MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Machine Learning for design and optimization of photonic devices

Kojima, Keisuke; Tang, Yingheng; Wang, Ye; Koike-Akino, Toshiaki

TR2021-142 January 08, 2022

Abstract

This chapter discusses how we can apply deep learning to photonic device design problems, by focusing on three types of modeling approaches. In forward modeling, the trained deep neural network (DNN) is integrated in the optimization loops of classical optimizers. In inverse modeling, the trained DNN directly constructs a structure with the desired target performance given as input. Generative modeling produces a series of optimized designs for a target performance. We take an example of designing compact silicon-on-insulator (SOI)-based 1 x 2 power splitters with various target splitting ratios, which can be a building block for various integrated photonic circuits. We also give an overview of their applications to metasurfaces/plasmonics, other types of photonic devices in these three categories. These approaches are expected to be applicable to a broad range of design for various types of photonic devices.

Machine Learning for Future Fiber Optic Communication Systems (Elsevier Book) 2021

© 2022Licensed under the Creative Commons BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/.

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Chapter 1

Machine Learning for design and optimization of photonic devices

Keisuke Kojima
Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139, USA.

Toshiaki Koike-Akino
Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139, USA.

Yingheng Tang
Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139, USA.
Electrical and Computer Engineering Dept., Purdue University, West Lafeyette, IN 47907, USA.

Ye Wang
Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139, USA.

Matt Brand
Mitsubishi Electric Research Laboratories (MERL), 201 Broadway, Cambridge, MA 02139, USA.

ABSTRACT
This chapter discusses how we can apply deep learning to photonic device design problems, by focusing on three types of modeling approaches. In forward modeling, the trained deep neural network (DNN) is integrated in the optimization loops of classical optimizers. In inverse modeling, the trained DNN directly constructs a structure with the desired target performance given as input. Generative modeling produces a series of

optimized designs for a target performance. We take an example of designing compact silicon-on-insulator (SOI)-based 1×2 power splitters with various target splitting ratios, which can be a building block for various integrated photonic circuits. We also give an overview of their applications to metasurfaces/plasmonics, other types of photonic devices in these three categories. These approaches are expected to be applicable to a broad range of design for various types of photonic devices.

KEYWORDS
Nanophotonics, Beam Splitter, Metasurface, Plasmonics, Deep Learning, Neural Network, Convolutional Neural Networks, Autoencoder, CVAE, Generative Model

1.1 INTRODUCTION

Subwavelength nanostructured materials can be used to control incident electromagnetic (EM) fields into specific transmitted and reflected wavefronts. Recent nanophotonic devices have used such complex structures to enable novel applications in optics, integrated photonics, sensing, and computational metamaterials in a compact and energy-efficient form. These include waveguide-type nanophotonic devices [1–9] and metasurfaces including plasmonics [10–17]. Optimizing nanostructures with a large number of possible combinations of parameters is a challenging task in practice.

EM simulations, including finite-difference time-domain (FDTD) methods and finite element methods (FEM), often require long simulation time, several minutes to hours depending on the size and the mesh of the photonic device, in order to estimate the optical transmission and/or reflection response. For designing nanostructures to achieve a target transmission/reflection profile, a large number of EM simulations needs to be performed, e.g., in a meta-heuristic manner. To resolve the issue, some optimization methods such as direct binary search (DBS) [4] and adjoint method [1,3,18,19] have been implemented. More recently, artificial intelligence using neural networks (NN) has been integrated in an optimization process that can accelerate optimization by reducing the required number of numerical simulations. For example, in [20], we demonstrated how NNs can help to streamline the design process, although the prediction accuracy was limited, due to the shallow network structure.

Deep learning methods are representation-learning techniques obtained by composition of non-linear models that transform the representation at the previous level into a higher and slightly more abstract level in a hierarchical manner [21]. The main idea is that by cascading a large number of such transformations, nearly arbitrary complex functions can be learned in a data-driven fashion using deep neural networks (DNN) [22]. The huge success of deep learning in modeling complex input-output relationship has attracted attention from several scientific communities such as material discovery and design [23–26], high energy physics [27], single molecule imaging [28], medical diagnosis [29], and particle physics [30]. The optical community also started working on signal processing and network automation of optical fiber communications [31–36], and inverse modeling for design of nanostructured optical components using DNN [37–42] Also, there have been reports on optical implementation of artificial neural networks [43–48] as well as characterization of optical pulses [49].

More recently, Liu *et al.* used a tandem NN architecture to learn non-unique EM scattering of alternating dielectric thin films with varying thickness [37]. Peurifoy *et al.* demonstrated DNNs to approximate light scattering of multilayer shell nanoparticles of SiO_2 and TiO_2 using a fully connected DNNs [40]. Asano and Noda provided an NN for prediction of the quality factor in two dimensional photonic crystals [42]. Hammond *et al.* used DNNs for forward and inverse modeling of strip waveguides and Bragg gratings [50]. Hedge paired DNN with evolutionary algorithms to accelerate antireflection coating designs [51]. Banerji *et al.* used reinforcement learning to design nanophotonic power splitters [52]. The design space for integrated photonic devices is considerably larger than previously demonstrated optical scattering applications, and calls for robust deeper networks such as Convolutional Neural Networks (CNN) [22].

Integrated photonic beam splitters have been widely used to equally divide the power into the output ports. Although an arbitrary split ratio can be applied in various applications such as signal monitoring, feedback circuits, or optical quantization [53], the design space is hardly explored due to design complexity. To design photonic power splitters with an arbitrary splitting ratio, the designer often begins with an overall structure based on analytical models and fine tunes the structure using parameter sweeps in numerical simulations. DNN can dramatically change the design process.

Metasurfaces and plasmonics are expected to play a major role in ultra-compact and multi-functional lenses and various other applications. Possible applications for fiber-optics include quarter-wave plates, vortex plates to create orbital angular momentum (OAM), and plasmonic collimators [54]. However, designing metasurfaces and plasmonic devices also has the same issue of vast design space and time consuming EM simulations. Including DNNs in the design process will significantly advance the field.

DNNs can be used to predict an optical response of a topology (Forward Modeling) as well as to design a topology given a desired optical response (Inverse Modeling). Another class of DNNs for designing devices and materials is a generative DNN model (Generative Modeling). This paper first overviews these three categories of DNNs applied to designing/optimizing nanophotonic power splitters. We demonstrate that by using deep learning methods, we could efficiently learn the design space of a broadband integrated photonic power splitter to realize a compact device. We also review parallel activities for metasurfaces and plasmomics. The general concept of using DNNs for designing/optimizing photonic devices will be readily applicable to a broader area of photonics.

1.2 DEEP NEURAL NETWORK (DNN) MODELS

Fig. 1.1 shows the three categories or methodologies of the DNN models for designing photonic devices; i.e., forward, inverse, and generative modeling.

The first category is forward modeling, where the DNN takes the device topology as an input and the optical response is the output, as shown in Fig. 1.1 (a). In this case, for inverse design, we would use the trained DNN in surrogate optimization, e.g., optimization metric calculation in meta-heuristic optimizers such as DBS and evolutionary strategies (ES). The forward-modeling DNN can skip time-consuming EM simulations to predict the optical response. Let $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^m$ be the topology parameters to design and the corresponding optical response, respectively. Given a device parameter $\boldsymbol{x}$, the corresponding response $\boldsymbol{y}$ can be analyzed through the EM simulation. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be such a nonlinear function, transforming the geometry parameter $\boldsymbol{x}$ to the response $\boldsymbol{y}$ as $\boldsymbol{y} = f(\boldsymbol{x})$. The forward-modeling DNN approximates the function f with a parameterized function f_θ such that $\hat{\boldsymbol{y}} = f_\theta(\boldsymbol{x})$ is close to the true $\boldsymbol{y}$. Because of the universal approximation theorem of DNN [55], the forward modeling can predict the response arbitrarily well as the size of DNN and training samples increases. Nevertheless, this forward-modeling DNN does not directly provide an optimized topology and, thus, it is less convenient for inverse design in general.

The second category is inverse modeling, where the optical response is the input and the device topology is the output, as shown in Fig. 1.1 (b). Letting $f^{-1} : \mathbb{R}^m \to \mathbb{R}^n$ be the inverse function of f, the inverse-modeling DNN tries to approximate f^{-1} such that its parameterized function $g_\phi : \mathbb{R}^m \to \mathbb{R}^n$ produces the output of $\hat{\boldsymbol{x}} = g_\phi(\boldsymbol{y})$ close enough to the original geometry parameter $\boldsymbol{x}$ given the optical response $\boldsymbol{y}$. In this case, using the trained network, when the target optical response is given, the desired device topology is directly generated. This inverse modeling can be more useful for inverse design than forward modeling as we can directly obtain device topology given a target profile. However, it typically requires more diverse training data as the design space is usually much larger than the diversity of the optical profile (specifically, $\mathbb{H}(\boldsymbol{x}) \gg \mathbb{H}(\boldsymbol{y})$, where $\mathbb{H}$ denotes entropy, i.e., amount of information) and there is no guarantee that target profile is feasible. In addition, the inverse-modeling DNN usually provides only one topology candidate per target response, which limits the capability to explore better geometries. In fact, the f-function through EM simulations is not always bijective (even non-injective and non-surjective) and, hence, inverse modeling for any target profile $\boldsymbol{y}$ to obtain a single $\boldsymbol{x}$ is challenging (potentially infeasible in a strict sense).

The third category is generative modeling such as conditional autoencoders (CAE) and generative adversarial networks (GAN), wherein the network is trained using the device topology and optical response. Once the network is trained, we give a target optical response as well as random numbers, and the network generates a series of improved topologies. Fig. 1.1 (c) illustrates conditional variational autoencoder framework, where the encoder summarizes the geometry $\boldsymbol{x}$ into a latent representation and the decoder takes as input the randomized variables of the latent representation along with the target profile $\boldsymbol{y}$ as a condition to generate the geometry $\hat{\boldsymbol{x}}$. Omitting the encoder to produce the latent variables, the decoder can be used alone with random latent variable inputs in a manner similar to sample generation with the generator block of a conditional GAN. When we further simplify it by excluding randomness and adversarial training, this model reduces to the inverse modeling. The generative modeling methods can resolve the issues of the forward and inverse modeling approaches as it can directly generate a series of inversely designed geometries.

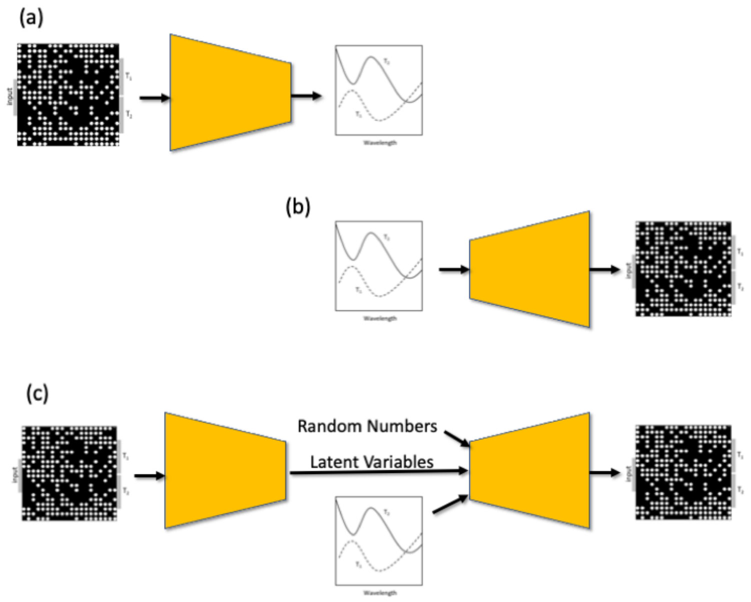


FIGURE 1.1 Three distinct categories of network models: (a) forward modeling, where the device topology is the input and the optical response is the output, (b) inverse modeling, where the optical response is the input and the device topology is the output, and (c) generative modeling, where the network is first trained with pairs of the device topology and the optical response, and later a new device topology is generated with the optical response and random numbers.

1.3 NANOPHOTONIC POWER SPLITTER

1.3.1 Device Structure

Integrated photonic beam splitters have been widely used to equally divide the power into the output ports. Although an arbitrary split ratio can be applied in various applications, the design space is hardly explored due to design complexity. Tian *et al.* [56] demonstrated a silicon-on-insulator (SOI)-based 1×3 coupler with variable splitting ratio in a 15×15 μm^2 device footprint with 60 nm wavelength range and 80% transmission efficiency. Xu *et al.* [57] optimized positioning of squared etched pixels to achieve 80% efficiency for arbitrary ratio power dividers in a 3.6×3.6 μm^2 device footprint. To design photonic power splitters with an arbitrary splitting ratio, the designer often begins with an overall structure based on analytical models and fine tunes the structure using parameter sweeps in numerical simulations. In this section, we consider a nanostructured power splitter with an arbitrary and fixed splitting ratio towards two output ports, targeting flat response with low insertion loss. Such a power splitter can be a building block of many types of photonic integrated circuits for optical communications and various other applications.

