Hardware-level simulations of nanophotonic neural networks

Ben Bartlett
Department of Applied Physics
Stanford University
benbartlett@stanford.edu

Abstract

Artificial neural networks are biologically-inspired computational models which have found extensive use in machine learning applications. Modern computing hardware is inefficient at implementing neural networks, primarily because digital matrix multiplication is an $\Omega(N^2)$ operation. Here, we describe a fully-optical architecture consisting of meshes of self-configuring nanophotonic interferometers which is capable of performing $\mathcal{O}(1)$ matrix multiplication on an input vector of light intensities. Using detailed physical simulations of our interferometer design, we develop a theoretical control system for our architecture which generates the on-chip layout and applied voltages necessary to implement and train an arbitrarily specified feed-forward neural network.

1 Introduction

Neural networks have revolutionized machine learning over the last two decades, enabling computers to solve a huge variety of abstract problems. However, they are computationally expensive for modern digital computing hardware to implement, even once fully trained. Significant effort has been made towards developing specialized computing architectures to implement neural networks with greater speed and efficiency, but all of these designs are limited by the fact that digital $N \times N$ matrix multiplication is at least an $\mathcal{O}(N^2)$ operation (with the best known algorithm running in $\mathcal{O}(N^{2.37})$).

Photonic devices present particular promise for solving this problem. Using interferometric effects, it is possible to perform matrix multiplication in constant time on an input vector of coherent light intensities. [1] The use of photonic devices for implementing neuromorphic computation would allow for much greater data throughput, on the order of 100GHz [2], as optical devices do not have the rate-limiting heating effects that electric devices do and are limited only by laser modulation constraints. Additionally, a fully optical design would function as a passive device, so computations on the input signals would require no theoretical energy cost [3] beyond entropic losses imposed by the Landauer limit.

The general design for this computing scheme involves two parts: a layer of optical "neurons", which use optical nonlinearities such as saturable absorption to physically implement an activation function, and a photonic synaptic mesh to implement matrix transformation between layers in the network. The mesh would be composed of three layers [4], [5]: a block of modulated Mach-Zehnder interferometers (MZIs) would implement an arbitrary unitary transformation U; a layer of tunable optical attenuators would implement a diagonal matrix Σ, and another block of MZIs would apply another unitary transformation V^*, allowing for an arbitrary matrix M to be implemented with singular value decomposition as $M = U\Sigma V^*$. A possible design for the synaptic mesh is shown in Figure 1. In this paper, we focus primarily on the synaptic mesh, which is the dominant part of the computational complexity of implementing neural networks, since activation can be done digitally in $\mathcal{O}(N)$ and is highly parallelizable.

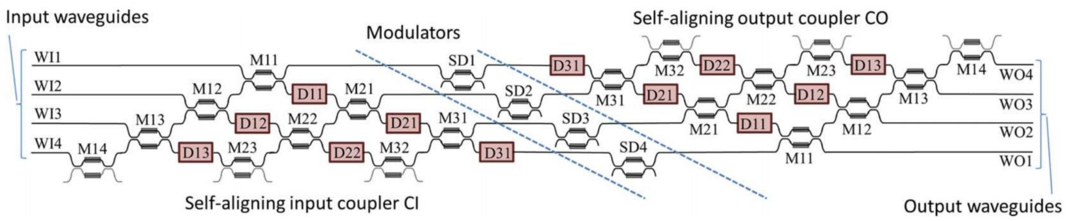

Figure 1: A possible design for the synaptic mesh which implements an arbitrary 4×4 matrix via singular value decomposition. Figure reproduced from (Miller 2013, "Self-configuring universal linear optical component" [1]).

2 Interferometer design

The synaptic layer consists of meshes of phase-shifted nanophotonic interferometers, each of which is capable of implementing an arbitrary $\mathrm{SU}(2)$ transformation (modulo some global phase). By choosing an appropriate arrangement of interferometers and phase shift values, any arbitrary linear transformation can be applied to an input vector of light intensities.

