A. NON-HOLOMORPHIC BACKPROPAGATION

In the previous derivation, we have assumed that the functions \( f_l(\cdot) \) are holomorphic. For each element of input \( Z_l \), labeled \( z \), this means that the derivative of \( f_l(z) \) with respect to its complex argument is well defined, or the derivative

\[
\frac{df_l}{dz} = \lim_{\Delta z \to 0} \frac{f_l(z + \Delta z) - f_l(z - \Delta z)}{2\Delta z}
\] (S1)

does not depend on the direction that \( \Delta z \) approaches 0 in the complex plane.

Here we show how to extend the backpropagation derivation to non-holomorphic activation functions. We first examine the starting point of the backpropagation algorithm, considering the change in the mean-squared loss function with respect to the permittivity of a phase shifter in the last layer OIU, as written in Eq. (7) of the main text as

\[
\frac{d\mathcal{L}}{d\epsilon_L} = \mathcal{R} \left\{ \Gamma_L^T \frac{dX_L}{d\epsilon_L} \right\}
\] (S2)

Where we had defined the error vector \( \Gamma_L \equiv (X_L - T)^* \) for simplicity and \( X_L = f_L(W_L X_{L-1}) \) is the output of the final layer.

To evaluate this expression for non-holomorphic activation functions, we split \( f_L(Z) \) and its argument into their real and imaginary parts

\[
f_L(Z) = u(\alpha, \beta) + iv(\alpha, \beta),
\] (S3)

where \( i \) is the imaginary unit and \( \alpha \) and \( \beta \) are the real and imaginary parts of \( Z_L \), respectively.

We now wish to evaluate \( \frac{dx_i}{d\epsilon_L} = \frac{dE_i(Z)}{d\epsilon_L} \), which gives the following via the chain rule

\[
\frac{df}{d\epsilon} = \frac{du}{d\epsilon} \odot \frac{d\alpha}{d\epsilon} + \frac{du}{d\beta} \odot \frac{d\beta}{d\epsilon} + i \frac{dv}{d\alpha} \odot \frac{d\alpha}{d\epsilon} + i \frac{dv}{d\beta} \odot \frac{d\beta}{d\epsilon},
\] (S4)

where we have dropped the layer index for simplicity. Here, terms of the form \( \frac{dx}{dy} \) correspond to element-wise differentiation of the vector \( x \) with respect to the vector \( y \). For example, the \( i \)-th element of the vector \( \frac{dx}{dy} \) is given by \( \frac{dx_i}{dy_i} \).

Now, inserting into Eq. (S2), we have

\[
\frac{d\mathcal{L}}{d\epsilon_L} = \mathcal{R} \left\{ \Gamma_L^T \odot \left( \frac{du}{d\alpha} + i \frac{dv}{d\alpha} \right)^T \frac{d\alpha}{d\epsilon_L} \right\} + \mathcal{R} \left\{ \Gamma_L^T \odot \left( \frac{du}{d\beta} + i \frac{dv}{d\beta} \right)^T \frac{d\beta}{d\epsilon_L} \right\}.
\] (S5)

We now define real and imaginary parts of \( \Gamma_L \) as \( \Gamma_R \) and \( \Gamma_I \), respectively. Inserting the definitions of \( \alpha \) and \( \beta \) in terms of \( W_L \)
and \( X_{L-1} \) and doing some algebra, we recover
\[
\frac{dL}{dL} = \mathcal{R} \left\{ \left( \Gamma_R \circ \frac{d}{\alpha} \right) \left( \Gamma_R \circ \frac{d}{\beta} \right) \frac{d}{\alpha} X_{L-1} \right\} (S7)
\]
\[
- \left( \Gamma_R \circ \frac{d}{\alpha} \right) \left( \Gamma_R \circ \frac{d}{\beta} \right) \frac{d}{\beta} X_{L-1} \right\} (S8)
\]
\[
-1 \left( \Gamma_R \circ \frac{d}{\alpha} \right) \left( \Gamma_R \circ \frac{d}{\beta} \right) \frac{d}{\beta} X_{L-1} \right\} (S9)
\]
\[
+1 \left( \Gamma_R \circ \frac{d}{\alpha} \right) \left( \Gamma_R \circ \frac{d}{\beta} \right) \frac{d}{\beta} X_{L-1} \right\} (S10)
\]
Finally, the expression simplifies to
\[
\frac{dL}{dL} = \mathcal{R} \left\{ \Gamma_R \circ \left( \frac{d}{\alpha} - i \frac{d}{\alpha} \right) \right\} (S11)
\]
\[
+ \Gamma_R \circ \left( -i \frac{d}{\alpha} + i \frac{d}{\beta} \right) \frac{d}{\beta} X_{L-1} \right\} (S12)
\]
As a check, if we insert the conditions for \( f_i(Z) \) to be holomorphic, namely
\[
d\alpha = \frac{dv}{d\beta} \quad \text{and} \quad \frac{d\alpha}{d\beta} = -\frac{dv}{d\alpha}
\]
Eq. (S10) simplifies to
\[
\frac{dL}{dL} = \mathcal{R} \left\{ \Gamma_R \circ \left( \frac{d}{\alpha} + \frac{dv}{d\alpha} \right) \right\}
\]
\[
\Gamma_R \circ \left( - \frac{dv}{d\alpha} + i \frac{d\alpha}{d\beta} \right) \frac{d}{\beta} X_{L-1} \right\}
\]
\[
\mathcal{R} \left\{ \left[ \Gamma_R \circ \frac{d}{\alpha} + \frac{dv}{d\alpha} \right] \frac{d}{\beta} X_{L-1} \right\}
\]
\[
= \mathcal{R} \left\{ \left[ \Gamma_R \circ f_i(Z) \right] \frac{d}{\beta} X_{L-1} \right\}
\]
\[
= \mathcal{R} \left\{ \delta_i \frac{d}{\beta} X_{L-1} \right\}
\]
as before.