We use an SOI-based structure with one input and two output ports having 0.5μm wide waveguides connected using a taper to the 2.25μm wide square power splitter design as shown in Fig. 1.2. Each hole is a circle with a maximum diameter of 72 nm that is easily fabricable using well-established lithography methods [19,58]. In order to comply with the fabrication limit, we also put a constraint that the minimum hole diameter is 40 nm, when continous variables are used for the design process as in Section 1.3.5.

1.3.2 Simulation and DNN Modeling Procedures

Lumerical FDTD simulations are used to generate labeled data for training, where Lumerical script language is used to modify the device topology when training data is generated on a cluster. On the other hand, Python or Matlab automation is used to modify the device topology for FDTD simulation, when the DNN is included in the optimization loop or active learning. This is only because it is difficult to run Python or Matlab script to automate FDTD simulation on a cluster in our environment. The fundamental transverse electric (TE) mode is used at input source and TE mode output power is recorded for transmission and reflection. It is verified that mode conversion from TE mode to transverse magnetic (TM) mode is negligible ($< 10^{-5}$).

The data for each structure consists of a hole vector (HV) of size 20×20 and labels of its spectral response (SPEC) at the two output ports and reflection at the input port whose size is 63 (3 data sets $\times$ 21 frequency points). In the HV, each hole can be represented by a binary state of 1 for etched (n_{Silicon}) and 0 for not etched (n_{Silica}) for the experiments described in Section 1.3.3 and 1.3.4, or continuous variables associated with hole area relative to the maximum size of 72 nm for the experiments described in Section 1.3.5. In order to comply with the fabrication limit, we also put a constraint that the minimum hole diameter is 40 nm, corresponding to the HV value of $40^2/72^2 = 0.31$. Hence, if the HV value is below 0.31, there is no hole. We use random patterned initial HVs and optimize them using heuristic optimization approaches for various optimization metrics to collect a diverse set of labeled training data for supervised learning. Even though each DBS optimization process itself is sequential, multiple DBS processes are run in parallel in a computer cluster, to accelerate the training data collection.

We use the open source machine learning framework of Keras using Tensorflow as a backend in Section 1.3.3, and PyTorch in Sections 1.3.4 and 1.3.5 in the Python language to build and test our DNNs. For the forward modeling case, the DNN needs to be invoked for training and mutiple design outputs prior to each FDTD run ($>$ 1,000 times for one optimization run), and Tensorflow/Keras has the advantage of shorter start-up time. On the other hand, for inverse and generative modeling, only a few start-ups are needed for the whole design process, and the choice of the framework is more like a historical or personal choice. The training data are generated by FDTD simulations using a high-performance

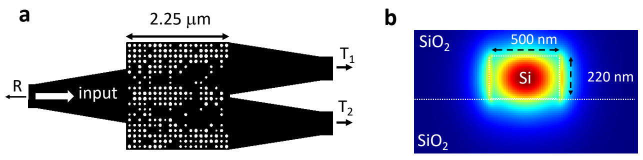


FIGURE 1.2 Schematic of the SOI-based power splitter. a) Top view, where T_1 and T_2 denote the modal transmissions of output ports 1 and 2, and R denotes the reflection at input port. b) Cross-section of the input/output waveguide [59]. By optimizing the binary sequence of positions of etch hole it is possible to adjust light propagation into either of the ports. In Section 1.3.5, continuous variables are used to represent the variable hole sizes.

computing cluster with more than 100 processors. The DNN training, testing, and subsequent FDTD simulations are conducted on a computer with a graphics processing unit (GPU) board Nvidia GeForce GTX Titan Z (12GB memory).

1.3.3 Deep Learning for Forward Modeling to Predict Optical Response

We first describe the forward regression model, using a DNN to predict the transmission and reflection spectra. Given the two-dimensional array (20×20), which is the binary image for the hole locations, we train a DNN using the corresponding transmission and reflection spectra vector which is 63-dimensional, consisting of spectral data for transmission at both port 1, port 2, and reflection at the input port each at 21 discrete wavelengths from 1300 to 1800 nm. Once the DNN is trained, it is used as the predicted (also called “surrogate”) spectra within a DBS optimization loop.

We initially used a fully-connected DNN with multiple layers where each layer has 100 neurons. The number of layers was considered as a hyperparameter which was optimized during the numerical experiments. However, we found that increasing the depth of the fully-connected DNNs did not improve the performance of the network. To achieve better results for deeper networks, we used a residual network (ResNet) to train both the forward and inverse problems as a next step [60].

Since the nanophotonic and photonic crystal designs are analogous to image processing and recognition problems, CNNs [22] are proposed to improve the forward prediction accuracy [42,61]. The 2D CNN architecture is shown in Fig. 1.3, where we treat the input HV data as two-dimentional images.

In our earlier work, we compared the generalization capability of CNN and fully-connected deep neural network (FCDNN) [61]. Fig. 1.4 (a) shows the target and predicted total transmittance of CNN and FCDNN, when the tested total transmission is lower than 0.4, i.e., training data and the testing data are generally very far. It is shown that the predicted total transmittance using CNN is much better correlated with the target values. Fig. 1.4 (b) shows that the

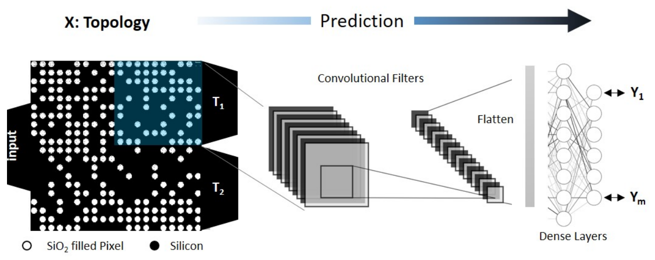


FIGURE 1.3 Convolutional neural network (CNN) architecture [61].

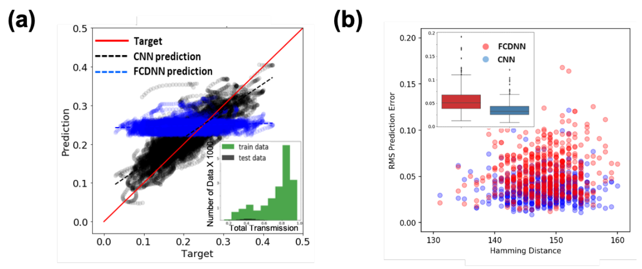


FIGURE 1.4 (a) Target and predicted total transmittance of CNN and FCDNN, and (b) RMS prediction error vs. hamming distance of CNN and FCDNN [61].

root mean square (RMS) error of predicted total transmittance as a function of Hamming distance (number of holes with different binary values), and CNN shows much smaller RMS error compared to FCDNN. These indicate that 2D CNN achieves superior performance when the input data possess image-like properties.

Our forward modeling network consists of three 2D convolutional layers followed by one fully connected layer. The three convolutional layers have 64, 128, and 256 filters each with kernel sizes of 10×10, 5×5, and 4×4, and strides of 1×1, 2×2, and 2×2, respectively, with the Rectified Linear Unit (ReLU) activation function. The fully connected layer has 256 neurons with the sigmoid activation function [62].

In the network training process, we minimized the mean-square error (MSE) as follows:

$$\text{MSE} = \frac{1}{N}\sum_{\lambda=\lambda_{\min}}^{\lambda_{\max}}\left[\left|T_1(\lambda)-T_1^{\star}(\lambda)\right|^2+\left|T_2(\lambda)-T_2^{\star}(\lambda)\right|^2+\left|R(\lambda)-R^{\star}(\lambda)\right|^2\right], \tag{1.1}$$

where $T_1(\lambda)$, $T_2(\lambda)$ and $R(\lambda)$ denote the transmission at the output ports 1 and 2 and reflection at the input port at a wavelegth of λ, respectively, and $N = 21$ is the total number of spectral points. We let $[\cdot]^{\star}$ denote the corresponding target values. Here, we take a sum across uniformly sampled wavelengths λ from a minimum wavelength $\lambda_{\min}$ to a maximum $\lambda_{\max}$. In the DBS optimization process, we minimized the following loss function:

$$\text{Metric} = \left|T_1 - T_1^\star\right|^2 + \left|T_2 - T_2^\star\right|^2 + \alpha\left|R - R^\star\right|^2, \tag{1.2}$$

where T_1 and T_2 are the lowest transmitted power within the spectral range of 1300 nm and 1800 nm, while R is the largest reflection power. We chose $\alpha = 10$ as a weighting factor. We start with random patterns of 20×20 binary HVs, and generated about 11,000 training data by using the standard DBS, with target splitting ratios of 0 (0 : 10), 0.2 (2 : 8), 0.35 (3.5 : 6.5), and 0.5 (5 : 5). Blue points in Fig. 1.5 show the training data, where the total transmission $(T_1 + T_2)$ is plotted against the splitting ratio $T_1/(T_1 + T_2)$. Due to the symmetry, the actual number of distinct data used in the training is about 22,000. We then try to design a power splitter with $T_1^\star = 0.27$ and $T_2^\star = 0.73$, starting from the three initial conditions indicated by the red circles in Fig. 1.5.

In the conventional DBS process, starting with the lower left corner hole, we flip the binary value to evaluate the metric using the FDTD simulation. When the new metric is lower, we choose the new binary value, or if the metric is higher, we revert back to the original binary value. Then, we continue with a sequential application of the same procedure to the other holes.

We use the forward DNN modeling to accelerate the DBS process. In the DNN-assisted DBS, we first train the DNN with 300 epochs using the initial training data, which takes about an hour on the computer with the GPU board. Next we virtually flip each of the 400 holes, using the output of the DNN as a predicted SPEC value. We then select the a new pattern with a flipped hole corresponding to the lowest metric and verify the spectra via an FDTD simulation. When the actual FDTD result is better as expected, the flipped pattern is retained, unless otherwise, we try another hole corresponding to the next best metric and verify with an FDTD simulation. We train with one epoch of the original training data and 15 epochs of the accumulated newly acquired data. This is essentially an accelerated DBS using metric values predicted by the DNN.

Fig. 1.6 shows a comparison between a conventional DBS and a DNN-assisted DBS (denoted as DL-DBS), plotting the metric as a function of the number of FDTD runs. It is confirmed that the DNN-assisted DBS optimizes the device structure faster than the conventional DBS and leads to better device designs overall. Note that each FDTD run takes about two minutes, while the additional training (active learning) for each FDTD run takes 20 seconds. Hence, there is an overhead of about 20% per FDTD. Fig. 1.7 shows an example of the optimized device via the DNN-assisted DBS, where the spectral response is very flat across a wide range of wavelengths at least from 1300 to 1800 nm. The total transmittance greater than 87.5% is achieved with a low reflection below 0.5%.

In this section, we verified that the forward modeling is effective to accelerate the optimization process to design high-performance devices. The key is to take advantage of the very fast forward model computation to predict performance of numerous candidates, which can save time-consuming EM simulations.

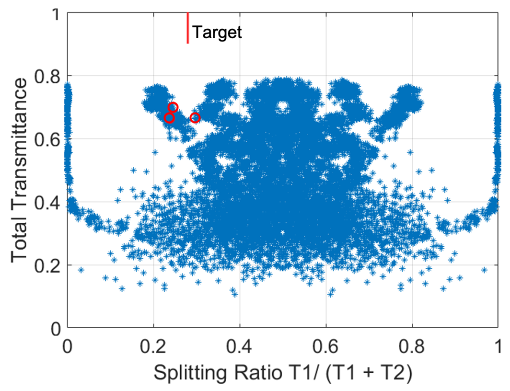


FIGURE 1.5 Total transmittance of 22,000 training data as a function of the splitting ratio. The red circles indicate the three initial conditions used for training the forward regression model, and the line shows the target splitting.

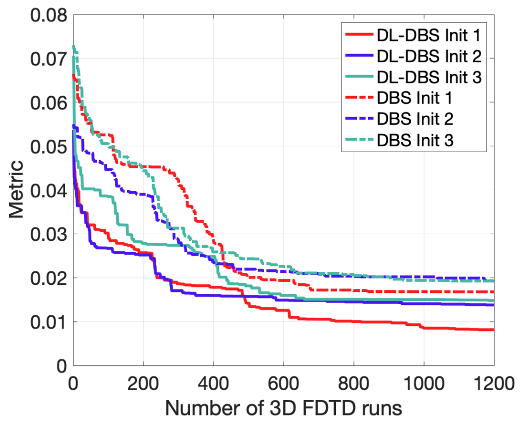


FIGURE 1.6 Metric as a function of the number of 3D FDTD runs, for the conventional DBS and the DNN-assisted DBS methods. Three different initial conditions are used [62].

1.3.4 Deep Learning for Inverse Modeling to Construct Device Topology

The forward modeling described in the previous section is generally good enough to predict the performance given the device topology. However, it needs to be combined with an external optimization method such as DBS or other meta-heuristic algorithms to explore different device topologies. In contrast, the

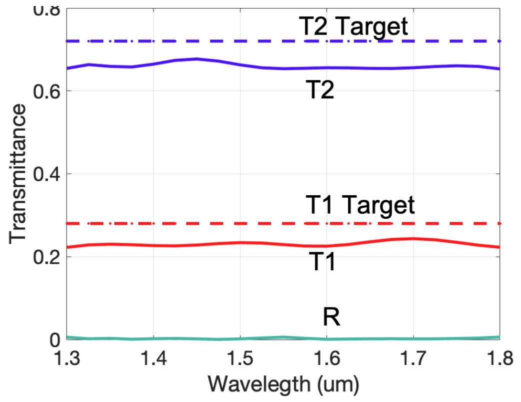


FIGURE 1.7 Example of a 1 : 3 power splitter designed by the forward regression model. For the range of 1300 nm and 1800 nm, the sum of the worst case T_1 and T_2 was 87.5%, while the worst case reflection was less than 0.5%.

inverse modeling takes the spectral response as the input and directly generates the device topology as the output, as shown in Fig. 1.1 (b).

