Supplemental Document

Optics Letters

Long short-term memory neural network for directly inverse design of nanofin metasurface: supplement

Wenqiang Deng,[1,2] Zhengji Xu,[1,2,*] Jinhao Wang,[1,2] and Jinwen Lv[1,2]

[1] *School of Microelectronics Science and Technology, Sun Yat-Sen University, China*
[2] *Guangdong Provincial Key Laboratory of Optoelectronic Information Processing Chips and Systems, Sun Yat-Sen University, Zhuhai 519082, China*
[*] *Corresponding author: xuzhj27@mail.sysu.edu.cn*

Supplement DOI: https://doi.org/10.6084/m9.figshare.20009003

Parent Article DOI: https://doi.org/10.1364/OL.458453

Long short-term memory neural network for directly inverse design of nanofin metasurface: supplement

WENQIANG DENG[1,2], ZHENGJI XU[1,2,*], JINHAO WANG[1,2], JINWEN LV[1,2]

[1]School of Microelectronics science and technology, Sun Yat-Sen University, China

[2] Guangdong Provincial Key Laboratory of Optoelectronic Information Processing Chips and Systems, Sun Yat-Sen University, Zhuhai 519082, China

**Corresponding author: xuzhj27@mail.sysu.edu.cn*

Long Short-term Memory Networks (LSTM):

Recurrent neural networks (RNNs) are typically used for speech recognition, voice recognition, time-series prediction, and natural language processing. This is useful for dealing with sequential data. Traditional neural networks assume that inputs and outputs are independent. Sometimes the data are interrelated; for example, a sentence contains many words, but we cannot know the meaning even if we know all the words in the sentence independently, we need to connect them to understand. The basic principle of an RNN is to save output and feedback to the input. Figure S1(a) shows the sample structure of an RNN. As shown in Figure S1(a), an RNN contains an input layer, output layer, hidden layer, and recurrent part. By unrolling this sample module, as shown in Figure S1(b), a more complex RNN is constructed that can handle more complicated sequential data. In figure S1(b), X(t-1), X(t), and X(t+1) represent the sequential data. H(t-1), H(t), and H(t+1) are the hidden layers in the sample structures t-1, t, and t +1, respectively. It is a chain network connecting the recurrent part labelled C.

When the sequential data increases, the RNN cannot connect with the data far from it. To overcome this issue, a new module referred to as long short-term memory networks (LSTM) was proposed. LSTM is a special type of RNN that is also a chain module capable of learning long-term sequential data. Figure S1(c) shows the chain structure of the LSTM. It contains many repeating modules. The repeating module of LSTM contains three processes, namely, the forget gate, input gate, and output gate, as shown in Figure S1(d). The basic formulas for these three processes are as follows.

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right), \text{(S1)}$$

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C_t}, \text{(S2)}$$

$$o_t = \delta\left(W_0 \times \left[h_{t-1}, x_t\right] + b_0\right), \text{(S3)}$$

$$h_t = o_t \times \tanh\left(C_t\right), \text{(S4)}$$

$$i_t = \sigma\left(W_i \times \left[h_{t-1}, x_t\right] + b_i\right), \text{(S5)}$$

$$\widetilde{C_t} = \tanh\left(W_c \times \left[h_{t-1}, x_t\right] + b_C\right), \text{(S6)}$$

where σ is the sigmoid function, W is the weight vector, C is the recurrent part, H is the hidden layer, b is a constant, t is a sequential number, and tanh is the activation function, $\tanh\left(x\right) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$. These six formulae provide a clear method of how LSTM works.