This derivation may be similarly extended to any layer \( l \) in the network. For holomorphic activation functions, whereas we originally defined the \( \delta \) vectors as
\[
\delta_i = \Gamma_l \circ f_i(Z_l),
\]
for non-holomorphic activation functions, the respective definition is
\[
\delta_i = \Gamma_l \circ f_i(Z_l),
\]
where \( \Gamma_l \) and \( \Gamma_i \) are the respective real and imaginary parts of \( \Gamma_l \), and \( \alpha \) and \( \beta \) are the real and imaginary parts of \( Z_l \), respectively.

We can write this more simply as
\[
\delta_i = \mathcal{R} \left\{ \Gamma_l \circ \frac{df}{d\alpha} \right\} - i \mathcal{R} \left\{ \Gamma_l \circ \frac{df}{d\beta} \right\}
\]

In polar coordinates where \( Z = r \exp(i\phi) \) and \( f = f(r, \phi) \), this equation becomes
\[
\delta_i = \exp(-i\phi) \left\{ \mathcal{R} \left\{ \Gamma_l \circ \frac{df}{dr} \right\} - i \mathcal{R} \left\{ \Gamma_l \circ \frac{df}{d\phi} \right\} \right\}
\]
where all operations are element-wise.

### B. PHOTONIC NEURAL NETWORK SIMULATION

In Sections 4 and 5 of the main text, we have shown, starting from Maxwell’s equations, how the gradient information defined for an arbitrary problem can be obtained through electric field intensity measurements. However, since the full electromagnetic problem is too large to solve repeatedly, for the purposes of demonstration of a functioning neural network, in Section 6 we use the analytic, matrix representation of a mesh of MZIs as described in Ref. [1]. Namely, for an even \( N \), the matrix \( \mathcal{W} \) of the OIU is parametrized as the product of \( N + 1 \) unitary matrices:
\[
\mathcal{W} = \mathcal{R}_N \mathcal{R}_{N-1} \cdots \mathcal{R}_2 \mathcal{R}_1 \mathcal{D}
\]
where each \( \mathcal{R}_i \) implements a number of two-by-two unitary operations corresponding to a given MZI, and \( \mathcal{D} \) is a diagonal matrix corresponding to an arbitrary phase delay added to each port. This is schematically illustrated in Fig. S1(a) for \( N = 3 \). For the ANN training, we need to compute terms of the form
\[
\frac{dL}{d\phi} = \mathcal{R} \left\{ \gamma \frac{d}{d\phi} \mathcal{W} \right\}
\]
for an arbitrary phase \( \phi \) and vectors \( X \) and \( Y \) defined following the steps in the main text. Because of the feed-forward nature of the OIU-s, the matrix \( \mathcal{W} \) can also be split as
\[
\mathcal{W} = \mathcal{W}_2 \mathcal{R}_2 \mathcal{W}_1,
\]
where \( \mathcal{R}_\phi \) is a diagonal matrix which applies a phase shift \( e^{i\phi} \) in port \( i \) (the other elements are independent of \( \phi \)), while \( \mathcal{W}_1 \)
and \( \hat{W}_2 \) are the parts that precede and follow the phase shifter, respectively (Fig. S1(b)). Thus, Eq. (S24) becomes

\[
\mathcal{R}\left\{Y^T\frac{d\hat{W}_2}{d\phi} X\right\} = \mathcal{R}\left\{Y^T\frac{d\hat{W}_2}{d\phi} \hat{W}_1 X\right\} = -\mathcal{I}\left\{(W_2^T Y)ei\phi (\hat{W}_1 X)_i\right\},
\]

where \((V)_i\) is the \(i\)-th element of the vector \(V\), and \(\mathcal{I}\) denotes the imaginary part. This result can be written more intuitively in a notation similar to the main text. Namely, if \(A_{\phi}\) is the field amplitude generated by input \(X\) from the right, measured right after the phase shifter corresponding to \(\phi\), while \(A_{\phi}^{adi}\) is the field amplitude generated by input \(Y\) from the right, measured at the same point, then

\[
\frac{d\mathcal{L}}{d\phi} = -\mathcal{I}\left\{A_{\phi} A_{\phi}^{adi}\right\}
\]  

(S27)

By recording the amplitudes in all ports during the forward and the backward propagation, we can thus compute in parallel the gradient with respect to every phase shifter. Notice that, within this computational model, we do not need to go through the full procedure outlined in Section 4 of the main text. However, this procedure is crucial for the \textit{in situ} measurement of the gradients, and works even in cases which cannot be correctly captured by the simplified matrix model used here.