We here consider an inverse modeling DNN consisting of three fully connected layers and two 2D deconvolution layers. The deconvolutional layers, not included in our original inverse modeling paper [60], improves the performance. Note that if we use very small number of performance metrics such as only transmitted power at a single wavelength, we will face a severe issue of non-bijectivity for the f-function, where multiple geometries $\boldsymbol{x}$ may end up having the identical performance $\boldsymbol{y}$, making the inverse design challenging as we discussed. Therefore, it is desirable to have a relatively large number of performance metrics (in our case $m = 63$) to increase the entropy of $\mathbb{H}(\boldsymbol{y})$ in comparison to $\mathbb{H}(\boldsymbol{x})$ for the HV dimension of $n = 400$.

The DNN is trained to predict HV in favor of minimizing the binary cross-entropy (BCE) loss as follows:

$$\text{BCE} = -\sum_{i=1}^{n} \left[x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i) \right], \tag{1.3}$$

where n is the maximum number of holes, x_i denotes the ith HV value of the training data, and $\hat{x}_i$ is the output of DNN as the corresponding estimate of x_i. The predicted HVs $\hat{x}_i$ can take any value from 0 to 1 from a Bernoulli distribution classifier. The classification tends to converge to either 0 or 1 as the loss reduces by increasing the number of training epochs. We train the network using the same data as used in Section 1.3.3.

To test the generalization capabilities of the network, we investigate the design performance on arbitrary and unseen target cases. We try to optimize the device at around a target splitting ratio of 0.27 as an example, where there are no good samples in the training dataset, as shown in Fig. 1.8. In the design stage, the target splitting ratios are chosen to be 0.23, 0.24, . . . , 0.32, and the total transmittances are 0.80, 0.82, 0.84, and 0.86, and hence we obtain 40 combinations of input spectrum to feed in the inverse-modeling DNN. The DNN output are quantized with a threshold of 0.5, and the binary sequence is then fed back into the FDTD solver. The results are shown as the red circles in Fig. 1.8. In this first round, some data points fill the gap, while many are overlapping with the original training data clouds.

In the second run, we add these 40 new data points to the training data, retrain the network, and repeat this active learning process. The results are shown as the dark blue triangles, and those from the third round are indicated by the green squares in Fig. 1.8. These results show that the inverse modeling has the capability of generating the unseen results out of the training data, and the results further improve with active learning.

In this process, the training takes about half an hour at each round on a high-performance computer with a GPU board. Once trained, the inverse design process is instantaneous (less than 1 s) for 40 devices.

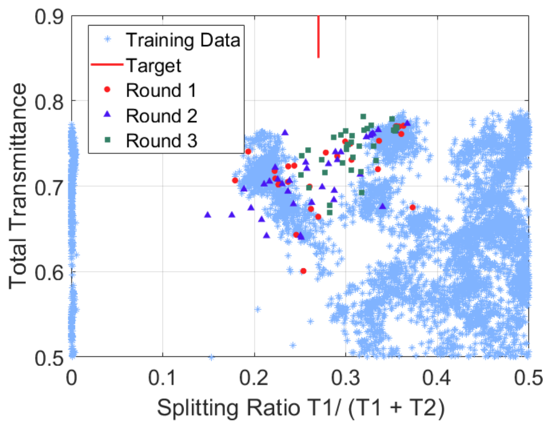


FIGURE 1.8 Demonstration of inverse design for a power splitter to fill gap around a splitting ratio of 0.27. The light blue asterisks, red circles, dark blue triangles, and green squares the training data, the first, the second, and the third round results, where the total transmittance is plotted against the splitting ratio. The red line denotes the target splitting ratio of 0.27 [62].

1.3.5 Deep Learning for Generative Modeling to Produce Device Topology Candidates

The inverse modeling has a potential to generate an unseen good device structure achieving near the target SPEC. However, the generated topology is usually deterministic given the desired SPEC input. Although we can still employ a stochastic variational sampling at the output nodes according to the Bernoulli distribution, the variation of the topology candidates tends to be limited. This can significantly constrain the capability to explore different candidates of topology to achieve improved devices. Besides the forward and inverse modeling, another methodology based on generative modeling has been proposed for photonic devices [37,38,59,63–66]. The generative network can produce a series of improved designs from the training data, based on random number sampling, in a more explicit and systematic way. We have applied generative deep learning models based on an improved version of CVAE to generate integrated photonics devices [66], and later added a concept of active learning to further improve the performance [59].

The CVAE network models the distribution of the device topologies associated with target spectrum characteristics. In our power splitter application, we define the topology pattern using variable-size holes (rather than binary holes). Such pattern has high probability to perform better at light guiding and more stable spectrum response. Our device footprints are $2.25 \times 2.25 \mu\text{m}^2$ as previously described in Fig. 1.2.

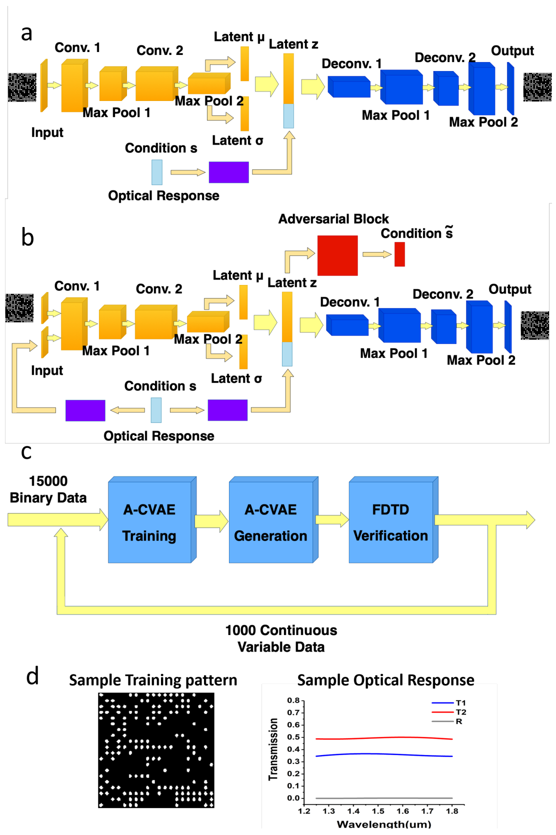


FIGURE 1.9 a) The CVAE model structure, wherein the input is the 20×20 hole vector. The first convolutional layer has 16 channels with a kernel size of 3. The second convolutional layer has 32 channels with a kernel of 3. The condition s is the transmission and reflection spectra obtained from the FDTD simulation to form a 3×21 matrix, and is fed to the latent variable through a fully connected (FC) layer. b) The A-CVAE model structure, wherein the input right now becomes the two channels of 20×20 hole vector, where the first channel is the hole vector of the device and the second channel is the decoded spectra data. The main difference from a) is that one adversarial block is added, which is composed by two FC layers. c) The active learning method added for the A-CVAE method, where 1000 newly generated continuous variable data are added to the original 15,000 binary training data. d) One sample training data, wherein the left figure is the pattern (hole vector) and the right figure shows the three spectra of the training device (the transmission for the two ports and the reflection). Each spectra response has 21 data points, which will be fed into the network as the optical response.

We first constructed a conventional CVAE [67] as shown in Fig. 1.9 (a). The original HVs (20×20) are passed to two convolutional layers [61] and reduced to two sets of intermediate parameters of a probability density function (PDF), representing the means $\mu := (\mu_1, \ldots, \mu_J)$ and standard deviations $\sigma := (\sigma_1, \ldots, \sigma_J)$. In order to have a complete back propagation flow for the network, the reparameterization trick is applied, described by the following equation:

$$z_i = \mu_i + \sigma_i \varepsilon_i, \tag{1.4}$$

where ε_i are independent and identical distributed samples drawn from the standard Gaussian distribution. After reparameterization, the latent variable $z := (z_1, \ldots, z_J)$ is concatenated with the encoded condition parameter s (the dimension reduced from the original 60 to 9 through one fully connected layer) to deconvolute back to the HV. The loss function for this conventional CVAE is constructed by two parts: a) a cross-entropy loss between the original HV $x := (x_1, \ldots, x_n)$ and the decoded HV $y := (y_1, \ldots, y_n)$, and the Kullback–Leibler (KL) divergence between the latent variables and the Gaussian prior. The first term is to reconstruct the original structure as accurate as possible. The second term is to make the latent variable as close to the standard normal distribution($\mathcal{N}(0, I)$) as possible. The equation of loss function is shown as follows:

$$\begin{aligned}\mathsf{Loss} = &-\sum_{i=1}^{n}\Big[y_i \log x_i + (1-y_i)\log(1-x_i)\Big] \\ &+\frac{1}{2}\sum_{j=1}^{J}\Big[\mu_j^2 + \sigma_j^2 - \log(\sigma_j^2) - 1\Big].\end{aligned} \tag{1.5}$$

For device generation, the trained decoder of the CVAE model is used with the desired condition along with a latent variable sampled from the normal distribution $\mathcal{N}(0, I)$, by which a series of HV topology candidates are generated. The HV will be expressed as hole structure on the MMI with different radius. However, the power splitters generated by CVAE have a total transmission of $\sim 80\%$, which still has quite some room for improvement. We believe the reason is that the latent variable in CVAE tends to be correlated with the condition SPEC s, which will result in degradation of the device performance for the pattern generation because random latent space sampling can adversely impact the target spectra.

To address such a problem of the conventional CVAE model, an Adversarial CVAE (A-CVAE) is introduced as shown in Fig. 1.9 (c), where a separate branch to the adversary block is used for isolating the latent variable z from the nuisance variations s (the target SPEC) [68–70]. Our A-CVAE model also has two convolutional layers both for the encoder and the decoder networks(similar to the CVAE model). The encoder is followed by one fully connected layer to obtain the latent variable. The number of channels for the two layers are 16 and 32, and the max pooling stride is 2, after that there is one fully connected layer to reduce the number of the latent variable to 63. The latent variables are then concatenated with the SPEC performance data and is fed into the decoder to generate the output HV. The validations are calculated by using the FDTD simulation to verify a figure of merit (FOM) of generated patterns, where the

FOM is calculated by:

$$\mathsf{FOM} = 1 - 10 \sum_{\lambda=\lambda_{\min}}^{\lambda_{\max}} \left[\left|T_1(\lambda) - T_1^{\star}(\lambda)\right|^2 + \left|T_2(\lambda) - T_2^{\star}(\lambda)\right|^2 + \alpha \left|R(\lambda) - R^{\star}(\lambda)\right|^2 \right], \tag{1.6}$$

where $\alpha = 4$ is used as a weighting factor to balance between the contributions from transmission and the reflection. We take the average of FOM over the FDTD simulation spectral range. As the SPEC performance approaches the target, the FOM increases towards 1, in which case we obtain an ideal power splitter without excess loss for $R^{\star}(\lambda) = 0$ and $T_1^{\star}(\lambda) + T_2^{\star}(\lambda) = 1$.

We use an encoder structure feeding the performance SPEC s projected on a 20×20 matrix which is combine with the original 20×20 hole vector to form a 2-channel input, then process it through two convolution layers. In A-CVAE, the latent z variable will also be fed into an adversarial block to estimate the SPEC $\bar{s} := (\bar{s}_1, \ldots, \bar{s}_n)$. The loss function for the A-CVAE model is shown as follows:

$$\begin{aligned} \mathsf{Loss} = &- \sum_{i=1}^{n} \left[y_i \log x_i + (1 - y_i) \log(1 - x_i) \right] \\ &+ \frac{1}{2} \sum_{j=1}^{J} \left[\mu_{zj}^2 + \sigma_{zj}^2 - \log(\sigma_{zj}^2) - 1 \right] - \frac{\beta}{n} \sum_{i=1}^{n} (s_i - \bar{s}_i)^2. \end{aligned} \tag{1.7}$$

The loss function has three parts. The first loss function term is the VAE reconstruction loss in the BCE criterion and the second term is the Kullback–Leibler (KL) divergence. These two terms are the same as in Eq. 1.5. The last term is a regularization term which is the MSE loss of the adversarial block. Since the condition information contained in the latent variable z shall be minimized, the MSE loss between s and $\bar{s}$ needs to be maximized. A complete update of the network generally requires alternating updates in two iteration cycles. The first iteration updates the CVAE model based on the loss function in (1.7). The second iteration updates the adversarial block solely based on the MSE loss between s and $\bar{s}$. The total training time using a computer with a GPU board is around 5 minutes.

Figure 1.10 shows the training loss and the validation result as a function of the training iterations. The solid black line represents the training loss and the solid blue line represents the trend for validations for the CVAE model. The plot shows that the training result approaches its optimal point when the epoch number is between 10 to 15. The validation results further decreased beyond the epoch number of 30. Here, we used about 15,000 binary training data, including power splitters semi-optimized by DBS with the target splitting ratios of $5:5$, $6.5:3.5$, and $1:0$. The validation result is an FOM calculated over 20 devices of each type. This result clearly indicates that A-CVAE outperforms CVAE with a large margin.

