Forward Prediction and Inverse Design of Nanophotonic Devices Based on Capsule Network

Ruiyang Shi, *Graduate Student Member, IEEE*, Jie Huang, Shulun Li, Lingfeng Niu, and Junbo Yang

***Abstract*—Deep neural networks have been successfully applied to forward predicting optical response and inverse designing topological structure of nanophotonic devices. However, the existing deep learning based methods need sufficient simulated data to train the model effectively. For those devices with complex structures that containing many design variables, obtaining enough training data through numerical simulations will become extremely time-consuming. In order to reduce the requirement of large amounts of training data, we present a new deep learning approach based on the Capsule Network in this paper. By employing the proposed model, we have designed and verified a series of silicon-based wavelength demultiplexer with more than one thousand design variables. The numerical simulations validate that the trained model can both effectively predict the optical response with a fixed topological structure, and inverse design the approximate topological structure for a needed given optical response. Comparison with the classical convolutional neural networks show that our model can obtain nearly the same performance when using only 60% of the training data.**

***Index Terms*—Capsule network, forward prediction, inverse design, nanophotonic device.**

I. Introduction

NANOPHOTONICS is devoted to the study of interaction between light and matter at subwavelength scale. In the past few decades, nanophotonic devices have practically achieved a large scale of integration in many fields, including optical switches [1], [2], programmable nanophotonic processors [3], all-optical memories [4], [5], photonic neural network circuits [6], quantum photonic circuits [7] and other photonic integrated circuits [8]–[10]. Although the field of nanophotonics is developing rapidly, the design process is still challenging. An complete designing scheme for nanophotonic devices contains two major parts: forward prediction that outputs the optical response given the structure, and inverse design that outputs the structure from the required optical response. During the design process, solving Maxwell equations iteratively is required, which is time-consuming and leads to low efficiency [11]. The optimization-based methods [12], such as genetic algorithms [13], direct binary search algorithms [14]–[16], level set methods [17], adjoint methods [18], topology optimization methods [19] and objective-first algorithms [20]–[23], can automatically design the device according to the objective function and greatly improve the performance. However, they usually take massive simulations before finding a reasonable design. These methods become too slow to bear as the complexity of device increases.

In recent years, Deep Learning (DL) becomes more and more popular [24]. It learns the internal patterns and mapping relationships from the large amounts of training data, and can get good generalization performance on the unseen test dataset. As a one-time cost process, once the model is trained, the results can be obtained quickly. As an alternative, DL based methods are utilized to solve the forward prediction and inverse design problem of nanophotonic devices [11], [25]–[28] nowadays. For example, D. Liu *et al.* [26] used a Fully Connected Deep Neural Network (FCDNN) architecture to design the thickness parameter of a thin film consisting of alternating layers of SiO_2 and Si_3N_4. I. Malkiel *et al.* [27] constructed a bidirectional FCDNN to design and characterize plasmonic nanostructure with eight variables. J.Peurifoy *et al.* [28] exploited FCDNNs to design the eight-shell nanoparticle made of alternating shells of TiO_2 and SiO_2 with tens of dielectric shells. To deal with complex structure, W. Ma *et al.* [11] designed chiral metamaterial with five parameters using a two bidirectional neural networks combined with FCDNNs and Convolutional Neural Networks (CNNs). Noticing that DL has achieved success in the device design with several to dozens of variables, in this paper, we would exploit it to design the nanophotonic device with more than a thousand variables. Specifically, we construct a DL model to design the silicon-based wavelength demultiplexer with 1,352 variables, which is based on digital metamaterials [12] and is one of the key components of photonic integrated circuits.

Manuscript received May 5, 2022; revised June 5, 2022; accepted June 7, 2022. Date of publication June 13, 2022; date of current version June 21, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 11991021, and in part by the Fundamental Research Funds for the Central Universities under Grant E1E41502. *(Corresponding authors: Lingfeng Niu; Junbo Yang.)*

Ruiyang Shi is with the School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the CAS Research Center on Fictitious Economy and Data Science, University of Chinese Academy of Sciences, Beijing 100190, China (e-mail: shiruiyang18@mails.ucas.ac.cn).

Jie Huang is with the Center of Material Science, National University of Defense Technology, Changsha 410073, China (e-mail: huangjie10@nudt.edu.cn).

Shulun Li is with the State Key Laboratory for Superlattice and Microstructures, Institute of Semiconductors, Chinese Academy of Sciences, Beijing 100083, China, and also with the Center of Materials Science and Optoelectronics Engineering, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: lishulun@semi.ac.cn).

Lingfeng Niu is with the CAS Research Center on Fictitious Economy and Data Science, University of Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Economics and Management, University of Chinese Academy of Sciences, Beijing 100190, China (e-mail: niulf@ucas.ac.cn).

Junbo Yang is with the Center of Material Science, National University of Defense Technology, Changsha 410073, China (e-mail: yangjunbo@nudt.edu.cn).

Digital Object Identifier 10.1109/JPHOT.2022.3182050

As the data-driven approach, the existing DL based methods need a large number of data to train the model effectively. When training data is insufficient, the performance of the DL model would decline. However, generating substantial data often involves high-cost simulations and requires significant amounts of computational resources [26]. Taking the wavelength demultiplexer we designed in this work as an example, producing one single data needs to run for about 3 minutes on a server with 20 cores, 40 threads and 64 G memory, which is very expensive. Therefore, how to tackle this problem is worth studying and is important for designing complex nanophotonic devices. Recently, the idea of generating training data iteratively during the training process has emerged [29]–[31]. These models draw lessons from generative adversarial network [32]. Specifically, the device topology based on realistic design targets is constructed subsequently, and the true physical response of these devices is calculated in another iteration of simulations. These generated data are appended to the training data. The generator is then trained again on the fresh training data and this process is repeated. However, since the network training process is expensive which needs to be repeated several iterations on generating data [33], this kind of approach has the disadvantage of high computational cost. For this purpose, we concentrate on designing the DL model which needs relatively less data in this paper.

