Modeling and Optimization of Thin-Film Optical Devices using a Variational Autoencoder

John Roberts and Evan Wang, {johnr3, wangevan}@stanford.edu

Introduction

Optical thin film systems are structures composed of multiple layers of different materials. They find applications in areas such as solar cell design, ellipsometry and metrology, radiative cooling, and dielectric mirrors. The main property of interest is the transmission or reflection spectrum, which exhibits a complicated dependence on the parameters of the thin film stack. An open problem in optical design is finding a device that exhibits a desired transmission spectrum, a process known as inverse design. As a model system, we will use unsupervised learning to analyze the transmission properties of a 5-layer stack of alternating glass and silicon layers across a wavelength range of 1000 – 2000 nm. The input features are the layer thicknesses and discretized transmission spectrum. We use a variational autoencoder (VAE) to compress the features down to a latent space and then reconstruct the input.

Related Work

Since optical thin film systems are of great interest to the optics community, there are numerous existing design methodologies. Among them are analytical methods the rely on intuitive descriptions of the underlying physics [1,2]. While analytical design methods are excellent for understanding the physics of thin film systems, they are fundamentally limited in their design space. For better performance and more complex functionality, we must turn to computational methods. Examples include particle swarm optimization [3] and genetic optimization [4]. These methods enable the design of systems with high efficiency, however they can require computationally expensive electromagnetics simulations in more complex systems.

Recently, the has been a surge of interest in applying machine learning and neural networks to tackling the problem of electromagnetic design. The first demonstration involved the design of spherical nanoparticle, a problem very similar to thin films, using a neural network (NN) [5]. The network is trained to simulate the spectrum of a given nanoparticle and backpropagate the gradients to update the input parameters towards the target spectrum. This method is simple but frequently lands in local optima. A global design technique was later demonstrated using a two-part neural network, one for forward simulation and one for inverse design [6]. This tandem network lifts helps alleviate the uniqueness issue – multiple designs can possess similar electromagnetic responses – that often arises when directly training inverse design NNs.

Dataset

We generate our training data using a transfer matrix model (TMM) of the optical properties of five-layer thin-film stacks. In the transfer matrix model, forward- and backward-propagating plane wave modes in each layer are coupled to the modes in each of the adjacent layers. The layer *i* has refractive index n_i and thickness d_i. The total transmission through the thin-film stack can be written in terms of the plane wave field amplitudes as

$$T = \left| \frac{E_{out,+}}{E_{in,+}} \right|^2$$

when the initial and final media are the same. Then the transmission of the total stack can be computed by

$$\begin{pmatrix} E_{in,+} \\ E_{in,-} \end{pmatrix} = I_1 P_1 I_2 \dots P_n I_{n+1} \begin{pmatrix} E_{out,+} \\ E_{out,-} \end{pmatrix}$$

The matrices *I* represent the coupling between modes at the interfaces. Whereas, the matrices *P* represent the phase difference after propagation through a layer. They are calculated as

$$I_n = \frac{1}{2}\begin{pmatrix} 1+A & 1-A \\ 1-A & 1+A \end{pmatrix}, \qquad P_n = \begin{pmatrix} \exp(i\phi) & 1 \\ 1 & \exp(-i\phi) \end{pmatrix}$$

where $A = \frac{n_i}{n_{i+1}}$, $\phi = k_i d_i$ and $k_i = n_i \frac{\omega}{c}$. The procedure is shown schematically in Figure 1a.

For this study, we limit the devices to five-layer glass/silicon/glass/silicon/glass stacks in air. Using the transfer matrix code, we simulate 100,000 random devices for the training set and another 1000 random devices for the test set. The thickness of each layer can take values between 0 and 300 nm. We calculated the transmission at 101 points between $\lambda = 1000$-2000 nm. In this range the refractive indices are approximately constant, around $n_{Si} \approx 3.5$ and $n_{glass} \approx 1.5$. Figure 1b shows an example of a discretized transmission spectrum for a representative device.