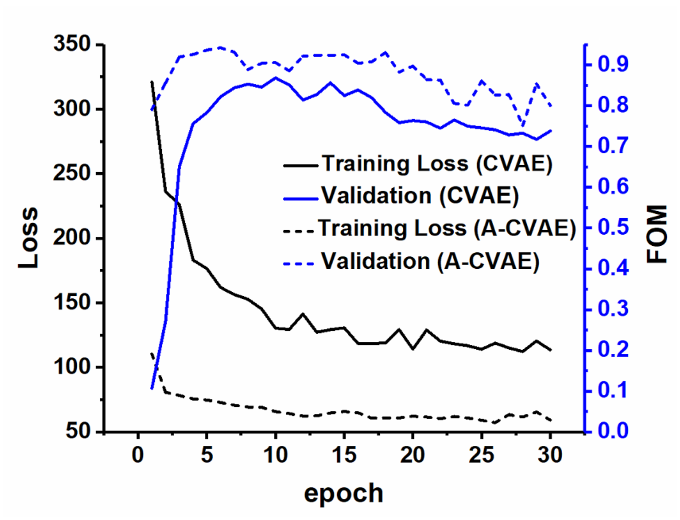


FIGURE 1.10 The general model performance (training loss and validation result) with different epoch number, for both CVAE and A-CVAE model. Regarding the validation, the FOM is calculated after running FDTD simulation for different generated HV patterns. It shows that the model has generally good performance when the epoch number is between 5 and 15.

Figure 1.11(a) shows the latent variable distribution of the CVAE model for groups of devices with four different splitting ratios. The original latent variables are in dimension of 63 and the t-distributed Stochastic Neighbor Embedding (t-SNE) method is used to reduce the dimension to 2 for better visualization. This clearly shows that the four groups are clustered, and their centroids are widely distributed. Figure 1.11(b) shows the similar plot for the A-CVAE model. The figure clearly shows that with adversarial censoring, all the latent variables are distributed similarly, with centroids almost overlapping. They obey the normal distribution $N(0, 1)$, which is expected. It implies that A-CVAE offers more degrees of freedom to generate different potential devices achieving the target performance by sampling normal latent variables concatenated with desired spectra in the conditional decoder.

The initial training data do not contain any data close to the splitting ratios of 6 : 4, 7 : 3, and 8 : 2, and it is difficult to generate very good designs around these splitting ratios. So we employ an active learning as shown in Fig. 1.9 (c). Here, for the second round of training, we uses the new data generated from the first trained model. When we train the second model, the original BCE is replaced with the MSE loss since the training data now contains non-binary data. After training the proposed machine learning model, we test the decoder output of the final A-CVAE model by sampling random latent variable z combined with different splitting conditions to generate nanopatterned power splitter devices according to the generated HVs. Fig. 1.12 shows the comparison

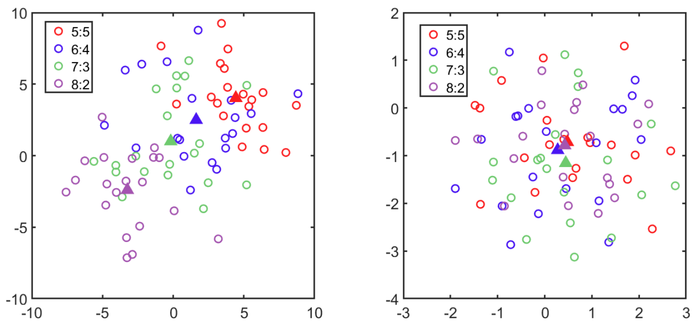


FIGURE 1.11 t-SNE output of the latent variables. The latent variables are obtained from 4 different group of devices with different splitting ratios. We use the t-SNE to reduce the dimension from 63 to 2 for better visualization. The filled triangle markers are the centroid for each device group. a) Latent variables for the CVAE model, which shows clear clustering. b) Latent variables for the A-CVAE model, where the centroid of the four groups overlap with each other.

of the performance among the devices generated by the CVAE model, by the A-CVAE model without active learning, and by the A-CVAE model with active learning, for four different splitting ratios ($5:5$, $6:4$, $7:3$, $8:2$). The FOM is calculated for 20 randomly generated devices from the trained CVAE, A-CVAE models, and A-CVAE models with active learning. This figure shows that the adversarial censoring and active learning generates devices with much better performance across a very broad bandwidth (from 1250μm to 1800μm), compared to the conventional CVAE model. The devices generated by our final A-CVAE model with active learning can fit the target splitting ratio better with excellent total transmission. The average FOMs for the CVAE model, A-CVAE model, and A-CVAE model with active learning are 0.771, 0.888, and 0.9009, respectively.

As the above results show, generative deep learning models can generate a series of improved results based on the statistical characteristics of the training data. In particular, it is that the adversarial censoring and especially with active learning further improves the performance of the model with high stability. Note that part of the good performances of CVAE and A-CVAE come from the fact that variable hole sizes are adopted, which could not be done in the case of DBS.

1.3.6 Nanophotonic Power Splitter Experiment

In order to verify the validity of the simulations our DNN-based design method, power splitters are prototyped using a commercial multip project wafer (MPW) service by Applied Nanotools Inc. The wafer is processed through a standard 220 nm SOI process with SiO_2 cladding. Direct electron beam writing is used for lithography. In this particular design, the size of the square region is $2.6{\times}2.6\mu m^2$

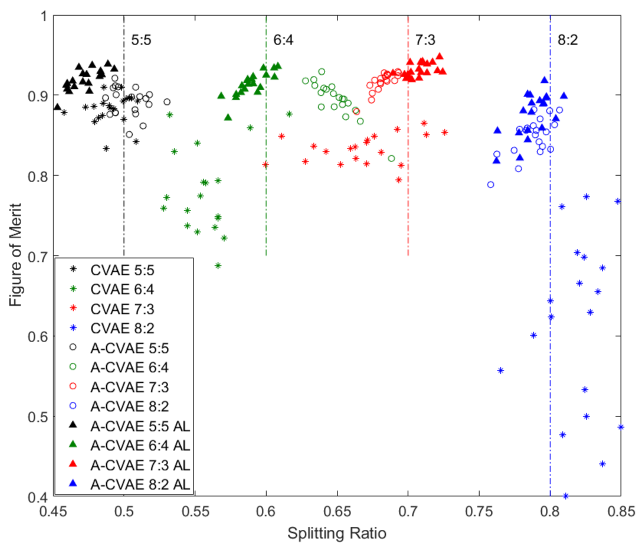


FIGURE 1.12 FOM comparison for different CVAE models: conventional CVAE (star marker), A-CVAE (round marker) and A-CVAE with active learning (triangle marker). Four different splitting ratios are used as a target value to test the model performance (marked with dashed lines). The devices generated by the active learning assisted CVAE model can fit the target splitting ratio better with excellent total transmission. The average FOM for the three models are: 0.771, 0.888, 0.901, respectively.

and the binary hole size is 90 nm in diameter, from a historical reason.

The scanning electron microscope (SEM) image of a 1 : 3 beam splitter is shown in Fig. 1.13. The device is designed by the inverse model as described in Section 1.3.4 using binary variables, and the holes are clearly defined.

The measurement is conducted with grating couplers, using amplified spontaneous emission (ASE) from an Erbium-doped fiber amplifier as a light source. The transmittance is defined by the ratio of the transmitted power of the device under test and a reference device having the same designs of grating couplers. The measurement wavelength range is limited by the grating couplers and the ASE source. Solid lines in Fig. 1.14 show the measured transmittance, and the dashed lines show the transmittance obtained from simulations. The measured values agree well with the simulated values, validating our simulation and optimization results. The cause of the small ripples in the measured transmittance is not clear at this moment.

1.3.7 Comparison with other optimization methods

The adjoint method is widely used in the inverse design of nanophotonic devices [1,3,71]. Given an ideal initial condition for parameters, the optimization

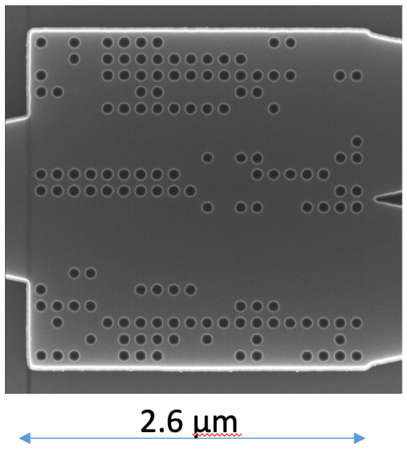


FIGURE 1.13 An SEM image of a prototyped 1 : 3 power splitter.

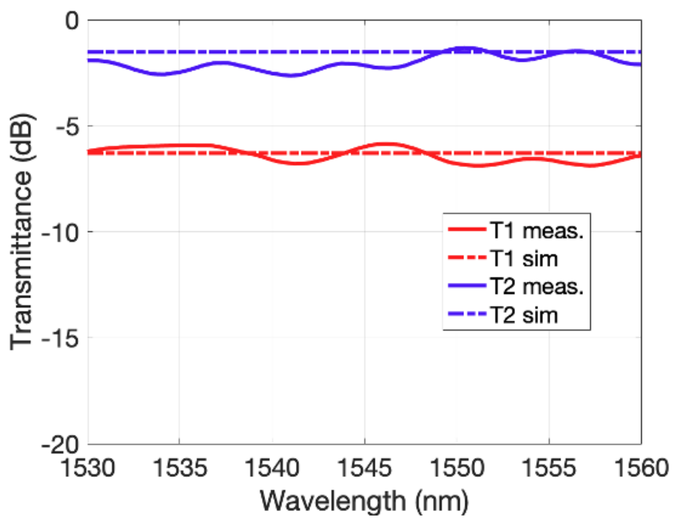


FIGURE 1.14 Solid and dashed lines show the measured and the simulated transmittance, respectively, of a 1 : 3 power splitter. The red and blue lines show transmittance to output ports 1 and 2, respectively.

process can be done in a small number of iterations (tens of EM simulations in some cases). However, the initial condition needs to be carefully chosen in order to obtain the optimal result. A DNN method is very different, in that it is trained from a library (training data) of EM simulation results. The training data may come from previous imperfect optimization results with multiple target condi-

TABLE 1.1 Comparison of simulation results for photonics power splitters using different optimization methods

Split ratio	Insertion loss	Bandwidth	Footprint	Method	Reference	Setup time	Computational time
1 : 1	0.09 dB	100 nm	$2 \times 2\ \mu\text{m}^2$	Adjoint	Lalau-Keraly et. al.[1]		NA
1 : 1	0.32 dB	40 nm	$2.6 \times 2.6\ \mu\text{m}^2$	Adjoint	Wang et. al.[71]		1.2 hr
1 : 1	0.13 dB	80 nm	$1.2 \times 2\ \mu\text{m}^2$	PSO	Zhang et. al.[72]		NA
4 : 6	1 dB (measured)	30 nm	$3.6 \times 3.6\ \mu\text{m}^2$	FSM	Xu et. al.[57]		120 hr
4 : 6	0.65 dB	550 nm	$2.25 \times 2.25\ \mu\text{m}^2$	A-CVAE	Our work [59]	~ 89 hr	~ 5 min
3 : 7	0.51 dB	550 nm	$2.25 \times 2.25\ \mu\text{m}^2$	A-CVAE	Our work [59]	~ 89 hr	~ 5 min

[*] Here the time is mostly for the data collection. The data collection is ~ 20 s × 16000 = 89 hr.
[**] The training time is ~ 5 min and the new design generation time is ~ 5 s.

tions (splitting ratio in the case of Section 1.3). Then the generative model will try to learn/generalize from the library, and generate a series of improved results for a given condition. Once the model is trained, inverse designs for multiple conditions can be generated in almost no time. Further FDTD simulations are not required, and can be used only for verification purposes.

Table 1.1 provides a comparison of power splitters using different optimization methods, including a more conventional y-junction device optimized by classical particle swarm optimization (PSO) [72], and nanophotonic splitters designed by the adjoint method [1,71] and the fast search method (FDM) [57]. Our work has the most broad bandwidth with very small device footprint and low insertion loss. Although, the set-up time is long, it is a one-time process, and the actual computational time is much shorter for each new condition, compared to other optimization methods.

1.4 METASURFACES AND PLASMONICS

Metasurfaces including plasmonics are in principle ultrathin layer of metamaterials which control the wavefront of optical beams [54,73]. There is so much freedom in the design of the nanostructures, and the wavefront can be made to be dependent on the wavelength, polarization, or the combination of both. These are expected to create ultra-thin, high-performance, and multi-functionality lenses. However, there are so many design parameters, so it can be very challenging to design metasurfaces. In this section, we overview the application of DNNs for the design of metasurfaces [38,63–65,74–79]. Just like we discussed three distinct types of DNN models for the inverse design of nanophotonic power splitters, inverse design on metasurfaces has attracted many research groups working on various types of DNNs. One difference is that in the former group as discussed in Section 1.3, the device structure is represented by $n = 400$ (20×20) vectors, while the latter group uses either very small number ($3 - 8$) of parameters to represent nanodisks or nanopillars, or very large number (as much as 64×64 or $45 \times 45 \times 10$) of pixels. Also, in cases where electric field components are used, DNNs typically treat the real and imaginary part of the electric field separately. Nonetheless, the overall concept of connecting the structure parameters to optical responses using DNNs for the inverse design is very similar as we will see in this section.