In deep neural networks, pooling, which originates from the visual mechanism, is a basic operation to abstract information. Although it brings many benefits, such as reducing information redundancy, improving the scale and rotation invariance, and preventing overfitting, it throws useful spatial information away at the same time [34], [35]. Thus, the resulting networks need more data to learn the intrinsic relationship between topology and spectral response [36] for compensating the loss of spatial information. To deal with this disadvantage, Hinton *et al.* proposed the concept of *capsule* [34] in 2011 and Sabour *et al.* first implemented the CapsNet [35] in 2017. Compared with a traditional neuron, which uses a single scalar to summarize the activities of feature detectors, a capsule is composed of a group of neurons, and is encapsulated into a vector of highly informative outputs [34]. For preserving the diversity of features, the CapsNet abandons the pooling operation and exploits a powerful dynamic routing mechanism instead to ensure that the output of the capsule can be sent to an appropriate parent in the layer above. These setups not only make the network require less data for training, but also enhance the interpretability of the model [35], [37]. Benefit from these superiorities, CapsNet has received lots of attention [38]–[41], and has been widely applied in many fields, including action detection [42], health care [43], [44], sentiment analysis [45], text classification [46], etc. In this work, we would apply CapsNet based DL model to forward predict optical response and inverse design nanophotonic device with thousands level design variables.

II. CapsNet for the Design Process of Nanophotonic Device

A. Silicon Based Wavelength Demultiplexer

The nanophotonic device we investigate in this paper is the wavelength demultiplexer. It is designed on a silicon-on-insulator (SOI) wafer. The schematic of the device is shown in Fig. 1. The SOI wafer has air cladding, 220 nm-thick Si layer and 2 μm-thick buried oxide layer. The broadband input light (fundamental TE mode) is launched from Port 1. After passing through the design region, the output light will be coupled out from Port 2 and Port 3. To ensure the fabricability of the designed demultiplexer, the design region is divided into $26 \times 52 = 1,352$ square pixels with the sizes of 120 nm $\times$ 120 nm. The state of each pixel is binary, namely 0 or 1. When the state of the pixel is 0, there will be a fully etched circle with the radius of 45 nm in the middle of the pixel. Otherwise there will be no etching. The simulated dataset is generated by the direct-binary-search (DBS) optimization algorithm [14]. Each data point in the dataset consists of the pixel matrix $\mathbf{x}$ and the corresponding spectral response T_2 of Port 2 and T_3 of Port 3. The spectral curve of each port is obtained by sampling 3000 points (from 1400 nm to 1700 nm). The states of 1,352 pixels are determined by figure-of-merit (FOM), which is defined as:

$$\mathrm{FOM} = 1 - 0.5\left(|1 - T_{2,\lambda_1}| + |1 - T_{3,\lambda_2}|\right). \tag{1}$$

Here T_{2,λ_1} and T_{3,λ_2} stand for the transmissions of Port 2 and Port 3 at the target wavelengths of λ_1 and λ_2, respectively.

B. CapsNet Modeling Process

In this paper, we exploit CapsNet to assist forward prediction and inverse design of wavelength demultiplexer. Specifically, the CapsNet we used can be divided into a Rectified Linear Unit (ReLU) convolutional layer, a Primary Capsule (PrimaryCaps) layer and a Digital Capsule (DigitCaps) layer. We denote the weight parameters of the first layer as $\mathbf{W}^{(1)} \in \mathbb{R}^{C_1 \times C_0 \times K_1 \times K_1}$, the second layer as $\mathbf{W}^{(2)} \in \mathbb{R}^{d_2 \times C_2 \times C_1 \times K_2 \times K_2}$, and the third layer as $\mathbf{W}^{(3)} \in \mathbb{R}^{M_2 \times M_3 \times d_3 \times d_2}$, where C_ℓ, K_ℓ are the dimensions of the ℓ-th weight tensor along the filter and spatial shape axes, respectively. d_ℓ is the dimension of each capsule in the ℓ-th layer. M_ℓ is the number of capsules in the ℓ-th layer. Here $M_2 = C_2(\lfloor(26 - K_1 - K_2 + 1)/\eta\rfloor + 1)(\lfloor(52 - K_1 - K_2 + 1)/\eta\rfloor + 1)$. M_3 is equal to the number of categories in the specific task. Besides, the bias parameters are denoted as $\mathbf{b}^{(1)} \in \mathbb{R}^{C_1}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^{C_2 d_2}$.

In the forward modeling process, the spectral response needs to be predicted according to the topology of the device. We train the CapsNet with mean square loss to predict the spectral transmission response vector, which is a two-dimensional vector with 3000 components in each of two ports. The ReLU convolutional layer receives $\mathbf{x}$ as input to extract low-level features. The obtained feature maps are sent to the next layer. A feature map $\mathbf{z}^{(1)} \in \mathbb{R}^{C_1 \times (26-K_1+1) \times (52-K_1+1)}$ is produced by:

$$\begin{aligned} z^{(1)}_{c_1,i,j} &= f\left(\left[\mathbf{x} \otimes \mathbf{W}^{(1)}_{c_1,:,:,:}\right]_{i,j} + b^{(1)}_{c_1}\right) \\ &= f\left(\sum_{c_0=1}^{C_0}\sum_{p_1=1}^{K_1}\sum_{p_2=1}^{K_1}\left[x_{c_0,p_1+i,p_2+j} W^{(1)}_{c_1,c_0,p_1,p_2}\right] + b^{(1)}_{c_1}\right), \end{aligned} \tag{2}$$

where f is the nonlinear activation, $\otimes$ is the convolutional operation, $c_0 \in \{1, \ldots, C_0\}$, $c_1 \in \{1, \ldots, C_1\}$, and ":" represents all