Our interferometer design, shown in Figure 2, consists of coupled waveguides and uses MEMS-based phase shifters, a departure from the better-explored use of thermal phase shifters. Two silicon nitride waveguides (refractive index 2.04) form a directional coupler with bends which approach to a 300nm separation. Two cantilevered silicon oxide blocks (refractive index 1.45) are suspended 250nm above the waveguides, tethered to a silicon nitride bridge. The bridge, pinned at both of the short edges, has an aluminum electrode on top of the nitride layer. By adjusting the potential difference between the electrode and the substrate electrode (not shown), the bridge can be made to flex downward due to capacitive forces, bringing the oxide block closer to the waveguide. The effective refractive index of the waveguide is changed depending on the proximity to the oxide block, inducing up to a $\frac{5}{2}\pi$ phase shift at the minimum gap distance.

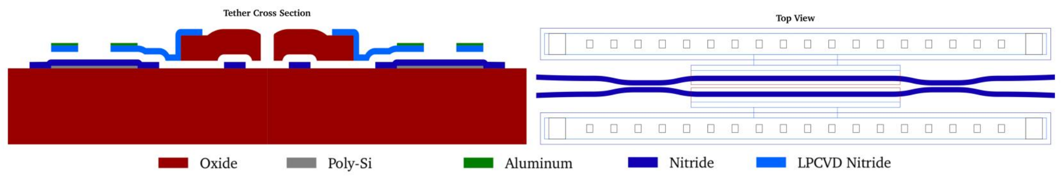

Figure 2: Our tunable MZI design uses silicon oxide blocks cantilevered from an actuated silicon nitride capacitive bridge to adjust the refractive index of the waveguide, inducing a tunable phase shift.

3 Interferometric matrix multiplication

To perform an arbitrary linear transformation on the input signals, we can apply some real diagonal transformation Σ on the inputs using optical attenuators in between two unitary transformations U and V^*, following singular value decomposition. The diagonal transformation is straightforward to implement, so the problem reduces to finding a method to perform an arbitrary transformation $U \in \mathrm{U}(N)$ on input signals using some arrangement of interferometers and single-mode phase shifters.

A single-mode phase shifter of the type presented in Section 2 can perform an arbitrary $\mathrm{U}(1)$ transformation $e^{i\phi}$ on its input. A single two-port interferometer can perform a transformation T on its inputs of the form:

$$T(\theta, \phi) = \begin{bmatrix} e^{i\phi}\cos\theta & -\sin\theta \\ e^{i\phi}\sin\theta & \cos\theta \end{bmatrix}. \tag{1}$$

When combined with a single-mode phase shifter, an arbitrary $\mathrm{SU}(2)$ transformation can be performed. If there are N input waveguides and the interferometer is connected to waveguides m and n, then a transformation $T_{m,n}$ (shown here for $m = n-1$) is performed:

$$T_{m,n}(\theta,\phi) = \begin{bmatrix} 1 & 0 & \cdots & & & & 0 \\ 0 & 1 & & & & & \vdots \\ \vdots & & \ddots & & & & \\ & & & e^{i\phi}\cos\theta & -\sin\theta & & \\ & & & e^{i\phi}\sin\theta & \cos\theta & & \\ & & & & & \ddots & \vdots \\ & & & & & 1 & 0 \\ 0 & \cdots & & & & 0 & 1 \end{bmatrix}. \tag{2}$$

3.1 Unitary matrix decomposition

Although most existing similar work uses triangular architectures based on Reck, *et al* [6], we follow a matrix decomposition approach based on the improved design of Clements, *et al* [7], the square architecture of which is more compact and exhibits greater fidelity with respect to component loss and network size. The decomposition algorithm relies on two facts: for any $U \in \mathrm{U}(N)$, there exist θ, ϕ to make any target element in row m or n of matrix $T_{m,n}(\theta,\phi)U$ equal to zero; similarly, $\exists \theta, \phi$ such that a target element in column m or n of $UT^{-1}_{m,n}(\theta,\phi)$ is zero. We refer to this as "nullifying" the target element of U. The decomposition algorithm we use is presented below.