1.4.1 Deep Learning for Forward Modeling

An *et al.* [75] and Nadell *et al.* [74] used DNNs for forward modeling of the all-dielectric metasurfaces in the terahertz region. They used four cylinders within a unit cell, wherein each cylider radius is denoted as r_1, r_2, r_3, and r_4, and each cylinder height is denoted as h_1, h_2, h_3, and h_4. In addition to these eight geometrical parameters, the ratio of these parameters, like r_1/h_1, r_1/h_2,... and r_4/h_4 are used as the inputs of the DNN. These ratios, motivated by the knowledge of the underlining physics, minimize the need for network parameters. This is analogous to the "tensor layer" [38], which essentially pre-computes products of input variables. The DNN comprises of nine fully connected layers, essentially upsampling the length 25 input vector to a length 165 output vector, followed by three transposed-convolution layers and. a convolution layer, further upsamling to a spectrum of 300 points. The first hidden layer is called “neural tensor network (NTN)”, which takes a tensor of input parameters as well as linear inputs. THis layer can related the two inputs multiplicatively instead of only implicitly through nonlinearity. This layer significantly accelerates the training process. An illustration of the DNN is shown in Fig. 1.15. Using 18,000 pairs of structural parameters and spectra data, this forward network is trained and the typical mean square error (MSE) is less than 3.4×10^{03}. The inverse design problem is performed using this trained forward DNN and a full search algorithm, where 814 million (13^8) forward modeling can be done very quickly (9,400 spectra/s) using a GPU board.

Wiecha and Muskens used a so-called U-Net to model the electric field in the 3D space [76]. U-Net, a variant of encoder-decoder network comprising of CNNs and shortcut connections, is known to be especially efficient in reconstructing spatial images [80]. As shown in Fig. 1.16, the 3D geometric data (0 for no silicon, 1 for silicon) are used as inputs, and six channels of electric fields (real and imaginary parts of the x, y, and z components of the electric field at 3D grid points) are used as outputs. The direct connection between the corresponding CNN layers in the input side and the output side, called "short cut connection", help to accurately reconstruct spatial information from the strongly compressed center layers of the DNN. After the training, given the input geometry, the electric fields are reconstructed, which can be used for calculating the near-field and far-field of scattering (reflection and transmission) and various other properties.

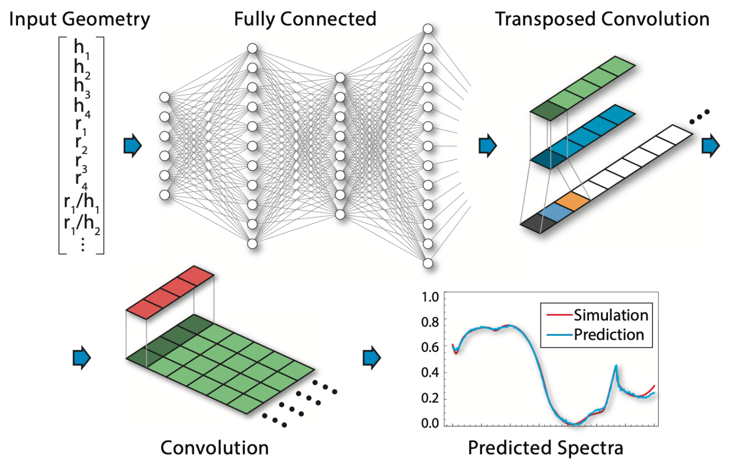


FIGURE 1.15 Forward DNN architecture for predicting the metasurface optical spectra. Nine fully connected layers feed a set of 24 geometric inputs (four radii, four heights, and 16 ratios of these parameters). Data is then smoothed and upsampled in a learnable manner via transpose convolution layers (top row). The final layer (bottom row) is a convolutional layer, which produces a predicted spectra of 300 points, shown as the blue curve, compared to the ground truth (red curve) [74].

1.4.2 Deep Learning for Inverse Modeling

An *et al.* [75] proposed to use an inverse modeling network called "a meta-filter model generator". It has four hidden layers with 500, 500, 500, and 50 neurons, respectively, and takes target spectra of 31 spectral points as inputs, and four device structure parameters, index, gap, thickness, and radius of nanoclylinders, as outputs as shown in Fig. 1.17 [75]. Then it is connected to a pretrained forward modeling network, called "predicting neural network" (PNN), consisting of four hidden layers with 50, 500, 500, and 200 neurons, respectively. The PNN predicts the real and imaginary parts of the actual reflection spectra. For the training of the inverse network, the difference between the target spectra and the output spectra of the PNN are minimized, while the weights of the PNN is fixed. This cascaded network solves the nonconvergence problem resulting from non-bijective solutions. Once the network is trained, inverse designs can be generated in 22 seconds.

Gao *et al.* [77] proposed to use a tandem (forward and inverse) DNN for modeling silicon metasurfaces for color filter applications. The metasurface involves a unit cell of four equally-spaced silicon nanodisks represented by four parameters, i.e., the diameter (D) and height (H) of the nanodisks, the gap (G) between the two nanodisks within a unit cell, and the period (P) of repeating units, as shown in Fig. 1.18. The reflectance spectrum of the metasurface can be represented by the three values x, y, and Y. For the forward design, they

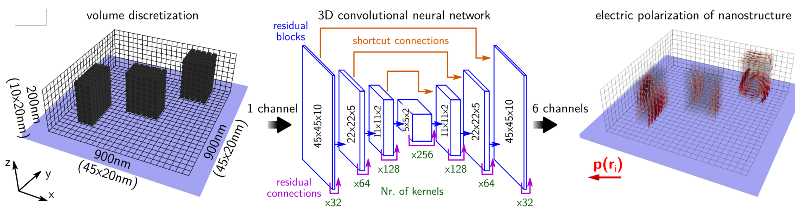


FIGURE 1.16 U-Net for the example of the silicon nanostructure model. The volume discretization of the three-dimensional geometry is fed into the neural network. The 3D convolutional network follows an encoder-decoder architecture and is organized in a sequence of residual blocks. The six output channels of the network contain the real and imaginary parts of the x, y, and z components of the complex electric field inside the nanostructure [76].

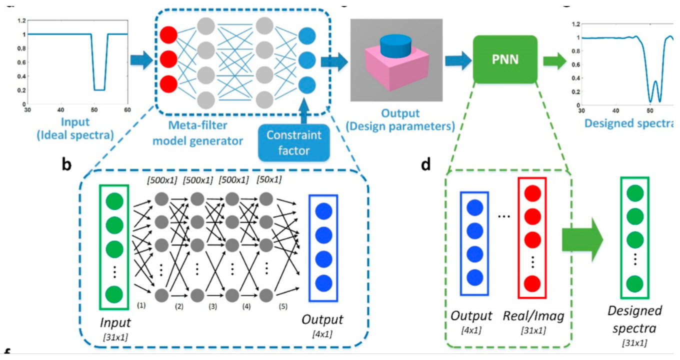


FIGURE 1.17 Architecture of the meta-filter design network and design examples. (a) Target spectra designated as input. (b) The model generator of the meta-filter design network, which is constructed using DNNs. (c) Output of the model generator, which is a combination of design parameters such as permittivity and meta-atom dimensions. (d) These parameters are then fed into the DNN to yield the complex transmission coefficient. (e) The refined amplitude response of the generated design as derived from the complex transmission coefficient [75].

use a DNN with four hidden layers with 320 neurons in each layer, with the four structural parameters as inputs, and the three color values as outputs. For the inverse design, they use three color parameters as inputs and four structural parameters as outputs. The optimal structure consists of four hidden layers each with 300 neurons. If the inverse network is used alone, the validation error is fairly large. In a tandem DNN as shown in Fig. 1.19, a pretrained network is added at the output of the inverse network. The inverse network is trained such that the differences between the input values x, y, and Y, and the output values of the pretaiend forward network x', y', and Y' are minimized. In the tandem DNN case, the validation error is significantly lower than the inverse network is used alone, indicating that the non-bijective issue between color and structure is overcome.

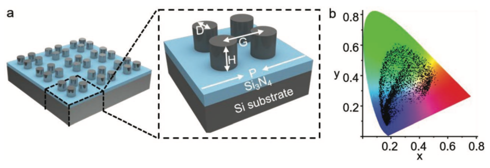


FIGURE 1.18 Schematic illustration of the silicon nanostructure and the generated colors. a) The studied geometry parameters include the diameter (D) and height (H) of nanodisks, the gap (G) between nanodisks in a unit and the period (P) of repeating units. b) The coverage of training data generated colors on the CIE 1931 chromatic diagram [77].

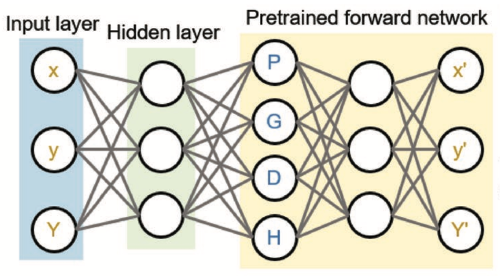


FIGURE 1.19 Tandem DNNs architecture with an input layer of color values x, y, and Y, hidden layers, and a connected pretrained forward modeling network. The tandem network is trained such that the differences between the input values x, y, and Y, and the output values of the pretaiend forward network x', y', and Y' are minimized [77].

1.4.3 Deep Learning for Generative Modeling

There are two mainstreams of generative modeling, i.e., CVAE and GAN. Generative adversarial networks (GANs) are algorithmic architectures that use two neural networks, pitting one against the other (thus the "adversarial") in order to generate new, synthetic instances of data that can pass for real data, and are used widely in image, video, and voice generation. For photonic device designs, both CVAE and GAN are similar in that they generate new device topology candidates based on random numbers and input conditions.

Ma *et al.* proposed a variant of CVAE for the inverse design of plasmonic metamaterials [64]. The design data consists of 64×64 binary image for the metal patterns, while the optical response consists of three polarization dependent reflection spectra each with 31 frequencty points. Fig. 1.20 shows the CVAE model for the inverse design of plasmonic metamaterials, comprising of three submodels, the recognition model, the prediction model, and the generation model. The recognition model encodes the metamaterial pattern with its optical response into a low-dimensional latent space. The prediction model outputs a deterministic prediction of the optical response given the metamaterial design. The generation model accepts the optical response and the sampled latent variable

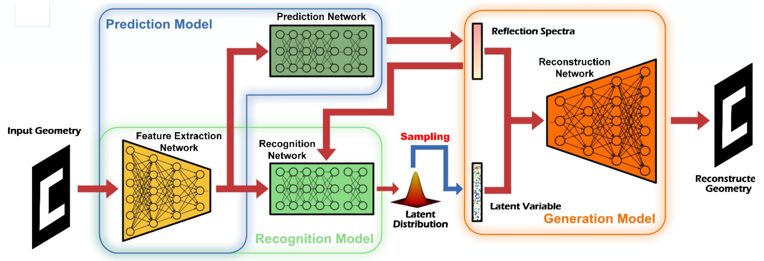


FIGURE 1.20 CVAE model for inverse design of metamaterials. Three submodels, the recognition model, the prediction model, and the generation model, constitute the complete architecture, which is implemented by four neural networks with deliberately designed structures for different purposes [64].

to produce feasible metamaterial designs according to specific requirements. The trained network can predict optical responses when a plasmonic metamaterial structure is given. Also, given target optical responses, a series of metamaterial designs can be generated with sampling of latent variables.

Liu *et al.* propsed using GAN for the inverse design of metasurfaces [63]. As shown in Fig.1.21, the proposed network consists of three networks, the generator (G), the simulator (S), and the critic (D, also called a discriminator). The generator (an inverse network) accepts the spectra T (four transmission polarization components comprising of 32 elements) and noise z (50 element vector) and produces possible patterns. The simulator is a pretrained forward network that approximates the transmittance spectrum $\hat{T}$ for a given pattern (64×64 binary image) at its input, and the critic evaluates the distance of the distributions between the geometric data and the patterns from the generator. While training the generator, the produced patterns vary according to the feedback obtained from S and D. Valid patterns are documented during the training process, and are smoothed to qualify as candidate structures.

There are also propsals of hybrid approaches to combine the generative model with other optimization technologies. Liu *et al.*, the same group as above, first trained a variational autoencoders (VAE) network as shown in Fig. 1.22 (a). The geometric data are 64×64 binary vectors, and the spectral data are T_{xx}, T_{xy}, T_{yx}, and T_{yy}, each with 32 spectral points between 170 THz and 600 THz. Random variables μ, σ, and υ have the same size of 10. The trained decoder works as an inverse network, with randomly sampled vectors as inputs, and randomly generated nanostructures as outputs (Fig. 1.23). It takes an input vector of length 10, and outputs 64×64 images, with four transposed convolutional layers. Fig. 1.23 shows the flowchart of framework combining both VAE and an evolutional strategy (ES) [78]. Here, the generator (G) is the trained decoder, and the Simulation (S) can be carried out by either a forward DNN, or a real EM simulation.