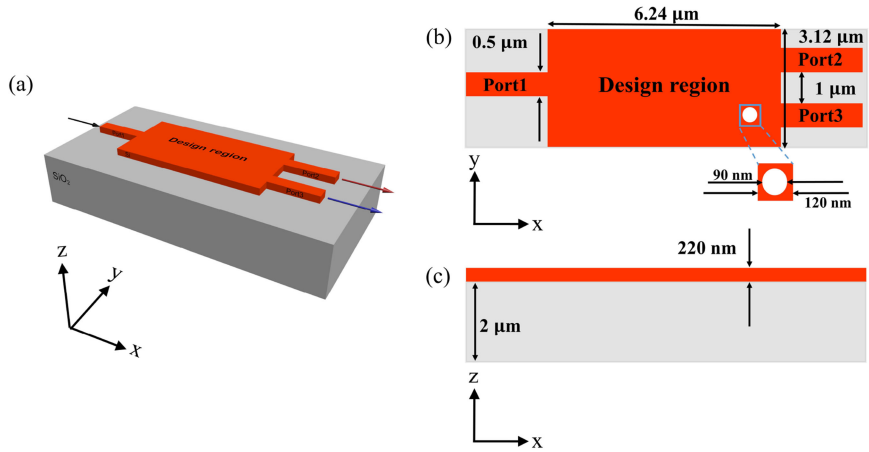

Fig. 1. Schematic of the device. (a) Three-dimensional diagram. (b) Top view. (c) Front view. Fundamental TE mode is launched from Port 1, after passing through the design region, the output light will be coupled out from Port 2 and Port 3.

subscripts for the specific dimension when it is used as subscripts in index expressions alone.

Then, the PrimaryCaps layer takes $\mathbf{z}^{(1)}$ as input, and stacks neurons together to form primary capsules. Compared with a traditional neuron, a capsule consists of a group of neurons. Instead of neurons that use a single scalar to summarize the activities of feature detectors, the capsule is proposed to operate some internal computations on their inputs and then encapsulates the results into a vector of highly informative outputs. The activities of the neurons within an active capsule represent the various properties of a specific entity appeared in the input image, including different types of instantiation parameters, such as position, size, orientation, etc. Among them, the position information of etching holes is the key for the task. The fact that capsules can store position information is the main reason we think that CapsNet is more suitable for nanophotonic device design than traditional neural networks.

Since convolutional operation has good advantages in image interpretation, and we would like to replicate learned knowledge across space, the primary capsules are still convolutional. Specifically, the PrimaryCaps layer is a convolutional capsule layer with C_2 channels of d_2 dimensional capsules (i.e. each primary capsule contains d_2 convolutional units with a $K_2 \times K_2$ kernel and a stride of η). Each primary capsule captures the outputs of all $C_1 \times K_1 \times K_1$ units in ReLU convolutional layer whose receptive fields overlap with the location of the center of the capsule. For each convolutional unit, the feature map $\mathbf{z}^{(2)}$ can be formulated as:

$$z^{(2)}_{d,c_2,i,j} = f\left(\left[\mathbf{z}^{(1)} \otimes \mathbf{W}^{(2)}_{d,c_2,:,:,:}\right]_{i,j} + b^{(2)}_{c_2}\right)$$
$$= f\left(\sum_{c_1=1}^{C_1}\sum_{p_1=1}^{K_2}\sum_{p_2=1}^{K_2}\left[x_{c_1,p_1+\eta i,p_2+\eta j} W^{(2)}_{d,c_2,c_1,p_1,p_2}\right] + b^{(2)}_{c_2}\right), \tag{3}$$

where η is the stride size, $c_2 \in \{1,\ldots,C_2\}$, and $d \in \{1,\ldots,d_2\}$. Compared with the former layer that generating a neuron, the PrimaryCaps layer needs d_2 convolution units to obtain capsules for each channel.

Algorithm 1: Dynamic Routing Algorithm.

procedure Routing$\hat{\mathbf{u}}^{(3)}_{j|i}, r$
  for all primary capsule i and digital capsule j: $a_{i,j} \leftarrow 0$
  for r iterations **do**
    for all primary capsule i: $c_{i,j} = \frac{\exp(a_{i,j})}{\sum_k \exp(a_{i,k})}$
    for all digital capsule j: $\mathbf{s}^{(3)}_j \leftarrow \sum_i c_{i,j}\hat{\mathbf{u}}^{(3)}_{j|i}$
    for all digital capsule j: $\mathbf{u}^{(3)}_j = \frac{\|\mathbf{s}^{(3)}_j\|^2}{1+\|\mathbf{s}^{(3)}_j\|^2}\frac{\mathbf{s}^{(3)}_j}{\|\mathbf{s}^{(3)}_j\|}$
    for all primary capsule i and digital capsule j: $a_{i,j} \leftarrow a_{i,j} + \mathbf{u}^{(3)}_j \cdot \hat{\mathbf{u}}^{(3)}_{j|i}$
  end for
  return $\mathbf{u}^{(3)}_j$
end procedure

The last layer is the DigitCaps layer with M_3 class capsules. Initially, each capsule is routed to all possible parents, and it is scaled down by coupling coefficients that sum to 1. For each possible parent, the routed capsule needs to compute a "prediction vector" by multiplying its own output with the corresponding transformation matrix. If the inner product of the prediction vector and the output of a possible parent is large, the dynamic routing mechanism will feed back this information to the coupling coefficient from top to bottom, which increases the coupling coefficient for this parent and decreases the coupling coefficients for other parents. The dynamic routing mechanism increases the contribution that the capsule makes to the parent, and enables the capsules representing different attributes to be clustered better. The dynamic routing process is given below.