Algorithm 1 $\mathrm{U}(N) \to \mathrm{U}(2)$ matrix decomposition

```
let $U \in \mathrm{U}(N)$ be an arbitrary unitary matrix and let $\hat{U} = U$
for $i = 1$ to $N-1$ do
    if $i$ is odd then
        for $j = 0$ to $i-1$ do
            find $\theta, \phi$ such that $T^{-1}_{i-j,i-j+1}(\theta,\phi)$ nullifies element $N-j, i-j$ of $\hat{U}$
            update $\hat{U} \Leftarrow \hat{U}T^{-1}_{i-j,i-j+1}(\theta,\phi)$
        end for
    else
        for $j = 1$ to $i$ do
            find $\theta, \phi$ such that $T_{N+j-i-1,N+j-i}(\theta,\phi)$ nullifies element $N+j-i, j$ of $\hat{U}$
            update $\hat{U} \Leftarrow T_{N+j-i-1,N+j-i}(\theta,\phi)\hat{U}$
        end for
    end if
end for
return $L = \{T^{-1}_{m,n}\}$, $\hat{U}$, $R = \{T_{m,n}\}$
```

At the end of the decomposition procedure, all off-diagonal elements of $\hat{U}$ have been nullified, so $\hat{U}$ is diagonal with elements $e^{i\phi_n}$ on the diagonal. Using the notation of Algorithm 1, if there are l elements in the ordered list of two-mode transformations $L = \{T^{-1}_{m,n}\}$ and r elements in $R = \{T_{m,n}\}$, then we have that $\hat{U} = T^{-1}_{m_1,n_1} \cdots T^{-1}_{m_l,n_l} U T_{m_1,n_1} \cdots T_{m_r,n_r}$. Let $T_R = \{(T^{-1}_{m_l,n_l})^{-1}, (T^{-1}_{m_{l-1},n_{l-1}})^{-1}, \cdots, (T^{-1}_{m_1,n_1})^{-1}\}$ be the inverse of each matrix in L, in reversed order, and construct T_L from R analogously as $T_L = \{(T_{m_r,n_r})^{-1}, \cdots, (T_{m_1,n_1})^{-1}\}$ Solving for the original operator U, we obtain the desired result – a representation of any $U \in \mathrm{U}(N)$ using a sequence of embedded $\mathrm{U}(1)$ and $\mathrm{SU}(2)$ transformations:

$$U = \left(\prod_{T^{-1}_{m,n} \in T_L} T^{-1}_{m,n} \right) \hat{U} \left(\prod_{T_{m,n} \in T_R} T_{m,n} \right). \tag{3}$$

Thus, since we have a way to implement any unitary matrix using only interferometers and phase shifters, if we include a layer of optical attenuators or gain elements, then we can implement any arbitrary matrix W as:

$$W = U\Sigma V^* = \left(\prod_{T_{U,L}} T_{m,n}^{-1} \cdot \hat{U} \cdot \prod_{T_{U,R}} T_{m,n} \right) \Sigma \left(\prod_{T_{V^*,L}} T_{m,n}^{-1} \cdot \hat{V}^* \cdot \prod_{T_{V^*,R}} T_{m,n} \right). \quad (4)$$

We note that the iterative nullification procedure that forms the basis of our decomposition algorithm suggests a fault-tolerant method of phase shifter configuration that would correct for imperfections in the fabrication process. Suppose we add a switch to each output waveguide at the end of the interferometer mesh which can redirect the signals to a row of on-chip photodetectors. Rather than choosing θ, ϕ values to nullify the target element and flashing the voltages V_θ, V_ϕ suggested by our theoretical model (see Appendix A) to the device, we can measure the intensity of the output waveguide corresponding to the target element and directly sweep the voltages V_θ, V_ϕ on the MZI in question to minimize the measured output intensity, compensating for abnormalities of any particular interferometer.