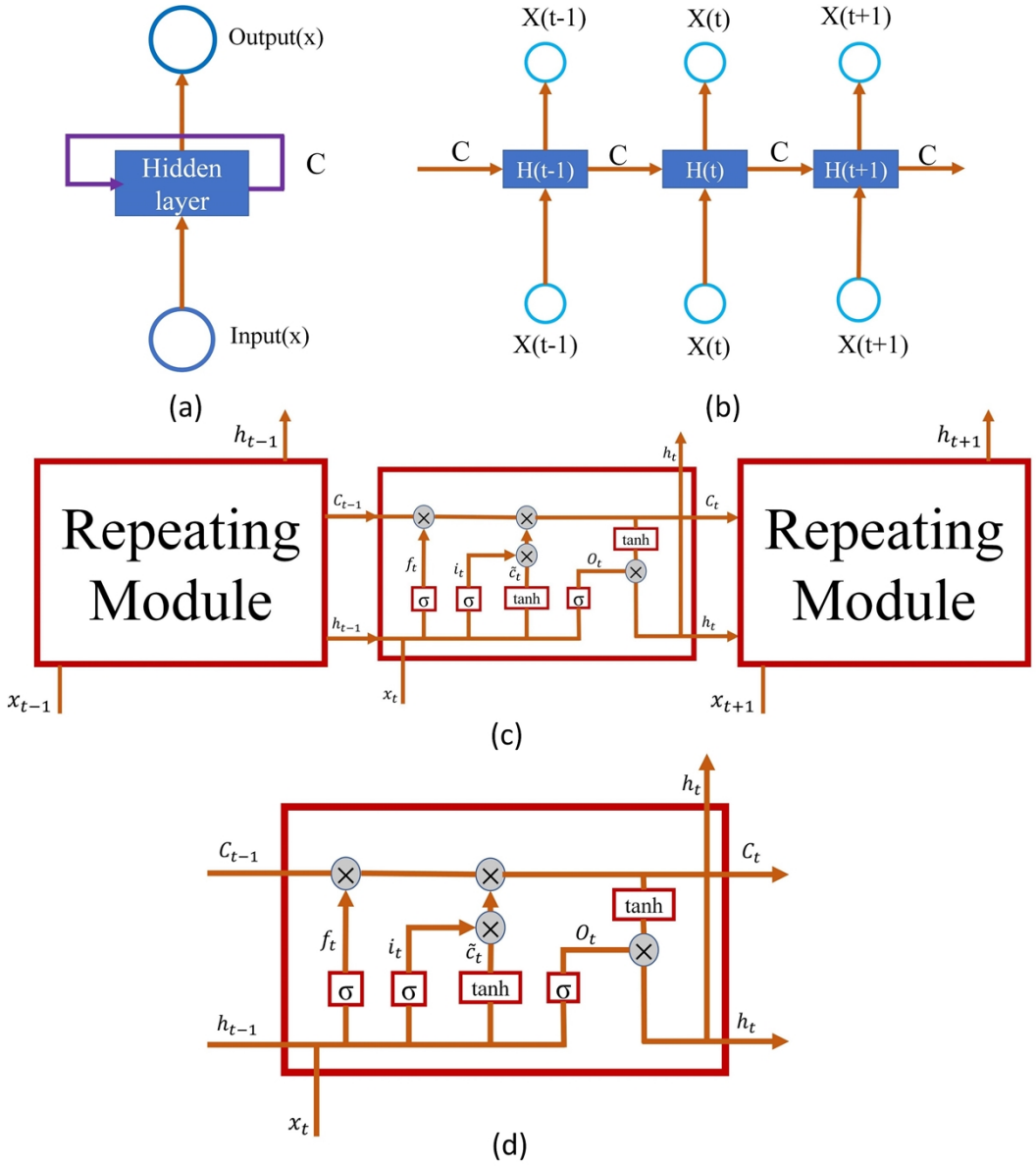

Figure1: (a) and (d) Basic unit of RNN and LSTM, (b) and (c) chain structure of RNN and LSTM.

The LSTM constructed in this study is shown in figure. S2(a). It contains six layers: one input layer, one output layer, and four hidden layers. The four hidden layers contain 10, 30, 50, and 80 cells, respectively. Figure S2(b) shows an inverse network. The inverse network is similar to the forward network. To ensure that the network has a rapid convergence speed, a linear connection was adopted in the last layer. It rotates 180° in contrast to the forward network. We generated 8000 groups of geometric parameters for an all-silicon unit cell and used the FDTD method to obtain the transmittances used for forward prediction and 30000 groups of data for inverse design. Four parameters are limited: P ∈ [6000 nm, 12000 nm], lx ∈ [800 nm, 2500 nm], ly ∈ [800 nm, 3000 nm], and h ∈ [6000 nm, 12000 nm].