In detail, we denote the coupling coefficients as $\mathbf{c} \in \mathbb{R}^{M_2 \times M_3}$ and the initial logits as $\mathbf{a} \in \mathbb{R}^{M_2 \times M_3}$. The capsule feature map $\mathbf{z}^{(2)}$ in the PrimaryCaps layer is first flattened to $\mathbf{u}^{(2)} = [\mathbf{u}^{(2)}_1, \ldots, \mathbf{u}^{(2)}_{M_2}] \in \mathbb{R}^{d_2 \times M_2}$ and capsules are represented as pose vectors. Then, each pose vector of capsule i in $\mathbf{u}^{(2)}$ needs to multiply a transformation matrix $\mathbf{W}^{(3)}_{i,j,:,:} \in \mathbb{R}^{d_3 \times d_2}$ to obtain the "prediction" vector $\hat{\mathbf{u}}^{(3)}_{j|i}$ corresponding to capsule j in the

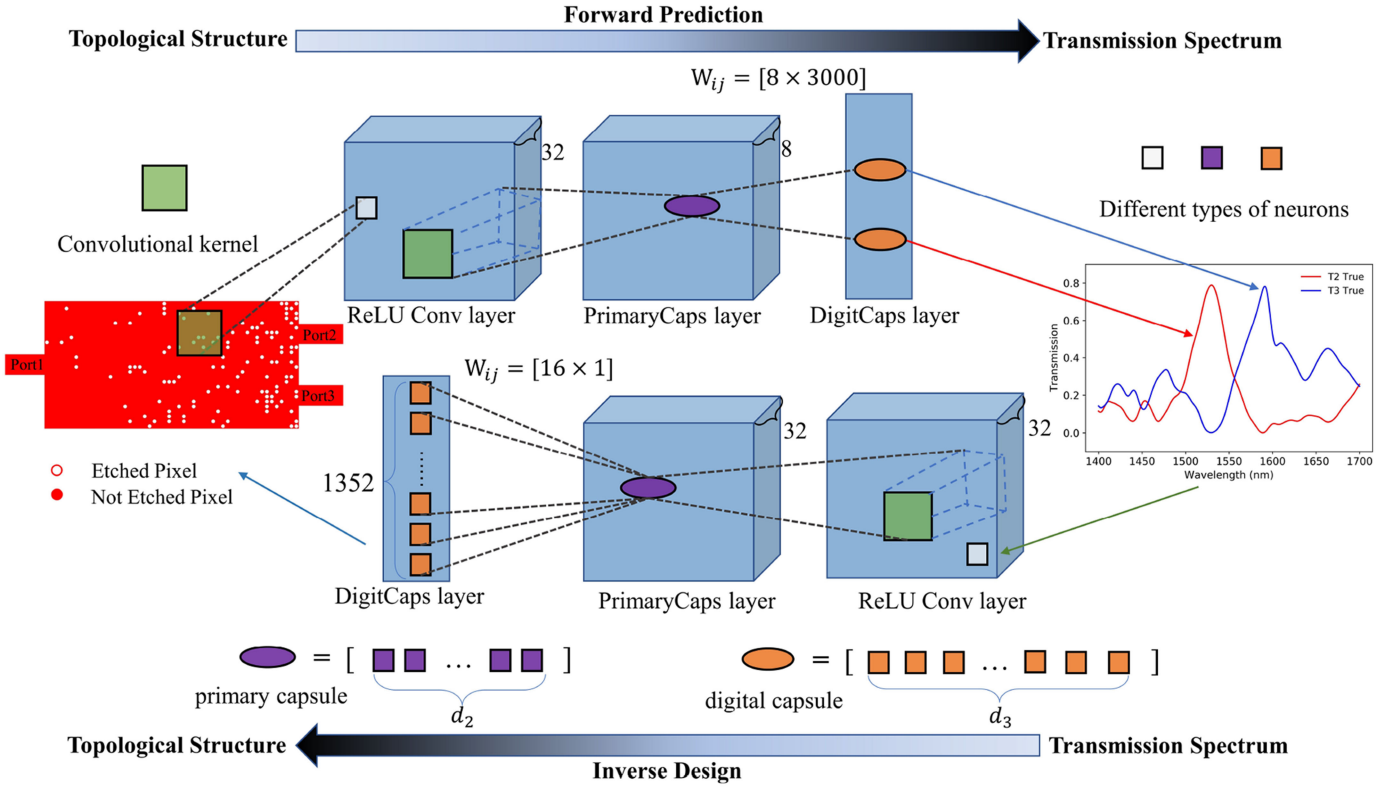

Fig. 2. Forward and inverse modeling process of CapsNet for wavelength demultiplexer. For the forward modeling process, the device topology binary image is the CapsNet model's input and the spectral response of the device is the output. For the inverse modeling process, the spectral response is first reshaped. Then the image-like response is the CapsNet model's input and the topology binary image of the device is the output.

class capsule layer, namely

$$\hat{\mathbf{u}}_{j|i}^{(3)} = \mathbf{W}_{i,j,:,:}^{(3)} \cdot \mathbf{u}_h^{(2)}. \tag{4}$$

The product of a "prediction vector" and a coupling coefficient, which represents the degree of relevance, is calculated to obtain the prediction of capsule i to the corresponding higher-level capsule j. Therefore, the input to capsule j will be

$$\mathbf{s}_j^{(3)} = \sum_i c_{i,j} \hat{\mathbf{u}}_{j|i}^{(3)}. \tag{5}$$

Finally, a "squashing" function

$$\mathbf{u}_j^{(3)} = \frac{\left\|\mathbf{s}_j^{(3)}\right\|^2}{1+\left\|\mathbf{s}_j^{(3)}\right\|^2} \frac{\mathbf{s}_j^{(3)}}{\left\|\mathbf{s}_j^{(3)}\right\|}. \tag{6}$$

limits the vector length in the interval of [0,1). As for the update rules of coupling coefficients $\mathbf{c}$ and initial logits $\mathbf{a}$, they are implemented by the dynamic routing mechanism. The coupling coefficients between capsule i and all the capsules in the layer above sum to 1 and are determined by

$$c_{i,j} = \frac{\exp\left(a_{i,j}\right)}{\sum_k \exp\left(a_{i,k}\right)}, \tag{7}$$

where the initial logits $a_{i,j}$ are the log prior probabilities that capsule i should be coupled to capsule j. Then, the initial coupling coefficients are iteratively updated by measuring the agreement between the current output $\mathbf{u}_j^{(3)}$ of each capsule j in the layer above and the prediction $\hat{\mathbf{u}}_{j|i}^{(3)}$ made by capsule i. The agreement is measured by the inner product $\mathbf{u}_j^{(3)} \cdot \hat{\mathbf{u}}_{j|i}^{(3)}$, which can be seen as a log likelihood and is added to $a_{i,j}$ before computing the new values for the coupling coefficients connecting capsule i to the higher level capsule j.

