Training deep neural networks for the inverse design of nanophotonic structures

Dianjing Liu, Yixuan Tan, Erfan Khoram, and Zongfu Yu*

Department of Electrical and Computer Engineering, University of Wisconsin, Madison, Wisconsin 53706, U.S.A

I. Data consistency in inverse design problems

The issue of data consistency in training data can be shown with the following example. Let X be an 8×1 real vector and Y be a 4×1 real vector (i.e., $X \in R^8$, $Y \in R^4$), while a nonlinear operator $\hat{O}$ defines a many-to-one mapping from the X space to the Y space:

$$Y = \hat{O}X \quad \text{(S1)}$$

The forward problem, i.e., calculating Y from X, is well-defined, and can be solved by training a forward neural network. However, when taking Y as the input and X as the output, the inverse network cannot be trained accurately. The following experiment shows that this is not only caused by non-unique instances in the training data, but also by inconsistency of the data set.

Let $\hat{O}_1^{-1}$ and $\hat{O}_2^{-1}$ be two different operators. For $\forall Y \in R^4$, the two operators satisfy

$$\hat{O}\left(\hat{O}_1^{-1}Y\right) = Y \quad \text{(S2)}$$

$$\hat{O}\left(\hat{O}_2^{-1}Y\right) = Y \quad \text{(S3)}$$

We generate data set D_1 from $\hat{O}_1^{-1}$ so that for each instance $< X_i, Y_i > \in D_1$, $X_i = \hat{O}_1^{-1}Y_i$. In this case, we say the data set D_1 is self-consistent, since instances in D_1 are sampled from the same mapping $\hat{O}_1^{-1}$. Another self-consistent data set D_2 is generated from $\hat{O}_2^{-1}$ in the same way. When D_1 and D_2 are put together to get a new data set $D_3 = D_1 \cup D_2$, the data set D_3 is not self-consistent.

The data set D_1, D_2, D_3 is used to train the inverse network, and the learning curves are shown in Fig. S1. The inverse networks are well trained by D_1 and D_2. However, the inconsistent data set D_3 cannot train an accurate neural network, even though instances are unique in D_3 (i.e., all instances have different Y values in D_3).

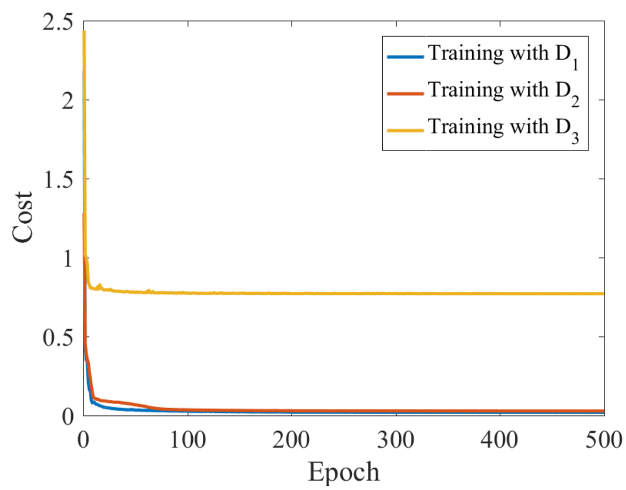

FIG. S1. Learning curve of an inverse network trained by data set D_1, D_2 and D_3. The sets D_1 and D_2 are self-consistent and can train accurate networks. The set D_3 fails to train an accurate network even though instances are unique within D_3.

II. Training forward neural network

In the following, we describe a specific implementation of the forward modeling network training process. To train the forward-modeling network for the multi-layer transmission problem, we experiment with networks having different sizes and depths. Fig. 6(a) compares the learning curves of the networks with different hidden layers. The architectures are as follows.

Architecture 1: $20 - 500 - 200$

Architecture 2: $20 - 500 - 200 - 200$

Architecture 3: $20 - 500 - 200 - 200 - 200$

Architecture 4: $20 - 500 - 200 - 200 - 200 - 200$

The 20 at the beginning and the 200 at the end are the numbers of input and output units, respectively. As the network becomes deeper, the error decreases, indicating more accurate predictions by the neural network. The network with four hidden layers (i.e., Architecture 4) has error ≈0.19 after 10,000 epochs of training.