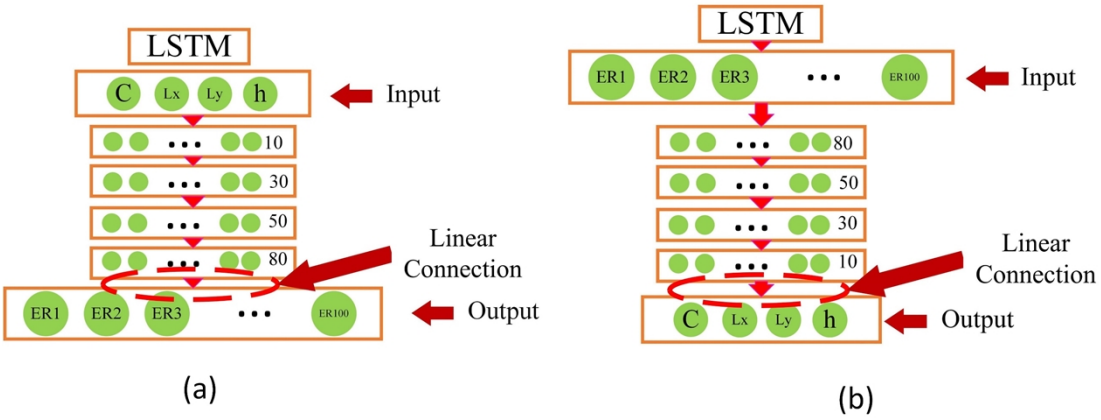

Figure 2S：Constructed LSTM: (a)Forward network; (b) Inverse network.

LSTM vs DNN:

As mentioned previously, LSTM is beneficial for dealing with sequential data. However, the input sequence can influence the output. We used the same dataset to train LSTM in forward training and used 200 groups of data that had no overlap of training data for testing to find the best input sequence. To quantify the results, we defined the mean value μ and standard deviation σ:

$$\mu = \frac{1}{200}\sum_{i=1}^{i=200}\{(\frac{1}{100}\sum_{j=1}^{j=100}\left|ER_{true}^{j} - ER_{pre}^{j}\right|\}), \text{(S7)}$$

$$\sigma = \sqrt{\left\{\frac{1}{200}\sum_{i=1}^{i=200}\left[\left(\frac{1}{100}\sum_{j=1}^{j=100}\left|ER_{true}^{j} - ER_{pre}^{j}\right|\right) - \mu\right]^{2}\right\}}, \text{(S8)}$$

where μ reflects the mean error for all 200 groups of the test data, and σ reflects robustness. The smaller the μ, the better the results, and the smaller the σ, the better the robustness of the predicted networks. Table S1 lists the detailed values for all the 24 input sequences. It can be observed that sequence 'h, lx, P, ly' has the smallest μ and σ. However, there was a slight difference in the input sequences. This indicates that the input sequence has little influence on our work.

Table S1: Different input sequences and DNN test results.

Sequence	**Mean (μ)**	**Standard Deviation (σ)**
h, ly, lx, P	0.1194137	0.188896
h, ly, P, lx	0.150545	0.246188
h, lx, ly, P	0.116764	0.179713
h, lx, P, ly	**0.104051**	**0.159589**
h, P, ly, lx	0.110479	0.210519
h, P, lx, ly	0.115903	0.191813
ly, h, lx, P	0.126897	0.206231
ly, h, P, lx	0.132201	0.246637
ly, lx, h, P	0.116858	0.221704
ly, lx, P, h	0.124892	0.23037
ly, P, h, lx	0.10783	0.16255
ly, P, lx, h	0.120799	0.213446
lx, h, ly, P	0.11137	0.165316
lx, h, P, ly	0.107926	0.181925
lx, ly, h, P	0.136157	0.232913
lx, ly, P, h	0.116059	0.193783
lx, P, h, ly	0.115937	0.192608
lx, P, ly, h	0.108862	0.176439
P, h, ly, lx	0.126021	0.200452
P, h, lx, ly	0.109699	0.181349
P, ly, h, lx	0.109689	0.172819
P, ly, lx, h	0.123038	0.256704
P, lx, h, ly	0.124796	0.221359
P, lx, ly, h	0.119976	0.205191
DNN	**0.22**	**0.32**

Because DNNs are most commonly used in deep learning for designing photonic structures, we constructed a six layers DNN for comparison with LSTM. The constructed DNN had the same units in each layer and number of layers, in contrast to the LSTM. We also used the 200 groups of test data mentioned above and calculated μ and σ. The training epochs is also the same with LSTM mentioned above, which is 30000. The test values for μ and σ are 0.22 and 0.32, respectively. These two values are much higher than those of LSTM. Table S2 presents the comparison of the DNN and LSTM.