In the inverse modeling process, the structure of the nanophotonic device needs to be designed according to the targeted spectral response. We use the spectral transmission response as the input and the structure matrix as the label. It is solved as a classification problem. Therefore, we use the cross entropy loss to train the CapsNet model. We reshape the spectral response vector with dimension (2,3000) to (2,60,50), and denote it as $\mathbf{y}$. $\mathbf{y}$ is fed to CapsNet and the model finally outputs the topology structure of the devices. The purpose of this operation is adapting to the feature extraction process and making the shape of the feature map more uniform. In addition, we also segment the transmission at different wavelengths, so that through the feature extraction of "image" spectral curve, the model can find the internal relationship between different transmission and the corresponding device structure in a certain region. The overall modeling process is shown in Fig. 2.

III. Results and Discussion

In this section, we present the numerical simulations of both forward prediction and inverse design process. First, we quantitatively compare the performance from multiple indicators of

our CapsNet with two state-of-the-art CNN models. The results show that the modeling process based on CapsNet has better performance than that based on CNNs. Second, we train the CapsNet on different percentage settings of the original training set. From the results, we find that the CapsNet can achieve similar performance of CNNs with only about 60% training data. Third, to study the asymptotic convergence, we plot the training curves of the relative loss. At last, to prove the feasibility of our CapsNet based model, we randomly select some unseen cases in the test set and compare the predicted results with the ground truths. Specifically, we display the outcome of the CapsNet for forward prediction of spectral response from the binary image, and inverse designing the hole position from a given spectral response.

Before presenting the details of our numerical simulations and results, we first describe the specific architectures of the CapsNet model.

- CapsNet model for forward prediction:
  - –ReLU convolutional layer: the kernel size ($C_1 \times C_0 \times K_1 \times K_1$) is $32 \times 1 \times 9 \times 9$, the stride is 1;
  - –PrimaryCaps layer: the kernel size ($C_2 \times C_1 \times K_2 \times K_2$) is $8 \times 32 \times 9 \times 9$, the stride is 4, and the dimension of capsules (d_2) is 8;
  - –DigitCaps layer: the number of capsules (M_3) is 2, and the dimension of capsules (d_3) is 3000.
- CapsNet model for inverse design:
  - –ReLU convolutional layer: the kernel size ($C_1 \times C_0 \times K_1 \times K_1$) is $32 \times 2 \times 9 \times 9$, the stride is 2;
  - –PrimaryCaps layer: the kernel size ($C_2 \times C_1 \times K_2 \times K_2$) is $32 \times 32 \times 9 \times 9$, the stride is 4, and the dimension of capsules (d_2) is 16;
  - –DigitCaps layer: the number of capsules (M_3) is 1352, and the dimension of capsules (d_3) is 1.

More details of the hyperparameters selection strategies for CapsNet can be found in [35]. To compare the performance with CNNs, we choose two popular CNN models AlexNet and VGG. Considering that the capsules in CapsNet are more complex than the neurons and CNNs are often deeper than CapsNet, to make the comparisons as fair as possible, we require that the number of parameters contained in the compared architectures should be comparable. In detail, the specific architectures of AlexNet and VGG we used in this paper are described as follows:

- *AlexNet*: It has eight layers, including five convolutional layers and three fully connected layers. We set the kernel size in each convolutional layers to $48 \times 1 \times 5 \times 5, 128 \times 48 \times 5 \times 5, 192 \times 128 \times 3 \times 3, 128 \times 192 \times 3 \times 3, 48 \times 128 \times 3 \times 3$, and the number of neurons in each fully connected layers to 328, 192, 6000 (forwardly) and 164, 96, 1352 (inversely), respectively. The stride in each convolutional layer is 1.
- *VGG*: We choose VGG8 with the first five convolutional layers of VGG16 and three fully-connected layers. We set the kernel size in each convolutional layers to $64 \times 1 \times 3 \times 3, 64 \times 64 \times 3 \times 3, 128 \times 64 \times 3 \times 3, 128 \times 128 \times 3 \times 3, 48 \times 128 \times 3 \times 3$, and the number of neurons in each fully connected layers to 328, 192, 6000 (forwardly) and 164, 96, 1352 (inversely), respectively. The stride in each convolutional layer is 1.

As for the dataset, we use the DBS optimization algorithm to generate the dataset. The dataset contains 40000 data points. We select 32000 input data for training and 8000 for testing. The split ratio of training set and test set is $4:1$.

In order to quantitatively measure the fitting performance, we introduce the goodness of fit. Goodness of fit of a regression curve reflects how well it fits the observations [47]. The statistics of goodness of fit is R^2. More specifically, suppose $\{y_i\}$ are the data points to be fitted, $\bar{y}$ is the mean of $\{y_i\}$, and $\hat{y}_i$ are the predicted data points. R^2 is calculated by:

$$R^2 = \frac{\sum_i (\hat{y}_i - \bar{y})^2}{\sum_i (y_i - \bar{y})^2} = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{8}$$

Obviously, the closer its value is to 1, the better fitting degree of the regression curve to the observed value. For the forward and inverse modeling process, we list the value of R^2 in Table I. It can be observed that when the training datasets are the same, the performance of CapsNet is much better than the CNNs with comparable number of parameters. Furthermore, we also find that the CapsNet can obtain nearly the same performance when using only 60% of the training data compared with VGG, the better of the two CNNs. This fact illustrates the advantage of CapsNet for saving training data, and demonstrates that the CapsNet model is indeed an advanced model compared with CNNs.