4 `neuroptical` – a photonic neural network simulator

Using the voltage to phase shift relation found in Appendix A and the decomposition routine described in Section 3.1, we developed a modular framework for simulating photonic neural networks. The framework, dubbed "neuroptical", provides a number of classes for constructing and training feed-forward neural networks and various levels of abstraction. We review the main classes below, in increasing order of abstraction.

4.1 `neuroptical.components.MZI`

The `MZI` class is the most fundamental class in the simulation, representing an individual interferometer in the network. It tracks the individual potentials (V_θ, V_ϕ) applied to each of its two simulated capacitive silicon nitride bridges, ranging from 0V-10V. From this, the induced (θ, ϕ) phase shifts are computed using the theoretical model developed in Appendix A, with a normally-distributed random noise term $\sigma_\theta = \sigma_\phi = 0.005$ added to simulate phase shifting imperfections (0.005 value taken from estimates in the supplementary materials of [4]). The `MZI.unitary()` method returns the effective $\mathrm{U}(2)$ transformation generated by the interferometer (generating new noise with each call), and `MZI.operator()` method returns the transformation embedded at input waveguides m, n in the N-dimensional Hilbert space of the network layer.

4.2 `neuroptical.components.MZIBlock`

The `MZIBlock` class represents a block of interferometers and one column of single-mode phase shifters which implements some $N \times N$ unitary matrix. The class is instantiated with a single argument (N) and generates an appropriate arrangement of interferometers using the modified Clements decomposition presented in Section 3.1, randomly initializing component voltages. The class includes an `MZIBlock.get_matrix()` method, which constructs the current matrix implemented by the system from the current voltages applied to each interferometer, accounting for noise, and an `MZIBlock.set_matrix(U)` method, which iteratively modifies each interferometer's parameters to implement the desired new matrix (which must have the same dimensions).

4.3 `neuroptical.components.OIU`

The `OIU` (Optical Interference Unit, in keeping with the terminology in [4]) class combines two `MZIBlocks` and a layer of gain components to implement an arbitrary matrix via singular value decomposition. The gain components are not simulated at a physical level, as the interferometric mesh is the primary focus of this study and since any of a variety of physical (or digital, in the case of a hybrid photonic-classical network) implementations could be used for this. The class includes methods for getting and setting the matrix which call their analogous `MZIBlock` methods.

4.4 `neuroptical.components.NetworkLayer`

The `NetworkLayer` class represents a single layer of a feed-forward neural network and is composed of an `OIU` to implement the weight matrix and an activation layer. It contains a variety of methods for implementing various activation functions, forward propagation, and backpropagation.

4.5 `neuroptical.components.PhotonicNeuralNetwork`

The `PhotonicNeuralNetwork` class has the highest level of abstraction and is a fairly standard implementation of a neural network. It is instantiated with a list of layer dimensions and activation functions, and includes methods for forward propagation, backpropagation, training, and classification. It also includes a `PhotonicNeuralNetwork.make_drawing()` method which generates a schematic representing the network architecture on chip and displays all parameters of the network, as shown in Figure 4.

5 Demonstration: planar data classification

To demonstrate the capabilities of the photonic network simulation framework described in this paper, it is informative to choose a sufficiently simple machine learning task such that the network required to implement it can be illustrated on this page. The planar data classification task from the beginning of CS230 is a good example. We used the same architecture as given in the assignment, with two inputs, a 5-dimensional hidden layer with a $\tanh$ activation function, and a sigmoid activation on the output layer. The cross-entropy loss during training and the post-training classification of the dataset are shown below in Figure 3. Finally, a diagram depicting the layout of this network on-chip and voltage and gain parameters necessary to implement the trained behavior is shown in Figure 4.