The combined input feature vectors consist of 5 normalized thickness values and 101 transmission values for a total of 106 input features.

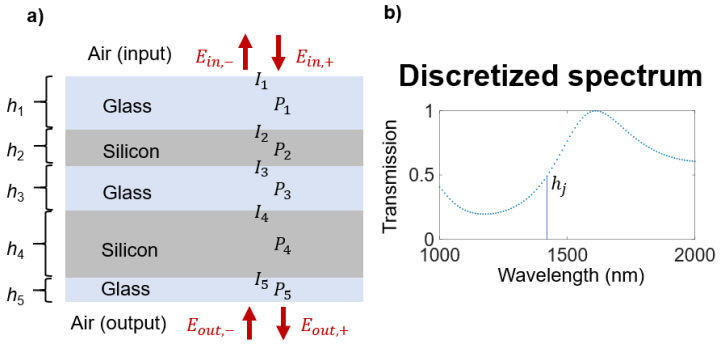

Figure 1: a) Schematic of transfer matrix method. **b)** Representative example of a discretized transmission spectrum

Methods

In order to study the interesting aspects of our data, we train a variational autoencoder, an unsupervised learning algorithm that allows us to compress the input data onto an underlying latent space. However, as a baseline, we first attempt to apply principle component analysis (PCA) in an attempt to achieve the same purpose. In PCA, we compute the principal eigenvectors of the covariance matrix, which is given by

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} x^{(i)T}$$

Taking the *n* largest eigenvectors, ranked by eigenvalue, allows us to reduce the dimensionality of the data from 106 to *n*. This compression is strictly linear, which means it is likely a poor model for the difficult to capture the behavior of thin film systems.

For that, we turn to VAEs; Figure 2 shows the structure of a VAE. In a VAE, the input data is compressed to a latent space using an encoder neural network and then reconstructed with a complementary decoder neural network [7-9]. The entire network is trained at the same time. In a variational autoencoder, rather than encoding to a specific point, we encode each input sample to a Gaussian distribution within the latent space with mean μ and variance σ^2. During training, this distribution is randomly sampled with the result being passed through the decoder. This step forces the VAE to represent similar input vectors near each other and creates high information density in the latent space.

The loss function for a VAE consists of a traditional loss function, in this case the weighted mean square error for reconstruction, as well as a Kullback-Leibler divergence regularization term.

$$l_i(\theta, \phi) = \frac{1}{m}\left(x^{(i)} - \hat{x}^{(i)}\right)^2 + KL(q_\theta(z|x^{(i)})||p(z))$$

Where $x^{(i)}$ is the input, $\hat{x}^{(i)}$ is the reconstructed output, θ are the network weights, z is the latent variable, q_θ is the encoded distribution, and $p(z)$ is the standard normal distribution [7, 9].

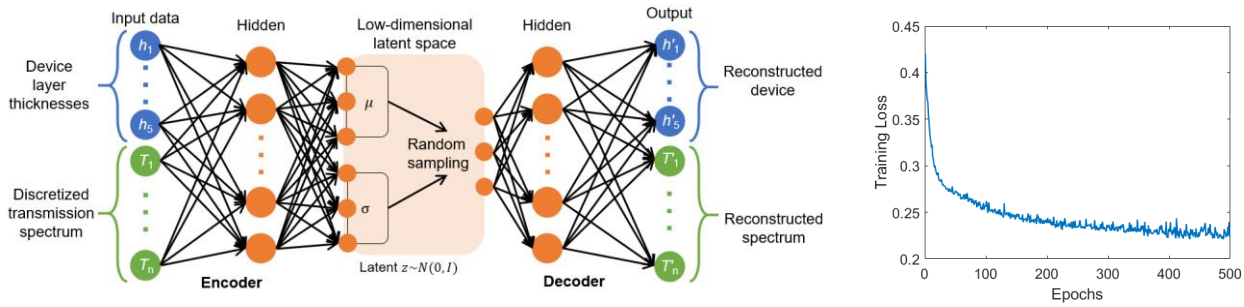

Figure 2: Left: Schematic of variational autoencoder showing input and output vectors, encoder and decoder portions and latent space. Right: Loss function during training.