Wen *et al.* used progressive growth of GANs (PGGANs) to learn spatially

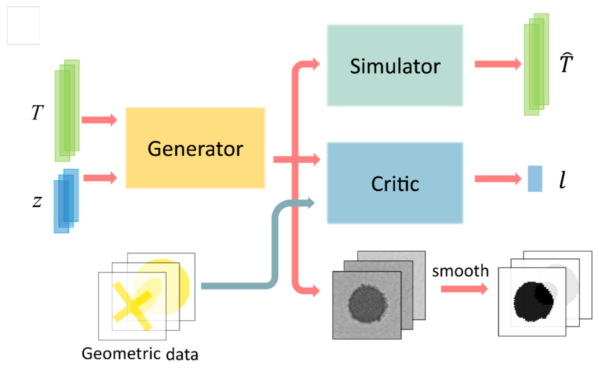


FIGURE 1.21 Architecture of the proposed GAN. Three networks, the generator (G), the simulator (S), and the critic (D) constitute the complete architecture. The generator accepts the spectra T and noise z and produces possible patterns [63].

fine features from high-resolution training data. PGGANs are often used in the computer vision community and they support improved training stability and the ability to capture spatially fine features from a high-resolution training set. Here, the generative model, the discriminative model, and network resolution progressively grow over the course of training.

Fig. 1.24 (a) shows the overview of a PGGAN training protocol. The initial training set consists of high-performing devices sampling the coarse grid of the device parameter space as shown in the left box. The training process includes multiple training cycles, each of which PGGAN is trained from scratch, evaluating/validating the generated devices once the training is complete, and augmenting the training set with the best generated devices (active learning). More specifically, in each training cycle, the training starts with a training set comprising of 8×16 pixels, downsampled from the 64×128 pixels. The training data sets having 16×32, 32×64, and 64×128 are used as the training progresses, and layer structures of the generator and the discriminator grow accordingly. The final trained PGGAN can output device layouts that continuously span the full device parameter space. The PGGAN is shown to significantly outperform the conventional GAN.

1.5 OTHER TYPES OF OPTICAL DEVICES

In this section, we review activities of using DNN for designing optical devices other than nanophotonic power splitters and metasurfaces. The examples below are forward modeling methods and generative modeling methods for waveguide type devices, dielectric films, and photonic crystals.

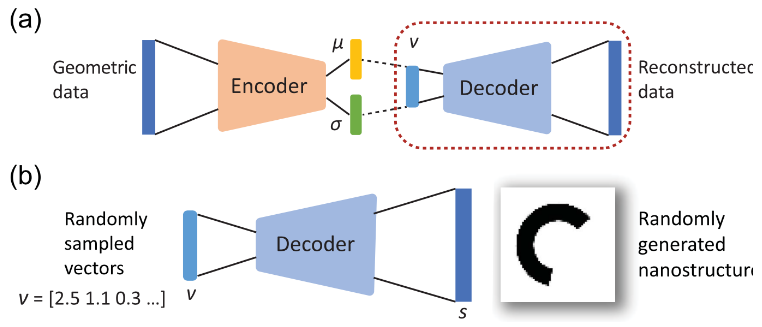


FIGURE 1.22 Basic architecture of a VAE implemented in the framework. (a) The illustration of the vanilla VAE. The encoder accepts the geometric data and produces two parametric vectors μ and σ . Random vectors υ are samples from the normal distribution with the mean and the standard deviation defined by μ and σ. The decoder then reconstructs vectors υ to images of photonic structures. The VAE encodes the geometric data into a compact latent space where the optimization algorithms can be applied efficiently. (b) After the training, the decoder encircled in (a) can be separated and treated as a generator of geometric data. It transforms randomly sampled vectors v to their correspondent structures [78].

1.5.1 Deep Learning for Forward Modeling

Hammond and Camacho used DNNs for forward and inverse modeling of strip waveguides and Bragg gratings [50]. In the case of silicon photonic sidewall-corrugated linearly-chirped Bragg gratings, the structural parameters are the width of the first part of the waveguide w_0, thickness (t), waveguide corrugation with difference $(w = w_1 - w_0)$, the length of the first grating period (a_0), and the last grating period (a_1). The output properties are the reflection spectrum and group delay response. They generated $104,000$ training grating structures with 250 wavelength points, using a layered dielectric media transfer matrix method (LDMTMM). The DNN consists of 10hidden layers and 128 neurons for each layer. Inverse design is performed using a gradient-based methods by directly evaluating the Jabocbian and Hessian tensors from the DNN without any extra sampling or discretization.

In order to accelerate design antireflection coating, Hedge paired DNN with evolutionary algorithms [51]. The structure is an air-clad 16-layered thin-film made of alternating layers of silica and titania on a semi-infinite substrate of refractive index 1.52. The input parameters are the thicknesses of each layer. The output parameters are refletion spectra with 64 points betwen 400 nm and 800 nm. The DNN comprises of fully connected 3 hidden layers each with 128 neurons. Once the DNN is trained with $>$ 50,000 training data, differential evolution (DE) algorithm, which is one of the evolutionary strategy variant, uses the output of the DNN, instead of the actual calculated reflection spectra by “oracle”, which is an actual optical simulator based on the transfer matrix method (TMM) [82]. This DNN-assisted DE significantly accelerates the optimization

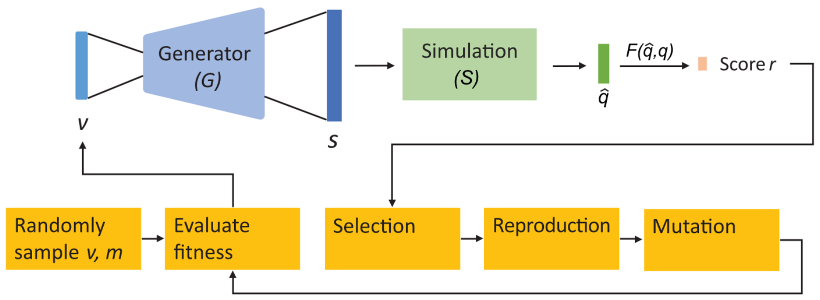


FIGURE 1.23 Flowchart of the VAE-ES framework. The generator (G) is utilized to reconstruct the structure of each v. Simulation process (S) is then followed to produce the simulation results. The simulation can be carried out by either a neural network approximation or a real physical simulation such as FEM. The yellow boxes show the functionalities of a traditional ES algorithm [78].

process.

Asano and Noda used a DNN as a forward regression model to predict the Q factor, to optimize the microcavity structure [42,83]. Fig. 1.25 shows the schematic of the photonic crystal structure with a three-missing-air-holes (L3) cavity as the base structure. The displacement of the 50 air holes are the structural parameters to be optimized to maximize Q. The network structure is shown in Fig. 1.26, where the input nodes are two-channel 2D tables with each channel corresponds to the x and y components of the displacement vector with a length of 50. The CNN layer consists of 50 filters with a size of 3 holes $\times$ 5 rows $\times$ 2 channels. After training a CNN using 1,000 initial training data calculated from 3D FDTD, a gradient method is used to obtain semi-optimal nanocavity structures. 70 new candidates are selected and 3D FDTD verifies the Q factor. The new data are added to the training data to update the CNN (active learning), and repeat the process. A record Q factor of 1.1×10^7 is obtained. They found that exploring structure not only near the present highest Q factor, but also those distant from it, accelerate the optimization process. Effects of fabrication errors on the Q factor are also analyzed using the CNN model.

1.5.2 Deep Learning for Generative Modeling

Christensen *et al.* used DNNs for predicting and generating photonic crystals [84]. They first used a CNN-based encoder and a deconder network to predict the band gap of photonic crystals as shown in Fig. 1.28. The encoder consists of three CNN layers and two fully connected layers, mapping 32×32 input space into a linear 64-dimensional feature space (latent variable). The decoder was implemented with six feed-forward networks, each consisting of five fully-connected lyaers that were separately optimized for each band. The relative error for the predicted bandgap is typically very small ($< 2\%$). As for data

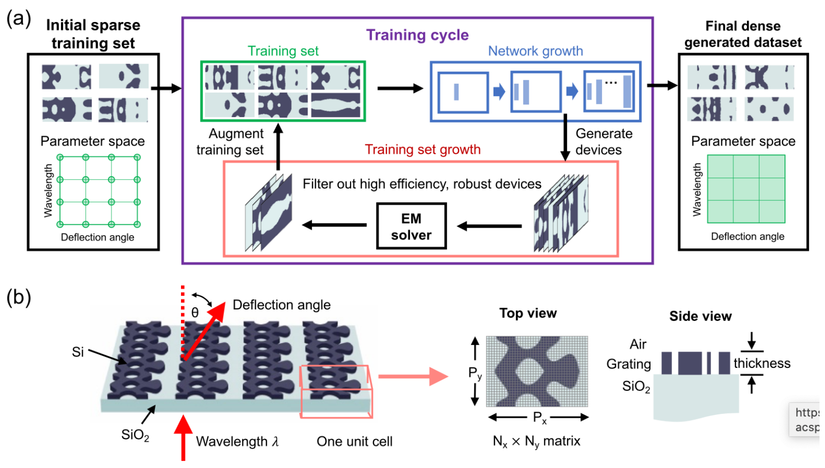
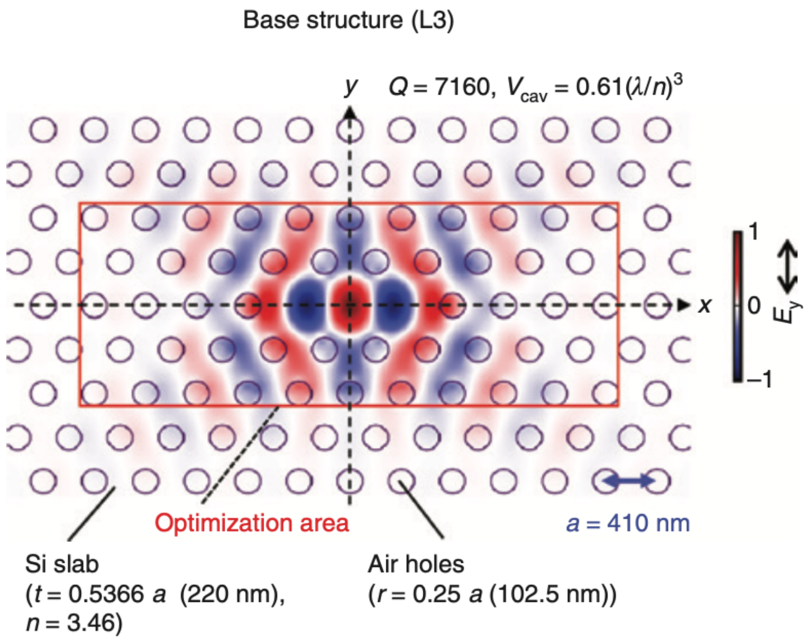


FIGURE 1.24 Overview of progressively growing GAN (PGGAN) network architectures and training sets. (a) Schematic of the training protocol. The initial training set consists of high-performing devices sampling the coarse grid of the device parameter space. The training process consists of multiple training cycles, each of which involves training the PGGAN from scratch, evaluating/validating the generated devices, and augmenting the training set with the best generated devices. The final trained PGGAN can generate device structures that span the full device parameter space. (b) Freeform silicon metagratings deflect normally incident light to the angle θ [81].

generation, they adapted a standard off-the-shelf implementation of GAN [85]. The input data are 64 unit cell permittivity profile, in which TM band gap is greater than 5%. The discriminator network outputs real when the bandgap is greater than 5% as shown in Fig. 1.29. With enough training, the GAN generates new data satisfying the bandgap greater than 5%. They compared a conventional GAN, a least squares GAN (LSGAN), and Deep Regret Analytic GAN (DRAGAN), and the fidelity (fraction of the unit cells hosting a band gap greater than 5% is slightly higher with the LSGAN than the conventional GAN, while it was lowest with the DRAGAN in this particular problem.

1.6 DISCUSSION

DNNs can be used to take device structure data (shape, depth, and permittivity) to predict the optical response of the nanostructure in the forward modeling framework. In this case DNNs can be used as a viable counterpart for fast approximation of the optical response, in comparison to the use of computationally heavy FDTD or FEM simulations. In contrast, inverse-modeling DNNs take an optical response to provide the user with an approximate solution of nanostructure in the inverse modeling framework, which does not rely on external meta-heuristic optimizers unlike forward modeling. The generative model, on the other hand, implicitly integrates forward and inverse models. Once the



Base structure (L3)
y
$Q = 7160$, $V_{cav} = 0.61(\lambda/n)^3$
x
1
0
–1
E_y
Optimization area
$a = 410$ nm
Si slab
($t = 0.5366\ a$ (220 nm),
$n = 3.46$)
Air holes
($r = 0.25\ a$ (102.5 nm))

FIGURE 1.25 A three-missing-air-holes (L3) cavity is used as the base structure for structural optimization. The lattice constant a is 410 nm. The distribution of the y component of the electric field (E_y) of the fundamental resonant mode is plotted in color. The theoretical Q factor of the base structure determined by FDTD is 7160. The displacements of the 50 air holes inside the red square are the structural parameters that are used to optimize the cavity design with respect to Q [83].