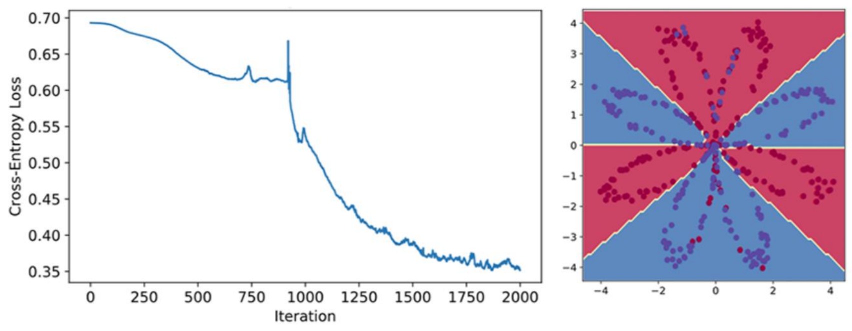

Figure 3: Cross-entropy loss during training and the post-training classification of the dataset. Note the stochastic nature of the loss during training; this is due to the phase-shifter noise simulated by the `MZI` class. After investigating this behavior with several other scenarios, we have found that sharp peaks in the loss, such as the one around iteration 900, tend to arise when attenuators are set to high gain values, increasing the effect of small number of interferometers on the overall network behavior.

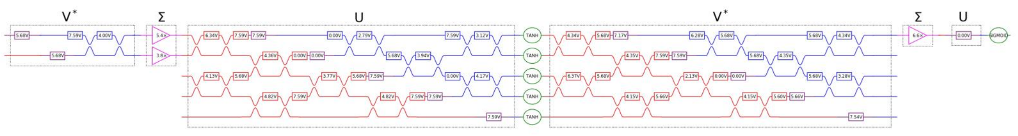

Figure 4: The on-chip layout of the trained photonic network which generates the classification shown in Figure 3. The voltages applied to each phase shifter are illustrated in the corresponding box, and the gain values are shown in the triangles. Although we do not currently have designs for the physical implementations of the gain elements (magenta) or activation elements (green), for scale, the dimensions of our current interferometer design (red or blue, depicted as two waveguides with two boxes) is approximately $100\mu\mathrm{m} \times 25\mu\mathrm{m}$.

References

[1] D. A. B. Miller, "Self-configuring universal linear optical component [Invited]," *Photonics Research*, vol. 1, no. 1, p. 1, 2013, ISSN: 2327-9125. DOI: `10.1364/PRJ.1.000001`. [Online]. Available: `https://www.osapublishing.org/prj/abstract.cfm?uri=prj-1-1-1`.

[2] L. Vivien, A. Polzer, D. Marris-Morini, J. Osmond, J. M. Hartmann, P. Crozat, E. Cassan, C. Kopp, H. Zimmermann, and J. M. Fédéli, "Zero-bias 40Gbit/s germanium waveguide photodetector on silicon," *Optics Express*, vol. 20, no. 2, p. 1096, 2012, ISSN: 1094-4087. DOI: `10.1364/OE.20.001096`. [Online]. Available: `https://www.osapublishing.org/oe/abstract.cfm?uri=oe-20-2-1096`.

[3] D. A. Miller, "Attojoule Optoelectronics for Low-Energy Information Processing and Communications," *Journal of Lightwave Technology*, vol. 35, no. 3, pp. 346–396, 2017, ISSN: 07338724. DOI: `10.1109/JLT.2017.2647779`.

[4] Y. Shen, N. C. Harris, S. Skirlo, D. Englund, and M. Soljačić, "Deep learning with coherent nanophotonic circuits," in *Summer Topicals Meeting Series, SUM 2017*, 2017, ISBN: 9781509065707. DOI: `10.1109/PHOSST.2017.8012714`.

[5] A. Annoni, E. Guglielmi, M. Carminati, G. Ferrari, M. Sampietro, D. A. Miller, A. Melloni, and F. Morichetti, "Unscrambling light - Automatically undoing strong mixing between modes," *Light: Science and Applications*, vol. 6, no. 12, 2017, ISSN: 20477538. DOI: `10.1038/lsa.2017.110`.