In order to further visualize the error between the predicted output and the target output, we plot the statistical histograms of mean squared error (MSE) of the forward model and the inverse model on the test set in Fig. 3. In detail, for each data point in the test set, the MSE between the predicted spectral response curve and the ground truth (or the targeted) spectral response curve is first calculated. Then the number of data points according to different intervals of MSE is counted. Finally, for each interval, the number of data points and the median corresponding to each interval are multiplied and summed together. Since the number of data sample points in the test set is fixed, the total height of all the intervals is constant. The better the fitting ability of a model is, the more the number of sample points falls in the interval with small MSE. Since the performance on VGG is better than that on AlexNet, here we only set the MSE results of VGG as baseline. From Fig. 3 and the reported results of weighted sum MSE, we can see that the error of forward prediction and inverse design based on CapsNet is smaller than that based on VGG, which demonstrates again that the CapsNet has better performance than CNNs. Although the advantage is limited, we believe that in the design of complex nanophotonic devices using DL approach, data saving is more practical than the substantial improvement in the performance of the model. Since in the process of designing complex nanophotonic device based on neural networks, the use of large amounts of training data is the main bottleneck restricting further developing of this kind of approach. Our proposed CapsNet that can save a lot of training data would have broad application prospects.

The relative loss curves for forward training and inverse training are given in Fig. 4, from which we observe that our

TABLE I
GOODNESS OF FIT COMPARISONS BETWEEN CNNS AND CAPSNETS MODEL FOR FORWARD AND INVERSE TASKS

Models	Forward prediction			Inverse design		
	Port 2	Port 3	Overall	Port 2	Port 3	Overall
AlexNet	0.8941	0.8753	0.8847	0.8313	0.8512	0.8413
VGG	0.9006	0.8768	0.8887	0.8471	0.8496	0.8484
CapsNet	**0.9442**	**0.9348**	**0.9395**	**0.9382**	**0.9656**	**0.9519**
CapsNet (90% training data)	0.9373	0.9226	0.923	0.9184	0.8946	0.9065
CapsNet (80% training data)	0.9266	0.9213	0.924	0.8942	0.8986	0.8964
CapsNet (70% training data)	0.9121	0.9135	0.9128	0.8773	0.8866	0.8821
CapsNet (60% training data)	0.9045	0.9011	0.9028	0.8632	0.8695	0.8664
CapsNet (50% training data)	0.8926	0.8851	0.8889	0.8479	0.8434	0.8457

The best results are highlighted in bold face.

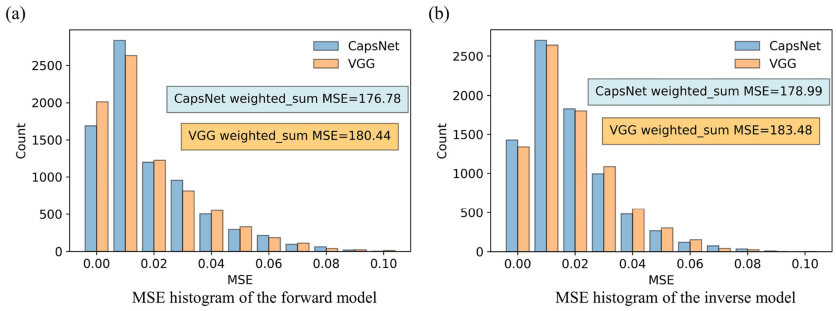

Fig. 3. MSE histograms of the forward model and the inverse model.

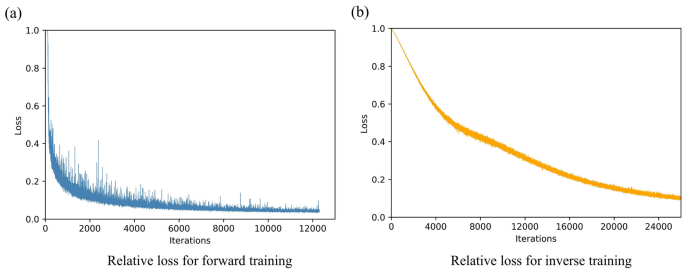

Fig. 4. Relative loss curves for forward training and inverse training.

model can be trained effectively and converge well. The visualizations based on CapsNet modeling process are displayed in Figs. 5 and 6. From Fig. 5, we observe that the model can predict transmission spectra quite accurately, which proves the feasibility of CapsNet to solve the forward prediction task. From Fig. 6, we find that the CapsNet model can indeed capture some local position information. For example, we can see that the overall device structures, especially the top right corner of the first two ground truth devices and inverse designed devices are quite similar. Consequently, their transmission curves fit well, which demonstrates that the CapsNet can inverse design the nanophotonic devices effectively.

In addition, we also observe an interesting phenomenon from the numerical results in this section. Since inverse design is naturally difficult to forward prediction, the effect of the inverse model is worse than that of forward model for almost all the existing approaches, which is also applicable to CNNs and our CapsNet. However, taking the performance of CNN model as

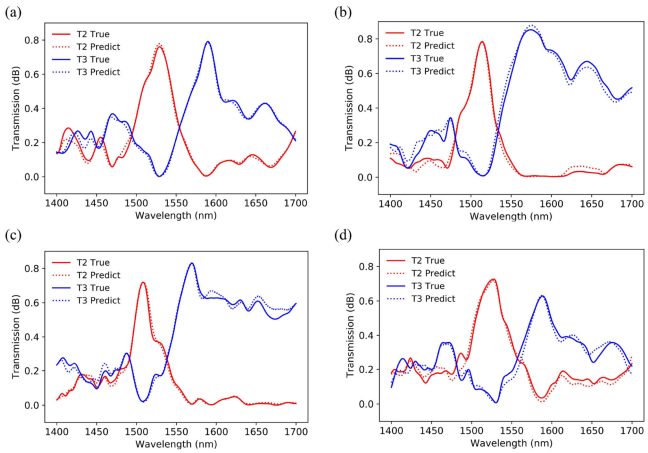

Fig. 5. Comparisons of CapsNet predicted spectral response to the numerically simulated spectral responses on four randomly chosen wavelength demultiplexers.

the benchmark, we are glad to find that the improvement of our new method in inverse design is more obvious than that in forward prediction. For example, in Fig. 3, the value of weighted sum MSE for the forward task is lower than the inverse task, and there are more devices from the orange bars in the low error region (0.00 - 0.01) of the leftmost two columns in (a). Similarly, the inverse modeling process converges slower than the forward process as shown in Fig. 4. This is because the process of inverse designing can make better use of the powerful location information and spatial information capture ability of CapsNet.

