allowing a wider audience to engage in the development and testing of new algorithms, and facilitating rapid prototyping. Furthermore, Python's integration with machine learning frameworks (such as TensorFlow and PyTorch) paves the way for future advancements, including data-driven CEM models and optimization techniques.

In the context of the TMM, this Python implementation can dramatically reduce the time and effort required for simulations of multi-layered materials, photonic devices, and other electromagnetic structures. The modularity and flexibility of Python also enable easier scalability for complex systems, which is crucial for applications in optics, telecommunications, and beyond. This approach brings electromagnetics into the modern Python-based scientific computing era, positioning it as a critical tool for both academia and industry.

4.2 Future Works

This paper lays the groundwork for more open-source CEM tools in Python, but it is only the tip of the iceberg. The algorithm still has many flaws, and although TMM is the bread and butter of CEM, there are more advanced algorithms such as rigorous coupled wave analysis, 3D finite difference time domain, and planar wave expansion method that need to be implemented. This aims to be the first of many implementations of various CEM algorithms.

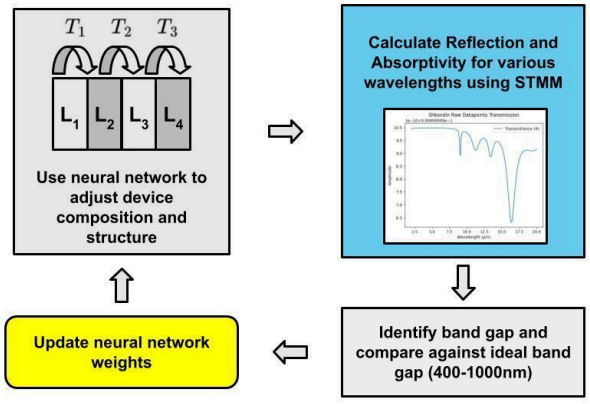

Figure 6: Potential reinforced learning loop to optimize solar panel coating

However, that is not all, as Python's existing machine learning libraries become trivial to integrate as a result of STMM's native Python implementation. Therefore, we can integrate this algorithm into more complex iterative reinforced learning loops as a part of the evaluation step. We propose that this algorithm can be integrated into a ML loop as shown in Figure 6 in order to optimize an antireflective coating for solar panels.

5. References

1. Tombe, Frederick. (2011). Maxwell's Original Equations.
2. Jarry, P., Beneat, J.N., Jarry, P. and Beneat, J.N. (2014). Electromagnetic in Linear, Homogeneous, Isotropic and Lossless Guides. In RF and Microwave Electromagnetism (eds P. Jarry and J.N. Beneat). https://doi.org/10.1002/9781118985090.ch4
3. Nizamudeen Cherupurakal, Mohammad Sayem Mozumder, Abdel-Hamid I. Mourad, Shubra Lalwani, Recent advances in superhydrophobic polymers for antireflective self-cleaning solar panels, Renewable and Sustainable Energy Reviews, Volume 151, 2021, 111538, ISSN 1364-0321, https://doi.org/10.1016/j.rser.2021.111538.
4. Rumpf, Raymond. “Computational Electromagnetics.” EMPossible, 14 Dec. 2023, empossible.net/academics/emp5337/.
5. E. Shkondin, O. Takayama, M. E. Aryaee Panah, P. Liu, P. V. Larsen, M. D. Mar, F. Jensen, and A. V. Lavrinenko. Large-scale high aspect ratio Al-doped ZnO nanopillars arrays as anisotropic metamaterials, Opt. Mater. Express 7, 1606-1627 (2017) (Numerical data kindly provided by Osamu Takayama)

6. Appendix 1: Simple TMM Code Unoptimized

In [ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:
```python
def transfer_matrix(k_0, n, d, theta):
    k_z = k_0 * n * np.cos(theta) # Calculate longitudanal K

    # Reduce redundant calculations in construction T_i
    q_1 = np.cos(k_z * d)
    q_2 = 1j * np.sin(k_z * d)
    n_cos_th = n * np.cos(theta)

    # Transfer matrix calculation derived fro Maxwell's equations
    return np.array([
        [q_1, q_2 / n_cos_th],
        [n_cos_th * q_2, q_1]
    ])
```