[6] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," *Physical Review Letters*, vol. 73, no. 1, pp. 58–61, 1994, ISSN: 00319007. DOI: `10.1103/PhysRevLett.73.58`.

[7] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, "An Optimal Design for Universal Multiport Interferometers," no. 2, pp. 1–8, 2016, ISSN: 2334-2536. DOI: `10.1364/OPTICA.3.001460`. [Online]. Available: `http://arxiv.org/abs/1603.08788`.

Appendix A Characterizing a voltage to phase shift relation

Of crucial importance to simulating our photonic neural network is characterizing how our phase shifter design responds to applied voltages. Since we do not yet have a fabricated sample to run empirical tests on, we instead theoretically characterize the way that potential difference across the bridge electrodes affects induced phase shift. We do this by breaking the problem into two parts: finding a relation between bridge flexure and phase shift, and finding a relation between potential difference across the electrodes and bridge flexure.

A.1 Bridge flexure to phase shift relation

A batch of 100 numerical finite finite-difference time-domain (FDTD) simulations (example shown below) were run in Matlab to determine the effective phase shift induced by the oxide block at various proximities to the waveguide, ranging from 0nm to 1000nm. An additional 100 simulations were performed in the range of 0nm to 100nm to achieve finer resolution, and an interpolating function was constructed in Mathematica from the simulation results. This was normalized by the dimensions of the oxide block to find the effective phase shift per unit length as a function of waveguide proximity, shown in Figure 5.

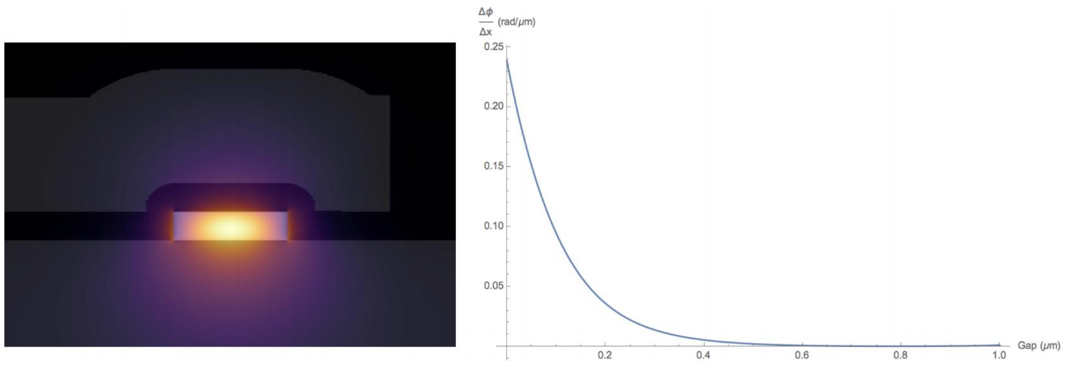

Figure 5: Left: an example finite-difference time-domain depicting the intensity of the electric field within the waveguide and oxide block. Right: the interpolated function describing the induced phase shift per unit length of oxide block as a function of gap distance between the block and waveguide.

A.2 Potential difference to bridge flexure relation

We modeled the bridge flexure as an Euler-Bernoulli beam, obeying the steady-state equation

$$\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 w(x)}{\partial x^2}\right) = q(x), \tag{5}$$

where x is the distance along the bridge axis (with $x = 0$ being the center of the bridge), E is the elastic modulus of the material (about 314GPa for $\mathrm{Si_3N_4}$), I is the second moment of area of the bridge, defined in Cartesian coordinates as $I = \iint_\Omega y^2 dx dy$ with Ω the parameterization of the shape, and $q(x)$ is the applied (capacitive) force per unit length.