The KL divergence ensures that the generated latent space distribution is Gaussian. The KL divergences gives the differences between the predicted distributions and the standard normal distribution. The MSE loss, or reconstruction loss, is given by the weighted MSE between the input and reconstructed vectors. The weights are assigned such that the 5 thicknesses have the same total weight as the 101 transmission points.

The VAE implementation we use is based on a PyTorch example by Diederik Kingma and Charl Botha [10-13].

Results and Discussion

In addition to the reconstruction loss, there is another metric of interest in this problem. When the decoder reconstructs a device and corresponding spectrum from the latent space, we need to ensure that the reconstructed device's real spectrum, as computed with the transfer matrix method, and the VAE's predicted spectrum match. The figure of merit for the accuracy of a batch with *m* samples is given by

$$MSE_{accuracy}^{(i)} = \frac{1}{m}\sum_{j}^{m}(T_{j,actual}^{(i)} - T_{j,generated}^{(i)})^2$$

To ensure that the VAE remains consistent with the physics of thin films, we evaluate our model's hyperparameters by checking the accuracy of randomly sampled points in the latent space (instead of using a validation set). Figure 3 demonstrates the procedure for computing the model accuracy. We first sample 100 random points in the latent space and decode them into a device and a spectrum. We compute the actual spectrum of the generated device using the transfer matrix model and compare it to the generated spectrum.

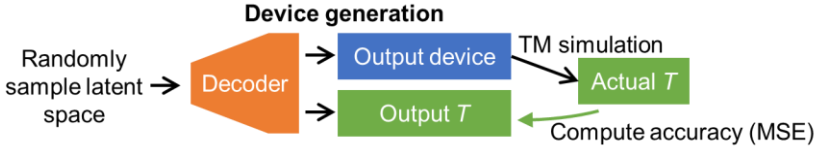

Figure 3: Procedure for assessing accuracy of VAE model by generating random examples. Random points are sampled from the latent space and decoded. The accuracy is defined as the MSE between the reconstructed spectrum and the real spectrum of the reconstructed device.

Since the transfer matrix method allows us to fully describe the spectrum with only the five thicknesses as input, we wanted to see if compression to an even smaller dimension was possible, thus we optimized the network for three latent dimensions. We compared the accuracy MSE for different numbers of hidden neurons and mini-batch sizes. Table 1 shows the results of the tests which resulted in our final architecture of one hidden layer for the encoder and one for the decoder, each with 80 neurons. The encoder uses a ReLU activation function and the decoder uses a sigmoid activation function to ensure that the normalized device thicknesses and transmission spectra are in the range [0, 1]. We trained the VAE with a mini-batch size of 100 samples.

Table 1: VAE Hyperparameter Tuning

Mini-Batch Size	Hidden Neurons	Accuracy (MSE)
100	20	1.86
100	50	1.43
100	**80**	**1.17**
100	200	1.7
1000	80	1.255
PCA Baseline		17.81

After training the VAE, we plot some of the devices and spectra that are generated for a qualitative assessment of the reconstruction and accuracy. Figure 4 shows reconstructed spectra from the test set. The spectrum on the left is poorly reconstructed and the reconstruction has poor accuracy. The spectrum on the right shows both good reconstruction and accuracy. Overall, it would be desirable to achieve greater accuracy in the model, so that the model prediction captures the physics of thin films. Our accuracy is likely limited by the latent space dimensionality.