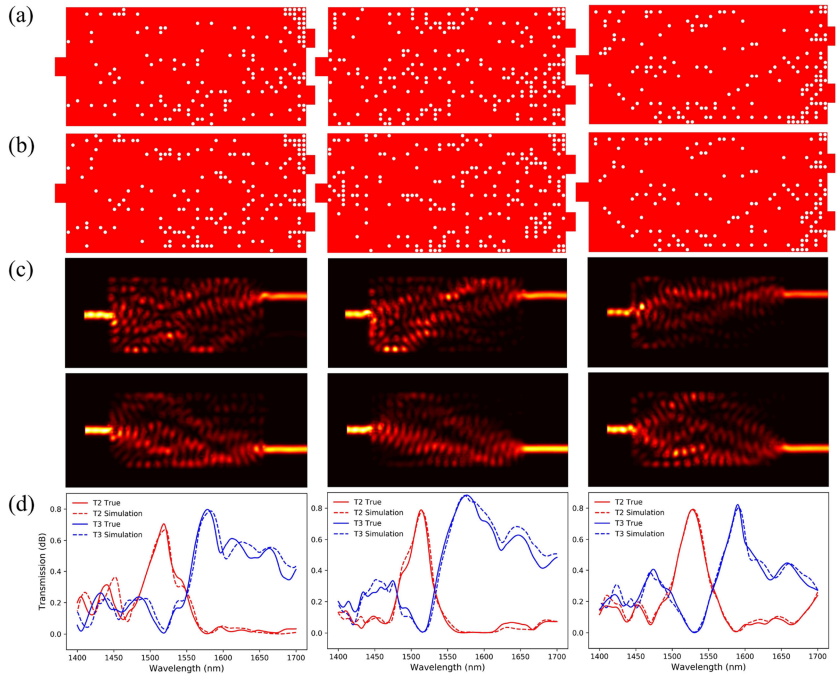

Fig. 6. Comparisons of CapsNet predicted topology structure's spectral responses to the targeted spectral responses on four given wavelength demultiplexers.

IV. Conclusion

In recent years, DL based approach has been applied to the modeling process of nanophotonic devices prediction and designing. Fast forward approximation of spectral response with DL models and DL aided inverse design can be regarded as an alternative to typical numerical methods. Excellent performances based on DL approaches have been reported recently. However, producing the training data of complex devices is a computationally heavy process. If the training data is insufficient, the performance of the DL model would decline. Thus, to reduce the requirement of training data, we utilize CapsNet model to assist forward prediction and inverse design of wavelength demultiplexer in this paper. Extensive numerical results are presented to validate the performance of our CapsNet model. The quantitative comparisons show that our CapsNet model performs better than CNNs in both forward prediction and inverse design problem, and can save approximately 40% training data. From the loss curves of training and the visualization outcomes, we observe that our CapsNet model has good generalization power under a huge design space. Besides, benefiting from the good generalization power of the CapsNet, the proposed DL method can also be applied to forward prediction and inverse design of other components of PICs, including power splitter, polarization beam splitter, mode demultiplexer and so on.

References

[1] T. J. Seok, N. Quack, S. Han, R. S. Muller, and M. C. Wu, "Large-scale broadband digital silicon photonic switches with vertical adiabatic couplers," *Optica*, vol. 3, no. 1, 2016, Art. no. 64.

[2] T. J. Seok, K. Kwon, J. Henriksson, J. Luo, and M. C. Wu, "Wafer-scale silicon photonic switches beyond die size limit," *Optica*, vol. 6, no. 4, pp. 490–494, 2019.

[3] N. C. Harris *et al.*, "Quantum transport simulations in a programmable nanophotonic processor," *Nature Photon.*, vol. 11, pp. 447–452, 2017.

[4] E. Kuramochi *et al.*, "Large-scale integration of wavelength-addressable all-optical memories on a photonic crystal chip," *Nature Photon.*, vol. 8, no. 6, pp. 474–481, 2014.

[5] K. Nozaki *et al.*, "Ultralow-power all-optical ram based on nanocavities," *Nature Photon.*, vol. 6, no. 4, pp. 248–252, 2012.

[6] Y. Shen *et al.*, "Deep learning with coherent nanophotonic circuits," *Nature Photon.*, vol. 11, pp. 441–446, 2017.

[7] J. Wang, S. Paesani, Y. Ding, R. Santagati, and M. G. Thompson, "Multidimensional quantum entanglement with large-scale integrated optics," *Science*, vol. 360, pp. 285–291, 2018.

[8] Z. Wang, T. Li, A. Soman, D. Mao, and T. Gu, "On-chip wavefront shaping with dielectric metasurface," *Nature Commun.*, vol. 10, no. 1, 2019, Art. no. 3547.

[9] Y. Liu *et al.*, "Arbitrarily routed mode-division multiplexed photonic circuits for dense integration," *Nature Commun.*, vol. 10, Jul. 2019, Art. no. 3263.

[10] D. Liu, L. Zhang, H. Jiang, and D. Dai, "First demonstration of an on-chip quadplexer for passive optical network systems," *Photon. Res.*, vol. 9, no. 5, pp. 757–763, May 2021.

[11] W. Ma, F. Cheng, and Y. Liu, "Deep-learning-enabled on-demand design of chiral metamaterials," *ACS Nano*, vol. 12, no. 6, pp. 6326–6334, 2018.

[12] J. Huang *et al.*, "Digital nanophotonics: The highway to the integration of subwavelength-scale photonics," *Nanophotonics*, vol. 10, no. 3, pp. 1011–1030, 2021.

[13] A. Gondarenko and M. Lipson, "Low modal volume dipole-like dielectric slab resonator," *Opt. Exp.*, vol. 16, no. 22, pp. 17689–17694, 2008.

[14] B. Shen, P. Wang, R. Polson, and R. Menon, "An integrated-nanophotonics polarization beamsplitter with $2.4 \times 2.4\ \mu m^2$ footprint," *Nature Photon.*, vol. 9, no. 6, pp. 378–382, 2015.