Using bridge dimensions of $\delta x, \delta y, \delta z = 55, 1.1, 0.5\mu\mathrm{m}$, we account for self-capacitive effects of the bridge by setting

$$q(x) = \frac{\delta y \epsilon_0 \Delta V^2}{2\left(d + w(x)\right)}, \tag{6}$$

where ΔV is the applied potential difference between the bridge electrodes and d is the distance between the electrodes when no potential is applied and the bridge is at equilibrium. Imposing rigid boundary conditions of $w(-\frac{\delta x}{2}) = w(\frac{\delta x}{2}) = 0$ and $w'(-\frac{\delta x}{2}) = w'(\frac{\delta x}{2}) = 0$, we numerically solve

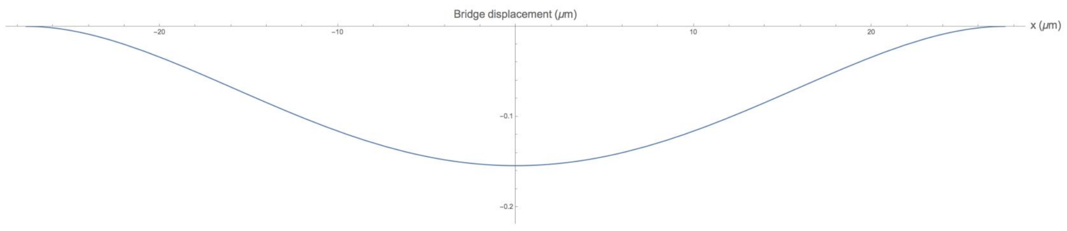

Figure 6: Bridge flexure at an applied potential difference of $\Delta V = 10\text{V}$. Note the differing scales on the axes.

the differential equation for a given voltage value to determine bridge flexure $w(x)$ at all points. An example bridge flexure for $\Delta V = 10\text{V}$ is shown in Figure 6.

A.3 Combining relations to obtain voltage to phase shift relation

Finally, we combine the two relations we have obtained to obtain a relationship between ΔV and $\Delta\phi$. Since the oxide block is cantilevered and attached only to the middle of the bridge, we approximate the displacement of the block by the displacement of the bridge at $x = 0$. This gives the desired relation, which is depicted in Figure 7.

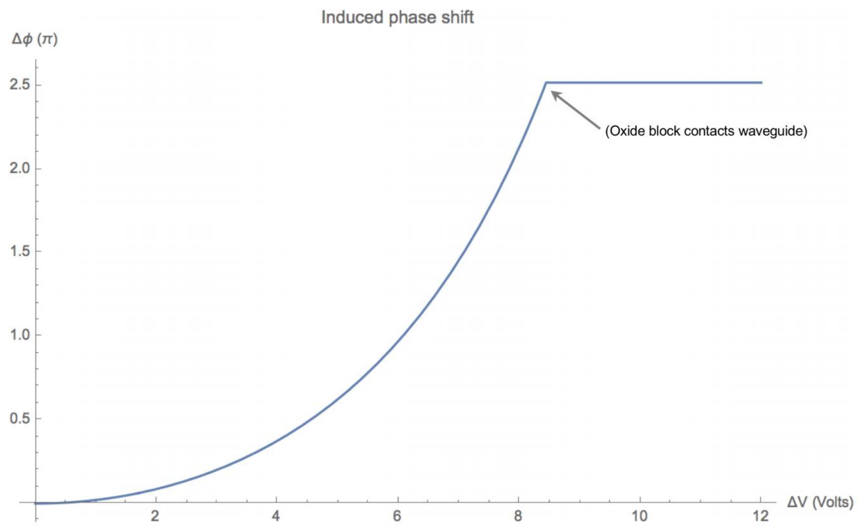

Figure 7: Our theoretical model for the relation between potential difference ΔV between the electrodes and the resulting phase shift $\Delta\phi$. The plateau at $\Delta V > 8\text{V}$ is due to the oxide block contacting the waveguide (hence being unable to get any closer).