C. TRAINING DEMONSTRATION

In the main text we show how the \textit{in situ} backpropagation method may be used to train a simple XOR network. Here we demonstrate training on a more complex problem. Specifically, we generate a set of one thousand training examples represented by input and target \((X_0 \rightarrow T)\) pairs. Here, \(X_0 = [x_1, x_2, P, 0]^T\) where \(x_1\) and \(x_2\) are the independent inputs, which we constrain to be real for simplicity, and \(P(x_1, x_2) = \sqrt{P_0 - x_1^2 - x_2^2}\) represents a mode added to the third port to make the norm of \(X_0\) the same for each training example. In this case, we choose \(P_0 = 10\). Each training example has a corresponding label, \(y \in \{0, 1\}\) which is encoded in the desired output, \(T\), as \([1, 0, 0, 0]^T\) and \([0, 1, 0, 0]^T\) for \(y = 0\) and \(y = 1\) respectively.

For a given \(x_1\) and \(x_2\), we define \(r\) and \(\phi\) as the magnitude and phase of the vector \((x_1, x_2)\) in the 2D-plane, respectively. To generate the corresponding class label, we first generate a uniform random variable between 0 and 1, labeled \(U\), and then set \(y = 1\) if

\[
\exp\left(-\frac{(r - r_0 - \Delta \sin(2\phi))^2}{2\sigma^2}\right) + 0.1 U > 0.5.
\]  

(S28)

Otherwise, we set \(y = 0\). For the demonstration, \(r_0 = 0.6\), \(\Delta = 0.15\), and \(\sigma = 0.2\). The underlying distribution thus resembles an oblong ring centered around \(x_1 = x_2 = 0\), with added noise.

As diagrammed in Fig. S2(a), we use a network architecture consisting of \(4 \times 4\) layers of unitary OLUs, with an element-wise activation \(f(z) = |z|\) after each unitary transformation except for the last in the series, which has an activation of \(f(z) = |z|^2\). After the final activation, we apply an additional ‘softmax’ activation, which gives a normalized probability distribution corresponding to the predicted class of \(X_0\). Specifically, these are given by \(s(z_i) = \exp(z_i) / \left(\sum_j \exp(z_j)\right)\), where \(z_{i=1,2}\) is the first/second element of the output vector of the last activation (the other two elements are ignored). The ANN prediction for the input \(X_0\) is set as the larger one of these two outputs, while the total cost function is defined in the cross-entropy form

\[
\mathcal{L} = \frac{1}{M} \sum_{m=1}^{M} -\log(s(z_{m,l})),
\]  

(S29)

where \(\mathcal{L}^{(m)}\) is the cost function of the \(m\)-th example, the summation is over all training examples, and \(z_{m,l}\) is the output from the target port, \(l\), as defined by the target output \(T^{(m)}\) of the \(m\)-th example. We randomly split our generated examples into a training set containing 75% of the originally generated training examples, while the remaining 25% are used as a test set to evaluate the performance of our network on unseen examples.

As in the XOR demonstration, we utilized our matrix model of the system described in Section B. As in the main text, at each iteration of training we compute the gradient of the cost function with respect to the phases of each of the integrated phase shifters, and sum this over each of the training examples. For the backpropagation through the activation functions, since \(|z|\) and \(|z|^2\) are non-holomorphic, we use eq. S22 from Section A, to obtain

\[
\delta_L = 2Z^T_i \odot \mathcal{R}\{\Gamma_L\},
\]  

(S30)

\[
\delta_i = \exp(-i\phi_i) \odot \mathcal{R}\{\Gamma_i\},
\]  

(S31)

where \(\phi_i\) is a vector containing the phases of \(Z_i\) and \(\Gamma_i\) is given by the derivative of the cross-entropy loss function for a single training example

\[
\Gamma_L = \frac{\partial \mathcal{L}^{(m)}}{\partial z_{m,l}} = s(z_{m,l}) - \delta_{l,l},
\]  

(S32)

where \(\delta_{l,l}\) is the Kronecker delta.

With this, we can now compute the gradient of the loss function of eq. S29 with respect to all trainable parameters, and perform a parallel, steepest-descent update to the phase shifters,
in accordance with the gradient information. Our network successfully learned the task in around 4000 iterations. The results of the training are shown in Fig. S2(b). We achieved a training and test accuracy of 91% on both the training and test sets, indicating that the network was not overfitting to the dataset. This can also be confirmed visually from Fig. S2(c). The lack of perfect predictions is likely due to the inclusion of noise.

REFERENCES

1. W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walsmley, “Optimal design for universal multiport interferometers,” Optica 3, 1460 (2016).