In [ ]:
```python
def solve_tmm(layers, wavelength, theta):
    # Scales wave propagation
    k_0 = (2 * np.pi) / wavelength

    # Global transfer matrix initialization
    M = np.eye(2)

    #
    n_0 = layers[0][0]
    n_l = layers[-1][0]

    for layer in layers[1:-1]:
        # Get layer parameters
        n = layer[0](wavelength) # Refractive index as a function
        d = layer[1]

        # Calculate transfer matrix
        theta_i = np.arcsin(n_0 * np.sin(theta) / n)
        T_i = transfer_matrix(k_0, n, d, theta_i)

        # Update global transfer matrix with dot product
        M = np.dot(M, T_i)

    # Calculate M_in and M_out
    q_1 = n_0 * np.cos(theta)
    theta_l = np.arcsin(n_0 * np.sin(theta) / n_l)
    q_2 = n_l * np.cos(theta_l)

    M_in = np.array([
            [1, 1],
            [q_1, -q_1],
        ])

    M_out = np.array([
            [1, 1],
            [q_2, -q_2],
        ])

    # Calculate M_global final
    M_total = np.dot(np.linalg.pinv(M_in), np.dot(M, M_out))

    # Extract reflection and transmission
    r = M_total[1, 0] / M_total[0, 0]
    t = 1 / M_total[0, 0]
    R = np.abs(r) ** 2
    T = np.abs(t) ** 2 * (n_l * np.cos(theta_l)) / (n_0 * np.cos(theta))

    return R, T
```

Statement on Outside Assistance

Researchers Name: Aniruth Ananthanarayanan

Title of Paper: Efficient Python-Based Transfer Matrix Method for Multilayer Optics: Analytical and Computational Optimizations

What steps led you to formulate your hypothesis? (Where did you get the idea for your research?) Please be specific.

I was led to formulate my hypothesis when trying to develop a foundational understanding of non-reciprocity. I realized that I wanted some kind of simulation to test out non-reciprocity. While researching libraries to do this, I found that there were not many resources, and any guides on how to develop these programs were focused on MATLAB implementations. I found various broken implementations, along with one that was very slow. This demonstrated the need for effective tools for computational electromagnetic tools in Python. Following along with some theoretical optimizations suggested by a YouTuber named EMPossible, I also discovered several potential computational optimizations that could be built into the program to increase performance and decrease memory usage.

Where did you conduct the major part of your work? (i.e. home, school, or other institutional setting, university lab, medical center, etc.)

I conducted most of my work in the university physics building, with some work being done in my dorm.

If you worked in an institutional setting, did you work on your project as part of a team or group? If so, how large was the team, and who was on the team (students, adult researchers, etc.)? Describe your role on the team.

I worked as a part of a group, with Dr. Khalsa guiding me. I would continuously iterate and develop my program, and Dr. Khalsa and I would meet weekly. In these meetings, Dr. Khalsa would help me identify gaps in my understanding so that I could have a fuller knowledge base to identify more potential optimizations. In summary, Dr. Khala guided me through the physics aspects while I did all of the programming.

Describe what parts of the research you did on your own and what parts you received help (i.e. literature search, hypothesis, experimental design, use of special equipment, gathering data, evaluation of data, statistical analysis, conclusions, and preparation of written report (abstract and/or paper).

I did a literature review, hypothesis formulation, and evaluation of data by myself. Dr. Khalsa guided me through the process of writing my abstract and reviewed my introduction. Dr. Khalsa also helped identify some potential libraries to benchmark against during the statistical analysis phase of my research.

Is your research current or a continuation of previous research? If a continuation, please describe the current work and advancement(s) of this research in comparison to the prior work and results.

My research is current research and was primarily conducted in the summer of 2024.