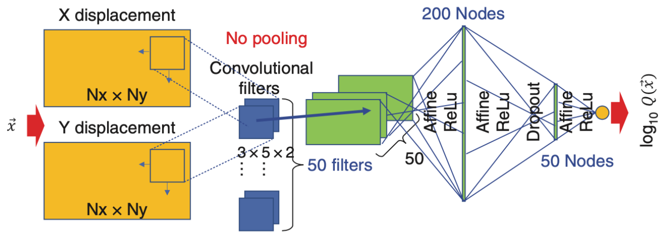


FIGURE 1.26 Configuration of the neural network prepared to learn the relationship between displacements of air holes and Q factors [83].

network is trained, the generation of new designs takes practically little time. In the case of CVAE, the use of adversarial censoring further improves the design capability. Overall, DNNs have the capability of learning the training data with a high generalizability.

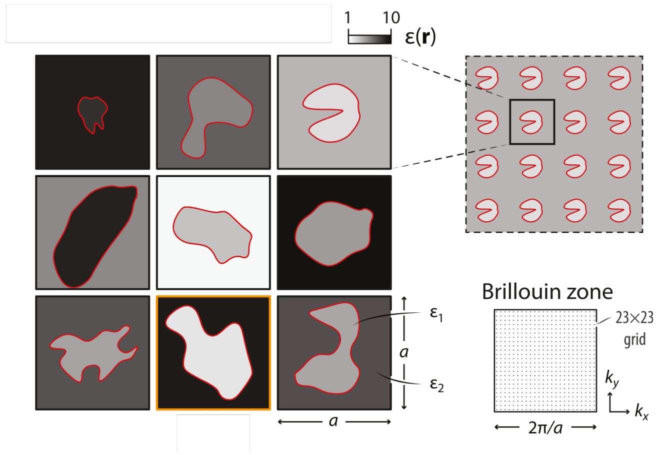


FIGURE 1.27 Several representative unit cells and the Brillouine grid-sampling used in the calculation of band structures. [84].

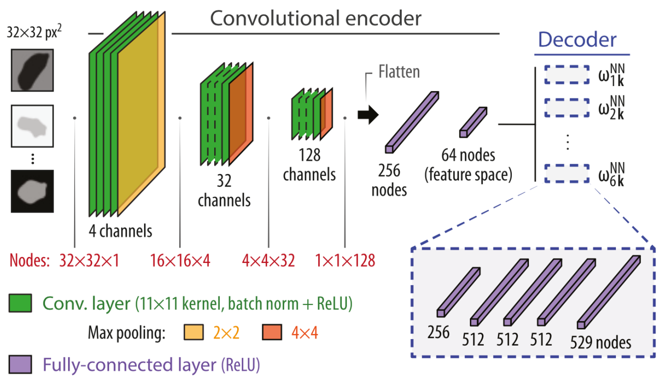


FIGURE 1.28 Predictive network architecture showing the convolutional encoder and fully-connected decoder [84].

Although the DNN initially needs a sufficient amount of data for the training purpose, it is possible to process several heuristic optimization metrics in parallel on a computing cluster to speed up generating the training data. We can design the nanostructured geometry in a fraction of a second once the network is trained to represent the topology as optical response and vice versa. There is an overhead of generating large amount of training data upfront, however, the data can be accumulated from multiple optimization runs, such as DBS, which did not necessarily generate optimal results, or targeting different splitting ratios or other optical performances.

We have seen that there are multiple ways to generate new device structures,

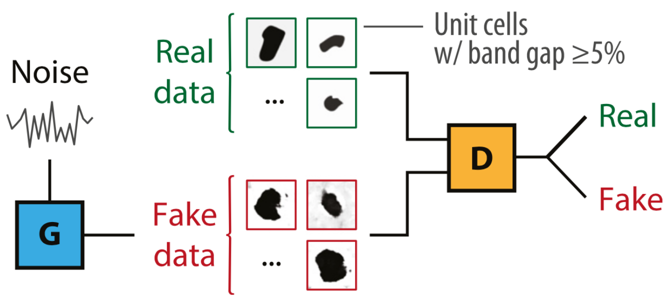


FIGURE 1.29 GAN consists of a generative (G) and a discriminative (D) network, and new synthetic examples (fake) of 2D unit cells with a TM band gap can be generated from a genuine data set (real) [84].

and also some of them use hybrid optimization algorithms. At this moment, there is no comprehensive study to compare all the variants of networks architectures using a common data set. Therefore, we do not know if one network architecture works best for most of the device categories, or different device categories call for different network architectures. This will be the task for the future.

1.7 CONCLUSION

We gave an overview of how photonic devices can be designed using deep learning, and how they are different from conventional optimization processes. Especially, to design complicated nanophotonic devices and metasurfaces with hundreds or thousands of parameters, a sophisticated design algorithm is necessary, and deep learning offers a promising solution. Most of the problems are characterized as structural parameters as inputs and transmission/reflection spectral characteristics as outputs.

We demonstrated three different types of DNN methodologies, i.e., forward modeling, inverse modeling, and generative modeling, to design nanophotonic power splitters.

The forward modeling uses a DNN to predict the spectral performances given the device topology. The DNN is integrated with an optimization method, to reduce the requirement of computationally intensive FDTD simulations. As more data are accumulated for online training, the prediction accuracy further improves. The inverse modeling uses a DNN to directly generate the device topology given a target performance. By supplying a series of modified SPEC performance, a series of good device structure candidates is generated. This can avoid the use of external optimizer methods. For the generative model, we reviewed a CVAE with adversarial censoring. Once trained, the CVAE can generate a series of improved device structures given spectral performance data as a condition along with random variables sampled from the normal distribution.

We confirmed that the adversarial censoring significantly improves the design capability.

We also gave an overview of parallel activities of these DNN modeling approaches for metasurfaces and plasmonics and other photonic devices.

Among all the efforts and activities for the design of photonic devices, we can find some common themes and techniques. 1) The current general trend is moving towards generative modeling. 2) When the structure is represented in periodic parameters such as grids, 2D/3D CNN is a very powerful tool. 3) Use large number of output values (such as fine spectrums) instead of a few numbers, to minimize non-bijectivity issue. 4) If complex numbers are used, such as electric fields, real and imaginary part are handled separately. 5) When small number of input structural parameters are used, if the physics allows, using multiplications or ratios of these parameters in addition to linear terms, can accelerate training or improve performance. 6) Active learning usually helps to improve the DNN for iterative design/optimization processes.

The area of deep learning-assisted photonic device design is quickly expanding, and these modeling methods will play important roles in the advancement of photonic device design activities. The future task will involve determing if there is a single network architecture or framework which work for almost all the device types, or each device category has one specific framework which work best for it.

[1] C. M. Lalau-Keraly, S. Bhargava, O. D. Miller, E. Yablonovitch, Adjoint shape optimization applied to electromagnetic design, Optics express 21 (18) (2013) 21693–21701.

[2] A. Silva, F. Monticone, G. Castaldi, V. Galdi, A. Alù, N. Engheta, Performing mathematical operations with metamaterials, Science 343 (6167) (2014) 160–163.

[3] A. Y. Piggott, J. Lu, K. G. Lagoudakis, J. Petykiewicz, T. M. Babinec, J. Vučković, Inverse design and demonstration of a compact and broadband on-chip wavelength demultiplexer, Nature Photonics 9 (6) (2015) 374–377.

[4] B. Shen, P. Wang, R. Polson, R. Menon, An integrated-nanophotonics polarization beamsplitter with 2.4×2.4 μm 2 footprint, Nature Photonics 9 (6) (2015) 378–382.

[5] M. Teng, K. Kojima, T. Koike-Akino, B. Wang, C. Lin, K. Parsons, Broadband SOI mode order converter based on topology optimization, in: 2018 Optical Fiber Communications Conference and Exposition (OFC), 2018, pp. 1–3.

[6] A. Motayed, G. Aluri, A. V. Davydov, M. V. Rao, V. P. Oleshko, R. Bajpai, M. E. Zaghloul, B. Thomson, B. Wen, T. Xie, et al., Highly selective nanostructure sensors and methods of detecting target analytes, uS Patent 9,983,183 (May 29 2018).

[7] Z. Chu, Y. Liu, J. Sheng, L. Wang, J. Du, K. Xu, On-chip optical attenuators designed by artifical neural networks, in: 2018 Asia Communications and Photonics Conference (ACP), IEEE, 2018, pp. 1–3.

[8] Z. Liu, X. Liu, Z. Xiao, C. Lu, H.-Q. Wang, Y. Wu, X. Hu, Y.-C. Liu, H. Zhang, X. Zhang, Integrated nanophotonic wavelength router based on an intelligent algorithm, Optica 6 (10) (2019) 1367–1373.

[9] M. Teng, A. Honardoost, Y. Alahmadi, S. S. Polkoo, K. Kojima, H. Wen, C. K. Renshaw, P. LiKamWa, G. Li, S. Fathpour, et al., Miniaturized silicon photonics devices for integrated optical signal processors, Journal of Lightwave Technology (2019).

[10] X. Ni, Z. J. Wong, M. Mrejen, Y. Wang, X. Zhang, An ultrathin invisibility skin cloak for visible light, Science 349 (6254) (2015) 1310–1314.

[11] A. Alù, N. Engheta, Achieving transparency with plasmonic and metamaterial coatings, Physical Review E 72 (1) (2005) 016623.

[12] F. Monticone, N. M. Estakhri, A. Alù, Full control of nanoscale optical transmission with a composite metascreen, Phys. Rev. Lett. 110 (2013) 203903. doi:10.1103/PhysRevLett.110.203903. URL `https://link.aps.org/doi/10.1103/PhysRevLett.110.203903`

[13] E. Arbabi, A. Arbabi, S. M. Kamali, Y. Horie, A. Faraon, Multiwavelength polarization-insensitive lenses based on dielectric metasurfaces with meta-molecules, Optica 3 (6) (2016) 628–633.

[14] A. K. Azad, W. J. Kort-Kamp, M. Sykora, N. R. Weisse-Bernstein, T. S. Luk, A. J. Taylor, D. A. Dalvit, H.-T. Chen, Metasurface broadband solar absorber, Scientific Reports 6 (20347) (2016).

[15] M. Khorasaninejad, W. T. Chen, R. C. Devlin, J. Oh, A. Y. Zhu, F. Capasso, Metalenses at visible wavelengths: Diffraction-limited focusing and subwavelength resolution imaging, Science 352 (6290) (2016) 1190–1194.

[16] A. Krasnok, M. Tymchenko, A. Alù, Nonlinear metasurfaces: A paradigm shift in nonlinear optics, Materials Today 21 (1) (2018) 8 – 21. doi:https://doi.org/10.1016/j.mattod.2017.06.007. URL `http://www.sciencedirect.com/science/article/pii/S136970211730233X`

[17] R. Pestourie, C. Pérez-Arancibia, Z. Lin, W. Shin, F. Capasso, S. G. Johnson, Inverse design of large-area metasurfaces, Optics express 26 (26) (2018) 33732–33747.

[18] L. H. Frandsen, O. Sigmund, Inverse design engineering of all-silicon polarization beam splitters, in: Photonic and Phononic Properties of Engineered Nanostructures VI, Vol. 9756, International Society for Optics and Photonics, 2016, p. 97560Y.

[19] A. Y. Piggott, J. Petykiewicz, L. Su, J. Vučković, Fabrication-constrained nanophotonic inverse design, Scientific Reports 7 (1) (2017) 1786.

[20] K. Kojima, B. Wang, U. Kamilov, T. Koike-Akino, K. Parsons, Acceleration of FDTD-based inverse design using a neural network approach, in: Integrated Photonics Research, Silicon and Nanophotonics, Optical Society of America, 2017, pp. ITu1A–4.

[21] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[22] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, Communications of the ACM 60 (6) (2017) 84–90. doi:10.1145/3065386. URL `http://doi.acm.org/10.1145/3065386`

[23] J. Ghaboussi, J. Garrett Jr, X. Wu, Knowledge-based modeling of material behavior with neural networks, Journal of Engineering Mechanics 117 (1) (1991) 132–153.

[24] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, A. Aspuru-Guzik, Automatic chemical design using a data-driven continuous representation of molecules, ACS central science 4 (2) (2018) 268–276.

[25] B. Sanchez-Lengeling, A. Aspuru-Guzik, Inverse molecular design using machine learning: Generative models for matter engineering, Science 361 (6400) (2018) 360–365.

[26] Z. Shi, E. Tsymbalov, M. Dao, S. Suresh, A. Shapeev, J. Li, Deep elastic strain engineering of bandgap through machine learning, Proceedings of the National Academy of Sciences 116 (10) (2019) 4117–4122.

[27] P. Baldi, P. Sadowski, D. Whiteson, Searching for exotic particles in high-energy physics with deep learning, Nature Communications 5 (4308) (2014).

[28] M. Yasui, M. Hiroshima, J. Kozuka, Y. Sako, M. Ueda, Automated single-molecule imaging in living cells, Nature communications 9 (1) (2018) 3061.