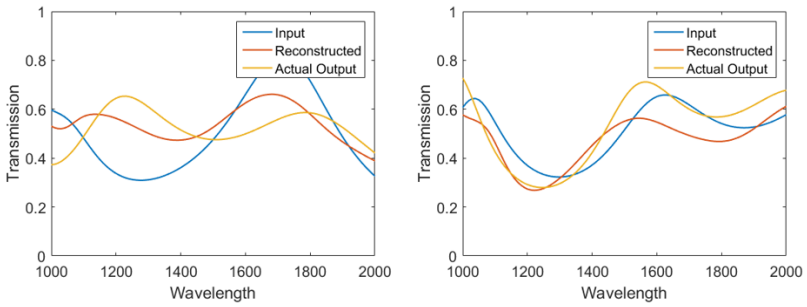

Figure 4: Example spectra sampled from the test set showing a comparison between the input, reconstructed and actual output spectra.

To understand the properties of the encoded latent space, we systematically sampled the latent space on a three-dimensional grid and decoded the sampled points. The decoded spectra vary smoothly across the latent space. The latent space variables appear to be strongly correlated with layer thicknesses (Figure 5), implying that the VAE learns to encode the data using the thickness parameters that generated the spectra. Significantly, two of the three latent space variables are always strongly correlated with the thicknesses of the two silicon layers (Table 2). Variations in these layers cause larger changes than variations in the glass layers because of silicon's significantly larger refractive index ($n_{Si} \approx 3.5$, $n_{glass} \approx 1.5$). This result demonstrates that when the latent space is low-dimensional enough that it cannot completely represent the degrees of freedom of the problem, the VAE learns to encode the more physically important features. This suggests that VAEs can be used to approximately simulate the physics of thin films, and potentially other, more complex devices, using a less complex representation.

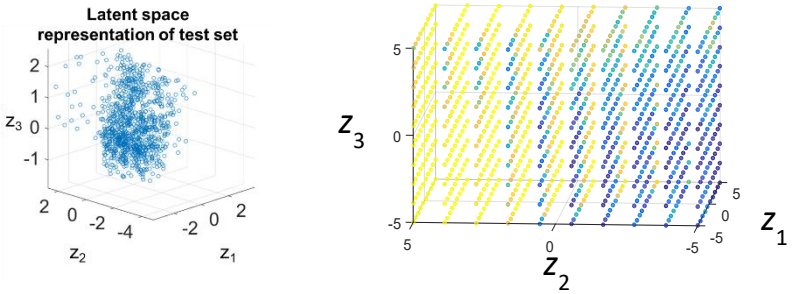

Figure 5: (Left) Latent space representation of the test set. (Right) Decoded thickness parameter h_2 (thickness of the first silicon layer) after sampling on a grid in the latent space with the trained VAE. The latent parameter z_2 appears to be strongly correlated with h_2.

Table 2: Correlation between latent variables and decoded layer thicknesses based on a grid sampling of the latent space. Each entry is a correlation cor(z_i, h_j). The layer thicknesses h_2 and h_4 correspond to the high-index silicon layers, which have a larger effect on the spectrum than the three glass layers.

	h_1	h_2	h_3	h_4	h_5
z_1	-0.23	0.33	0.34	**0.82**	-0.22
z_2	-0.04	**0.80**	0.10	-0.24	0.16
z_3	**-0.84**	0.02	-0.06	-0.11	-0.49

Finally, with our trained VAE, we attempt to use the model for inverse design, the process of generating devices that exhibit a desired spectrum. Our approach utilizes the property that VAEs are robust to input noise. Because the input data is compressed to a very small latent space, small perturbations in the often lead to the same point in latent space. Figure 6 illustrates the method. We pair our target spectrum with random thickness values and input them into the VAE. After the input is encoded and decoded in the VAE, we examine the output device. In principle, if the randomly generated device is close in latent space to a device that actually exhibits the target spectrum, our output should be close to that device.

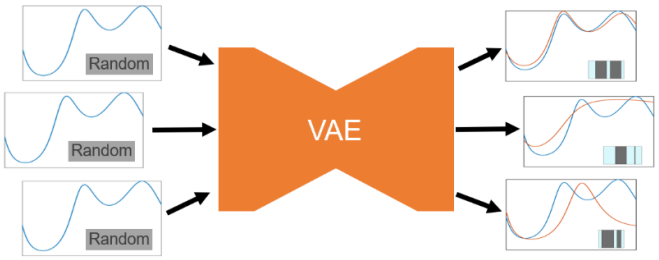

Figure 6: Application of VAE for inverse design. A random device is input with the target spectrum. Example outputs are shown on the right.