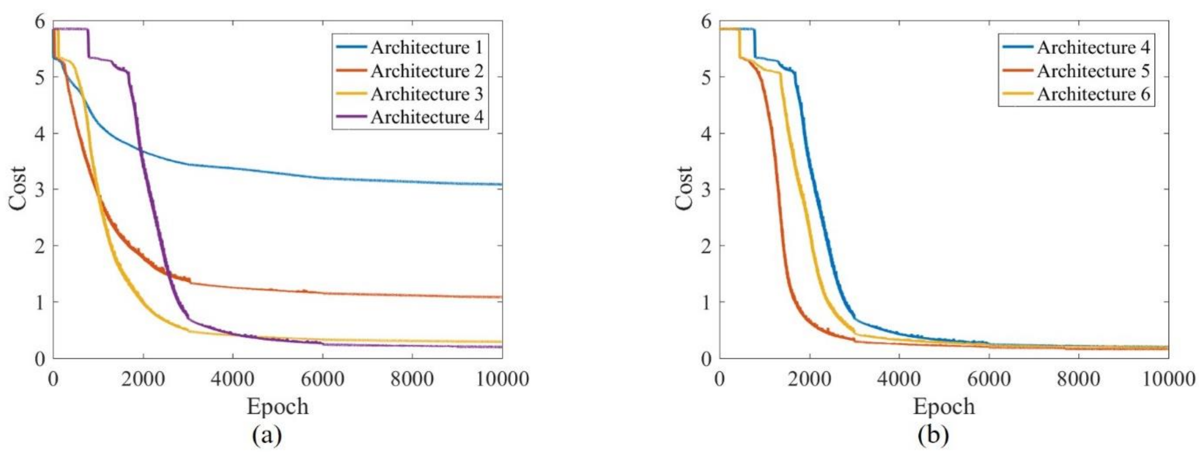

FIG. S2. (a) The learning curve for forward networks with different hidden layers. Architectures 1 to 4 have 1, 2, 3, and 4 hidden layers respectively. (b) The learning curve for forward networks with the same depth but different network sizes.

Fig. S2(b) compares networks with the same depth but different network sizes (number of hidden units in the hidden layers). The architectures are as follows.

Architecture 4: $20 - 500 - 200 - 200 - 200 - 200$,

Architecture 5: $20 - 500 - 500 - 200 - 200 - 200$,

Architecture 6: $20 - 500 - 500 - 500 - 200 - 200$.

The results indicate that larger networks could be trained faster, although as the training goes on, the performance difference becomes very little.

The network with Architecture 5 has an error ≈0.16 after 12,000 epochs of training. Fig. S3 shows its predictions on three instances randomly chosen from the test set. The ground truth (true transmission spectra) is shown in blue lines for comparison.

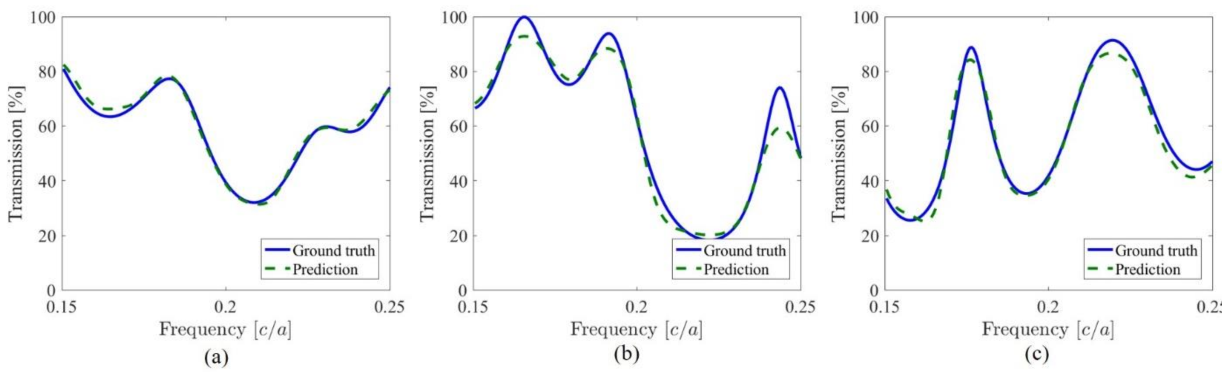

FIG. S3. Example test results of the forward network. The predictions by the network fit well with the ground truth.

III. Training neural network to design transmission phase delay of 2D structure

When designing 2D structures to modulate transmission phase delay, the forward modeling neural network has 6 hidden layers with each layer having 1024 – 512 – 512 – 256 – 256 – 128 hidden units. The inverse design network has 2 hidden layers with 512 and 256 hidden units. The learning rate is initially 0.0005 and exponentially decays to 10^{-6} at the end of the training. Learning curves of the forward modeling network and the tandem network are shown in Fig. S4.

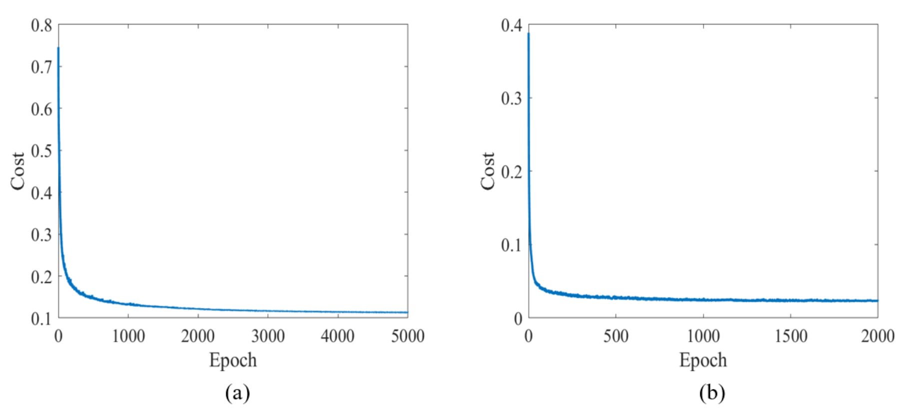

FIG. S4. Learning curve of (a) the forward modeling neural network and (b) the tandem network.