Table S2: LSTM vs DNN in 200 groups of data

Models	**LSTM**	**DNN**
Mean Value (μ)	0.11	0.22
Stand Deviation (σ)	0.16	0.32
Accuracy	Better	-
Robustness	Better	-

Table S3: Well-trained networks: LSTM vs DNN

ID	**Number of hidden layers**	**Total nodes in all hidden layers**	**Training epochs (*10000)**	**Learn rate**	**μ (Mean error)**	**σ (standard deviation)**
LSTM	4	170	4	10^{-2}	0.025	0.018
DNN1	4	170	12	10^{-3}	0.059	0.048
DNN2	4	1000	8	10^{-3}	0.026	0.023
DNN3	6	1000	8	10^{-3}	0.024	0.022
DNN4	6	2000	8	10^{-3}	0.021	0.019

Since the LSTM and DNN are different architectures, we construct DNNs with different number of layers and nodes. Table S3 gives the results of well-trained LSTM and DNNs with same training dataset. The same 200 groups of data were used to testing. It shows that LSTM has higher accuracy and robustness than DNNs with same number of layers and nodes. Also, these DNNs need more training epochs to be well-optimized with same training dataset. Well-constructed DNNs can obtain the same accuracy and robustness. However, these DNNs have much greater number of layers and nodes than LSTM. It shows the virtue of LSTM to deal with extinction ration sequential data. Figure 3S shows the examples of predicting results for DNNs and LSTM.

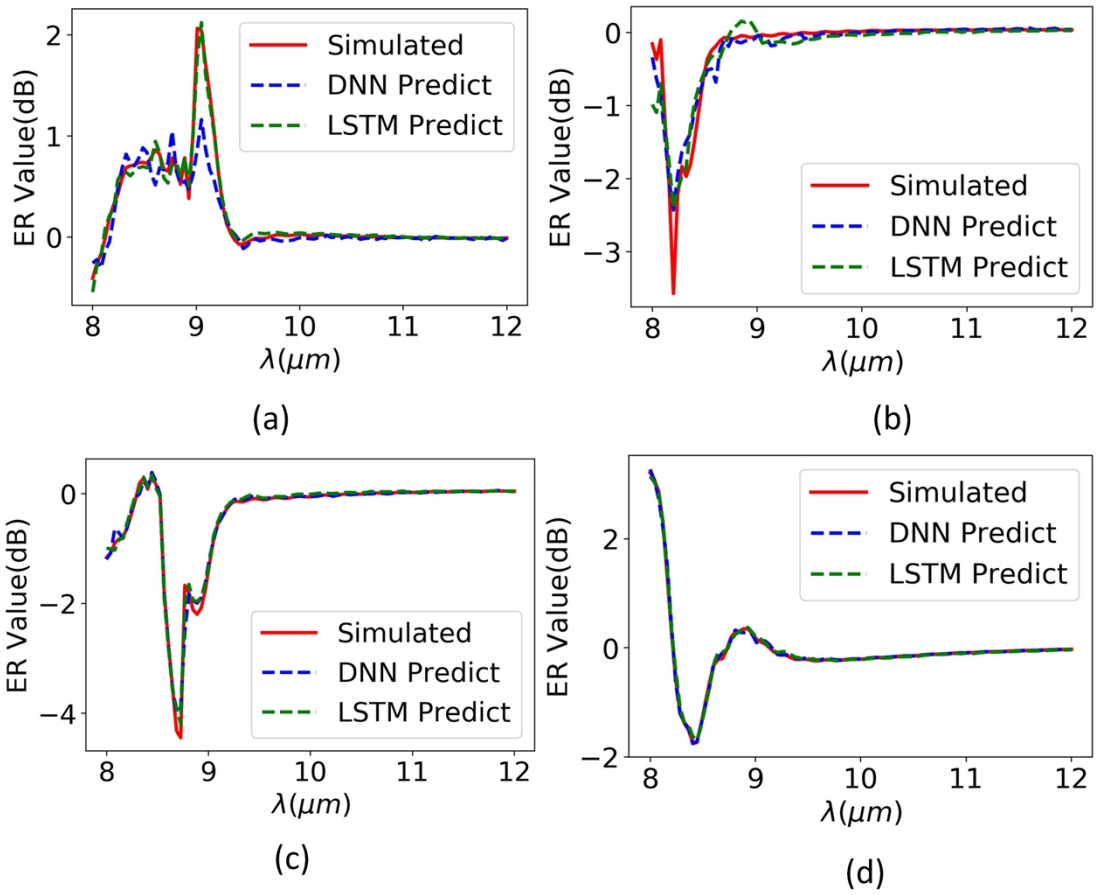

Figure 3S: The predicting examples for well-trained LSTM and DNNs. (a) LSTM vs DNN1; (b) LSTM vs DNN2;(c) LSTM vs DNN3;(d) LSTM vs DNN4