We perform this for a batch of 100 random devices for a target spectrum. Some outputs are shown on the right of Figure 6. In this case we can see that at least one out of the 100 random inputs generated a device that exhibits the target spectrum. There is a flaw to this method: because of the symmetry between the devices and spectra, we are just as likely to receive the output spectrum for the random device as we are to receive the output device for the target spectrum. Thus, while this type of method shows promise in theory, some modifications are required for a successful implementation.

Conclusion and Future Work

We use a variational autoencoder to represent optical thin-film stacks and their transmission spectra in a low-dimensional latent space. We show that it is possible to represent thin-film devices in a latent space distribution, and to generate new devices and their approximate spectra by sampling from the latent space. When the latent space is lower-dimensional than the degrees of freedom of the thin-film stack, the latent space variables are strongly correlated with the more physically important parameters.

In the future, a modified network architecture may be required to perform efficient inverse design. In addition, the accuracy of our model is likely limited because the latent space is lower-dimensional than the problem. While this reveals interesting behavior of the VAE, a more accurate model for the purpose of inverse design could be implemented by increasing the number of latent space dimensions.

Code

Our code is available at:
https://drive.google.com/drive/folders/1knAhigCB4OEyxg_z8TPT1KT4notkD3DS?usp=sharing

Contributions

John wrote the transfer matrix code. Evan ran the dataset generation. John set up the framework for the VAE code. Evan ran the training and sample generation. John mapped the latent space. Evan implemented the inverse design.

References

(1) Gerken, M.; Miller, D. A. B. Multilayer Thin-Film Structures with High Spatial Dispersion. Appl. Opt. 2003, 42, 1330−1345.

(2) Shen, Y.; Ye, D.; Celanovic, I.; Johnson, S. G.; Joannopoulos, J. D.; Soljacič , M. Optical Broadband Angular Selectivity. Science 2014, 343, 1499−1501.

(3) Rabady, R. I.; Ababneh, A. Global Optimal Design of Optical Multilayer Thin-Film Filters Using Particle Swarm Optimization. Optik 2014, 125, 548−553.

(4) Shi, Y.; Li, W.; Raman, A.; Fan, S. Optimization of Multilayer Optical Films with a Memetic Algorithm and Mixed Integer Programming. ACS Photonics 2018, 5, 684-691.

(5) Peurifoy, J. E.; Shen, Y.; Jing, L.; Cano-Renteria, F.; Yang, Y.; Joannopoulos, J. D.; Tegmark, M.; Soljacic, M. Nanophotonic Inverse Design Using Artificial Neural Network. Science Advances 2018, eaar4206

(6) Liu, D.; Tan, Y.; Khoram, E.; Yu, Z. Training Deep Neural Networks for the Inverse Design of Nanophotonic Structures. ACS Photonics 2018, 5, 1365-1369.

(7) C. Doersch, "Tutorial on Variational Autoencoders," arXiv:1606.05908v2 [stat.ML], 2016.

(8) R. Gómez-Bombarelli *et al*., "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules, ACS Central Science 2018, 4, 268-276.

(9) J. Altosaar, "Tutorial – What is a variational autoencoder?", https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

(10) Botha, C. Variational Autoencoder in PyTorch, commented and annotated. 2018. [online] vxlabs. Available at: https://vxlabs.com/2017/12/08/variational-autoencoder-in-pytorch-commented-and-annotated/ [Accessed 20 Nov. 2018].

(11) "Basic VAE Example", https://github.com/pytorch/examples/tree/master/vae

(12) Paszke, A.; Gross, S.; Chintala, S. and Chanan, G. PyTorch. 2017.

(13) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O. and Vanderplas, J. Scikit-learn: Machine learning in Python. Journal of machine learning research 2011, 2825-2830.