[29] Y. Jun, T. Eo, T. Kim, H. Shin, D. Hwang, S. H. Bae, Y. W. Park, H.-J. Lee, B. W. Choi, S. S. Ahn, Deep-learned 3D black-blood imaging using automatic labelling technique and 3D convolutional neural networks for detecting metastatic brain tumors, Scientific reports 8 (1) (2018).

[30] A. Radovic, M. Williams, D. Rousseau, M. Kagan, D. Bonacorsi, A. Himmel, A. Aurisano, K. Terao, T. Wongjirad, Machine learning at the energy and intensity frontiers of particle physics, Nature 560 (7716) (2018).

[31] F. N. Khan, Y. Zhou, A. P. T. Lau, C. Lu, Modulation format identification in heterogeneous fiber-optic networks using artificial neural networks, Optics express 20 (11) (2012) 12422–12431.

[32] T. Koike-Akino, Perspective of statistical learning for nonlinear equalization in coherent optical communications, in: Advanced Photonics for Communications, Optical Society of America, 2014, p. ST2D.2. doi:10.1364/SPPCOM.2014.ST2D.2.
URL `http://www.osapublishing.org/abstract.cfm?URI=SPPCom-2014-ST2D.2`

[33] D. Zibar, M. Piels, R. Jones, C. G. Schäeffer, Machine learning techniques in optical communication, Journal of Lightwave Technology 34 (6) (2015) 1442–1452.

[34] D. Rafique, L. Velasco, Machine learning for network automation: Overview, architecture, and applications (invited tutorial), J. Opt. Commun. Netw. 10 (10) (2018) D126–D143. doi:10.1364/JOCN.10.00D126.
URL `http://jocn.osa.org/abstract.cfm?URI=jocn-10-10-D126`

[35] B. Karanov, M. Chagnon, F. Thouin, T. A. Eriksson, H. Bülow, D. Lavery, P. Bayvel, L. Schmalen, End-to-end deep learning of optical fiber communications, Journal of Lightwave Technology 36 (20) (2018) 4843–4855.

[36] T. Koike-Akino, Y. Wang, D. S. Millar, K. Kojima, K. Parsons, Neural turbo equalization to mitigate fiber nonlinearity, in: European Conference on Optical Communication (ECOC), 2019, p. Tu.1.B.1.

[37] D. Liu, Y. Tan, E. Khoram, Z. Yu, Training deep neural networks for the inverse design of nanophotonic structures, ACS Photonics 5 (4) (2018) 1365–1369.

[38] W. Ma, F. Cheng, Y. Liu, Deep-learning enabled on-demand design of chiral metamaterials, ACS Nano 12 (6) (2018) 6326–6334.

[39] I. Malkiel, M. Mrejen, A. Nagler, U. Arieli, L. Wolf, H. Suchowski, Deep learning for the design of nano-photonic structures, in: 2018 IEEE International Conference on Computational Photography (ICCP), 2018, pp. 1–14. doi:10.1109/ICCPHOT.2018.8368462.

[40] J. Peurifoy, Y. Shen, L. Jing, Y. Yang, F. Cano-Renteria, B. G. DeLacy, J. D. Joannopoulos, M. Tegmark, M. Soljačić, Nanophotonic particle simulation and inverse design using artificial neural networks, Science Advances 4 (6) (2018). arXiv:http://advances.sciencemag.org/content/4/6/eaar4206.full.pdf, doi:10.1126/sciadv.aar4206.
URL `http://advances.sciencemag.org/content/4/6/eaar4206`

[41] Y. Sun, Z. Xia, U. S. Kamilov, Efficient and accurate inversion of multiple scattering with deep learning, Optics Express 26 (11) (2018) 14678–14688.

[42] T. Asano, S. Noda, Optimization of photonic crystal nanocavities based on deep learning, Optics express 26 (25) (2018) 32704–32717.

[43] J. Ohta, K. Kojima, N. Y, T. Shuichi, K. Kazuo, Optical neurochip based on a three-layered feed-forward model, Optics letters 15 (23) (1990) 1362–1364.

[44] A. N. Tait, T. F. Lima, E. Zhou, A. X. Wu, M. A. Nahmias, B. J. Shastri, P. R. Prucnal, Neuromorphic photonic networks using silicon photonic weight banks, Scientific Reports 7 (1) (2017).

[45] A. Mehrabian, Y. Al-Kabani, V. J. Sorger, T. El-Ghazawi, PCNNA: A photonic convolutional neural network accelerator, arXiv preprint arXiv:1807.08792 (2018).

[46] X. Lin, Y. Rivenson, N. T. Yardimci, M. Veli, Y. Luo, M. Jarrahi, A. Ozcan, All-optical machine learning using diffractive deep neural networks, Science (2018). arXiv:http://science.sciencemag.org/content/early/2018/07/25/science.aat8084.full.pdf, doi:10.1126/science.aat8084.
URL `http://science.sciencemag.org/content/early/2018/07/25/science.aat8084`

[47] J. Chiles, S. M. Buckley, S. W. Nam, R. P. Mirin, J. M. Shainline, Design, fabrication, and metrology of 10×100 multi-planar integrated photonic routing manifolds for neural networks, APL Photonics 3 (10) (2018).

[48] T. W. Hughes, M. Minkov, Y. Shi, S. Fan, Training of photonic neural networks through in situ backpropagation and gradient measurement, Optica 5 (7) (2018) 864–871.

[49] W. Xiong, B. Redding, S. Gertler, Y. Bromberg, H. D. Tagare, H. Cao, Deep learning of ultrafast pulses with a multimode fiber, APL Photonics 5 (9) (2020) 096106.

[50] A. M. Hammond, R. M. Camacho, Designing integrated photonic devices using artificial neural networks, Optics express 27 (21) (2019) 29620–29638.

[51] R. S. Hegde, Photonics inverse design: Pairing deep neural networks with evolutionary algorithms, IEEE Journal of Selected Topics in Quantum Electronics 26 (1) (2019) 1–8.

[52] S. Banerji, A. Majumder, A. Hamrick, R. Menon, B. Sensale-Rodriguez, Machine learning enables design of on-chip integrated silicon T-junctions with footprint of 1.2μm $\times$ 1.2μm, arXiv preprint arXiv:2004.11134 (2020).

[53] Z. Kang, X. Zhang, J. Yuan, X. Sang, Q. Wu, G. Farrell, C. Yu, Resolution-enhanced all-optical analog-to-digital converter employing cascade optical quantization operation, Optics Express 22 (18) (2014) 21441–21453.

[54] N. Yu, F. Capasso, Optical metasurfaces and prospect of their applications including fiber optics, J. Lightwave Technol. 33 (12) (2015) 2344–2358.
URL `http://jlt.osa.org/abstract.cfm?URI=jlt-33-12-2344`

[55] Y. Lu, J. Lu, A universal approximation theorem of deep neural networks for expressing probability distributions, Advances in Neural Information Processing Systems 33 (2020).

[56] Y. Tian, J. Qiu, M. Yu, Z. Huang, Y. Qiao, Z. Dong, J. Wu, Broadband 1×3 couplers with variable splitting ratio using cascaded step-size MMI, IEEE Photonics Journal 10 (3) (2018) 1–8.

[57] K. Xu, L. Liu, X. Wen, W. Sun, N. Zhang, N. Yi, S. Sun, S. Xiao, Q. Song, Integrated photonic power divider with arbitrary power ratios, Optics Letters 42 (4) (2017) 855–858.

[58] L. Lu, M. Zhang, F. Zhou, D. Liu, An ultra-compact colorless 50: 50 coupler based on PhC-like metamaterial structure, in: Optical Fiber Communications Conference and Exhibition (OFC), 2016, IEEE, 2016, pp. 1–3.

[59] Y. Tang, K. Kojima, T. Koike-Akino, Y. Wang, P. Wu, Y. Xie, M. H. Tahersima, D. K. Jha, K. Parsons, M. Qi, Generative deep learning model for inverse design of integrated nanophotonic devices, Laser & Photonics Reviews (2020) 2000287.

[60] M. H. Tahersima, K. Kojima, T. Koike-Akino, D. Jha, B. Wang, C. Lin, K. Parsons, Deep neural network inverse design of integrated photonic power splitters, Scientific Reports 9 (1) (2019) 1368.

[61] M. H. Tahersima, K. Kojima, T. Koike-Akino, D. Jha, B. Wang, C. Lin, K. Parsons, Nanostructured photonic power splitter design via convolutional neural networks, in: CLEO: Science and Innovations, Optical Society of America, 2019, pp. SW4J–6.

[62] K. Kojima, M. H. Tahersima, T. Koike-Akino, D. Jha, Y. Tang, Y. Wang, K. Parsons, Deep neural networks for inverse design of nanophotonic devices (invited), Journal of Lightwave

Technology (to be published).

[63] Z. Liu, D. Zhu, S. P. Rodrigues, K.-T. Lee, W. Cai, Generative model for the inverse design of metasurfaces, Nano letters 18 (10) (2018) 6570–6576.

[64] W. Ma, F. Cheng, Y. Xu, Q. Wen, Y. Liu, Probabilistic representation and inverse design of metamaterials based on a deep generative model with semi-supervised learning strategy, Advanced Materials 31 (35) (2019) 1901111.

[65] S. An, B. Zheng, H. Tang, M. Y. Shalaginov, L. Zhou, H. Li, T. Gu, J. Hu, C. Fowler, H. Zhang, Multifunctional metasurface design with a generative adversarial network, arXiv preprint arXiv:1908.04851 (2019).

[66] Y. Tang, K. Kojima, T. Koike-Akino, Y. Wang, P. Wu, M. H. Tahersima, D. Jha, K. Parsons, M. Qi, Generative deep learning model for a multi-level nano-optic broadband power splitter, in: 2020 Optical Fiber Communications Conference and Exhibition (OFC), 2020, p. Th1A.1.

[67] K. Sohn, H. Lee, X. Yan, Learning structured output representation using deep conditional generative models, in: Advances in neural information processing systems, 2015, pp. 3483–3491.

[68] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, M. Ranzato, Fader networks: Manipulating images by sliding attributes, in: Advances in neural information processing systems, 2017, pp. 5967–5976.

[69] Y. Wang, T. Koike-Akino, D. Erdogmus, Invariant representations from adversarially censored autoencoders, arXiv preprint arXiv:1805.08097 (2018).

[70] O. Özdenizci, Y. Wang, T. Koike-Akino, D. Erdoğmuş, Transfer learning in brain-computer interfaces with adversarial variational autoencoders, in: 2019 9th International IEEE/EMBS Conference on Neural Engineering (NER), IEEE, 2019, pp. 207–210.

[71] K. Wang, X. Ren, W. Chang, L. Liu, D. Liu, M. Zhang, Inverse design of digital nanophotonic devices using the adjoint method, Photonics Research 8 (2020) 528–533.

[72] Y. Zhang, S. Yang, A. E.-J. Lim, G.-Q. Lo, C. Galland, T. Baehr-Jones, M. Hochberg, A compact and low loss y-junction for submicron silicon waveguide, Opt. Express 21 (2013) 1310–1316.

[73] N. Yu, F. Capasso, Flat optics with designer metasurfaces, Nature materials 13 (2) (2014) 139–150.

[74] C. C. Nadell, B. Huang, J. M. Malof, W. J. Padilla, Deep learning for accelerated all-dielectric metasurface design, Optics express 27 (20) (2019) 27523–27535.

[75] S. An, C. Fowler, B. Zheng, M. Y. Shalaginov, H. Tang, H. Li, L. Zhou, J. Ding, A. M. Agarwal, C. Rivero-Baleine, et al., A deep learning approach for objective-driven all-dielectric metasurface design, ACS Photonics 6 (12) (2019) 3196–3207.

[76] P. R. Wiecha, O. L. Muskens, Deep learning meets nanophotonics: A generalized accurate predictor for near fields and far fields of arbitrary 3D nanostructures, Nano Letters 20 (1) (2019) 329–338.

[77] L. Gao, X. Li, D. Liu, L. Wang, Z. Yu, A bidirectional deep neural network for accurate silicon color design, Advanced Materials 31 (51) (2019) 1905467. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/adma.201905467, doi:https://doi.org/10.1002/adma.201905467.
URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201905467`

[78] Z. Liu, L. Raju, D. Zhu, W. Cai, A hybrid strategy for the discovery and design of photonic structures, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 10 (1) (2020) 126–135.

[79] J. Jiang, M. Chen, J. A. Fan, Deep neural networks for the evaluation and design of photonic devices, Nature Reviews Materials (2020) 1–22.

[80] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[81] F. Wen, J. Jiang, J. A. Fan, Robust freeform metasurface design based on progressively growing generative networks, ACS Photonics 7 (8) (2020) 2098–2104.

[82] S. J. Byrnes, Multilayer optical calculations (2020). arXiv:1603.02720.

[83] T. Asano, S. Noda, Iterative optimization of photonic crystal nanocavity designs by using deep neural networks, Nanophotonics 8 (12) (2019) 2243–2256.

[84] T. Christensen, C. Loh, S. Picek, D. Jakobović, L. Jing, S. Fisher, V. Ceperic, J. D. Joannopoulos, M. Soljačić, Predictive and generative machine learning models for photonic crystals, Nanophotonics 9 (13) (2020) 4183–4192.

[85] H. Kang, T. Wang, Pytorch generative model collections (2017).
URL `https://github.com/znxlwm/pytorch-generative-model-collections`