[15] L. Lu *et al.*, "Inverse-designed single-step-etched colorless 3 dB couplers based on RIE-LAG-insensitive PHC-like subwavelength structures," *Opt. Lett.*, vol. 41, no. 21, pp. 5051–5054, Nov. 2016.

[16] W. Chang *et al.*, "Ultra-compact mode (de) multiplexer based on subwavelength asymmetric y-junction," *Opt. Exp.*, vol. 26, no. 7, pp. 8162–8170, Apr. 2018.

[17] C. Y. Kao, S. Osher, and E. Yablonovitch, "Maximizing band gaps in two-dimensional photonic crystals by using level set methods," *Appl. Phys. B*, vol. 81, no. 2, pp. 235–244, 2005.

[18] K. Wang, X. Ren, W. Chang, L. Lu, D. Liu, and M. Zhang, "Inverse design of digital nanophotonic devices using the adjoint method," *Photon. Res.*, vol. 8, no. 4, pp. 528–533, Apr. 2020.

[19] J. Jensen and O. Sigmund, "Topology optimization for nano-photonics," *Laser Photon. Rev.*, vol. 5, no. 2, pp. 308–321, 2011.

[20] J. Lu and J. Vučković, "Nanophotonic computational design," *Opt. Exp.*, vol. 21, no. 11, pp. 13351–13367, Jun. 2013.

[21] A. Y. Piggott, J. Lu, K. G. Lagoudakis, J. Petykiewicz, T. M. Babinec, and J. Vučković, "Inverse design and demonstration of a compact and broadband on-chip wavelength demultiplexer," *Nature Photon.*, vol. 9, no. 6, pp. 374–377, 2015.

[22] J. Huang *et al.*, "Implementation of on-chip multi-channel focusing wavelength demultiplexer with regularized digital metamaterials," *Nanophotonics*, vol. 9, no. 1, pp. 159–166, 2019.

[23] J. Huang *et al.*, "Ultra-compact broadband polarization beam splitter with strong expansibility," *Photon. Res.*, vol. 6, no. 6, pp. 574–578, Jun. 2018.

[24] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[25] W. Ma, Z. Liu, Z. Kudyshev, A. Boltasseva, W. Cai, and Y. Liu, "Deep learning for the design of photonic structures," *Nature Photon.*, vol. 15, pp. 1–14, Oct. 2020.

[26] D. Liu, Y. Tan, E. Khoram, and Z. Yu, "Training deep neural networks for the inverse design of nanophotonic structures," *ACS Photon.*, vol. 5, no. 4, pp. 1365–1369, 2018.

[27] A. N. I. Malkiel *et al.*, "Plasmonic nanostructure design and characterization via deep learning," *Light: Sci. Appl.*, vol. 7, 2018, Art. no. 60.

[28] J. Peurifoy *et al.*, "Nanophotonic particle simulation and inverse design using artificial neural networks," *Sci. Adv.*, vol. 4, no. 6, 2018, Art. no. eaar4206.

[29] Z. A. Kudyshev, A. V. Kildishev, V. M. Shalaev, and A. Boltasseva, "Machine learning-assisted global optimization of photonic devices," *Nanophotonics*, vol. 10, no. 1, pp. 371–383, 2021.

[30] H. Wang, Z. Zheng, C. Ji, and L. J. Guo, "Automated multi-layer optical design via deep reinforcement learning," *Mach. Learn.: Sci. Technol.*, vol. 2, no. 2, 2021, Art. no. 025013.

[31] A. Sheverdin, F. Monticone, and C. Valagiannopoulos, "Photonic inverse design with neural networks: The case of invisibility in the visible," *Phys. Rev. Appl.*, vol. 14, 2020, Art. no. 024054.

[32] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, vol. 27, pp. 2672–2680.

[33] P. R. Wiecha, A. Arbouet, C. Girard, and O. L. Muskens, "Deep learning in nano-photonics: Inverse design and beyond," *Photon. Res.*, vol. 9, no. 5, pp. B182–B200, 2021.

[34] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. Artif. Neural Netw. Mach. Learn.*, 2011, pp. 44–51.

[35] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3856–3866.

[36] A. Shahroudnejad, P. Afshar, K. N. Plataniotis, and A. Mohammadi, "Improved explainability of capsule networks: Relevance path by agreement," in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, 2018, pp. 549–553.

[37] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–15.

[38] X. Zhang and L. Chen, "Capsule graph neural network," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–16.

[39] A. Jaiswal, W. AbdAlmageed, Y. Wu, and P. Natarajan, "CapsuleGAN: Generative adversarial capsule network," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2018, pp. 526–535.

[40] L. Zhang, M. Edraki, and G. Qi, "CapproNet: Deep feature learning via orthogonal projections onto capsule subspaces," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5814–5823.

[41] D. Wang and Q. Liu, "An optimization view on dynamic routing between capsules," in *Proc. Int. Conf. Learn. Representations Workshops*, 2018, pp. 1–4.

[42] K. Duarte, Y. Rawat, and M. Shah, "VideoCapsuleNet: A simplified network for action detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7610–7619.

[43] P. Afshar, A. Mohammadi, and K. N. Plataniotis, "Brain tumor type classification via capsule networks," in *Proc. IEEE Int. Conf. Image Process.*, 2018, pp. 3129–3133.

[44] T. Iesmantas and R. Alzbutas, "Convolutional capsule network for classification of breast cancer histology images," in *Proc. Int. Conf. Image Anal. Recognit.*, 2018, pp. 853–860.

[45] Y. Du, X. Zhao, M. He, and W. Guo, "A novel capsule based hybrid neural network for sentiment classification," *IEEE Access*, vol. 7, pp. 39321–39328, 2019.

[46] M. Yang *et al.*, "Investigating capsule networks with dynamic routing for text classification," in *Proc. Int. Conf. Empir. Methods Natural Lang. Process.*, 2018, pp. 3110–3119.

[47] D. R. Cox, *Principles of Statistical Inference*. Cambridge, U.K.: Cambridge Univ. Press, 2006.