Photonics Inverse Design: Pairing Deep Neural Networks With Evolutionary Algorithms

Ravi S. Hegde

***Abstract*—Deep Neural Networks (DNN) have shown early promise for inverse design with their ability to arrive at working designs much faster than conventional optimization techniques. Current approaches, however, require complicated workflows involving training more than one DNN to address the problem of non-uniqueness in the inversion and the emphasis on speed has overshadowed the far more important consideration of solution optimality. We propose and demonstrate a simplified workflow that pairs forward-model DNN with evolutionary algorithms which are widely used for inverse gg design. Our evolutionary search in forward-model space is global and exploits the massive parallelism of modern GPUs for a speedy inversion. We propose a hybrid approach where the DNN is used only for preselection and initialization that is more effective at optimization than a standalone DNN and performs nearly as well as a vanilla evolutionary search with a significantly reduced function evaluation budget. We finally show the utility of an iterative procedure for building the training dataset which further boosts the effectiveness of this approach.**

***Index Terms*—Optical system design, optics and lens design, thin films, deep learning, artificial intelligence, evolutionary optimization.**

I. Introduction

DETERMINING a set of globally optimal design parameters for a photonic device to meet a specified response is a challenging inverse problem. that is assuming an increasing importance as advancements in fabrication techniques and materials research allow us to explore complex structure and material combinations. While advances in computational power have made forward simulations faster, inverse problems, especially those involving large degrees of freedom, remain challenging due to the so-called "curse of dimensionality" [1] (a term that refers to the exponential explosion of the search space volume with linear increase in dimensionality). Optimal photonics design is thus either restricted to limited searches in global space (limited due to the large number of computations required) or to gradient based searches that tend to get stuck at local optima [2]. The current resurgence of neural computing in the form of Deep Learning (DL) [3] has raised the intriguing possibility of artificial intelligence (AI) based methods that could potentially overcome the "curse of dimensionality". Application of DL has shown early promise in the design of optical thin-films [4], nanostructures [5]–[8] metasurfaces [9]–[11] and integrated photonics [12], [13]. Inverse design could be a potential application of all-optical neural networks given their unique attributes.

Inverse design problems exhibit two key differences in comparison to typical pattern-recognition tasks (where DL techniques have shown remarkable success): (1) a given response can be realized through multiple designs, while a single design has a unique response (non-uniqueness); (2) accuracy of the solution and its optimality is far more important than split-second turnaround times (typical optimizations take many hours). Peurifoy and coworkers [14] achieve inversion by freezing the weights of the trained forward DNN and performing a gradient descent search over the input parameters. This approach amounts to a local search in the approximate fitness landscape and is highly dependent on a good initialization and domain-wide model accuracy to succeed. The tandem network approach (Liu and coworkers [4]) and the generative networks with critic moderation approach (Cai and coworkers [5]) train a second DNN to speed up this inversion. Subsequent workers have continued to use two or more DNNs [7], [9], [12].

Having multiple networks increases the burden of model development, but a far more insidious problem is that the overall model accuracy drifts that occur around the blind-spot regions of the forward-model. It is well known that DNNs tend to perform well on problems where most of the input information can be discarded as "noise" and perform somewhat poorly on problems which are sensitive to input parameter values [15]. Furthermore, DNNs do not adequately model their own prediction uncertainty [16]. The implication is that a trained DNN may perform quite well on the training data as well as on test data, but may model some regions of the design space poorly affecting its ability to arrive at optimal designs. To the best of our knowledge, it remains unclear whether DNNs acting alone can outperform evolutionary algorithms with regards to optimality.

The problem of multilayered thin-film design, in particular, the problem of broadband antireflection coating (ARC) design has received extensive attention from researchers [17]–[21] and a broad range of theoretical and computational techniques (including DE) [21]–[25] have been applied to it. For our purpose, we note that this is a challenging multi-modal optimization problems with regions of flat fitness [17], [21], [26] with strong mathematical and computational evidence regarding the existence of global optima [18]–[20].

Differential Evolution (DE) [27] is the evolutionary algorithm used here and fitness estimation is carried out in three ways:

Manuscript received April 1, 2019; revised July 17, 2019; accepted July 30, 2019. Date of publication August 7, 2019; date of current version August 21, 2019.

The author is with the Department of Electrical Engineering, Indian Institute of Technology, Gandhinagar 382355, India (e-mail: hegder@iitgn.ac.in).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTQE.2019.2933796

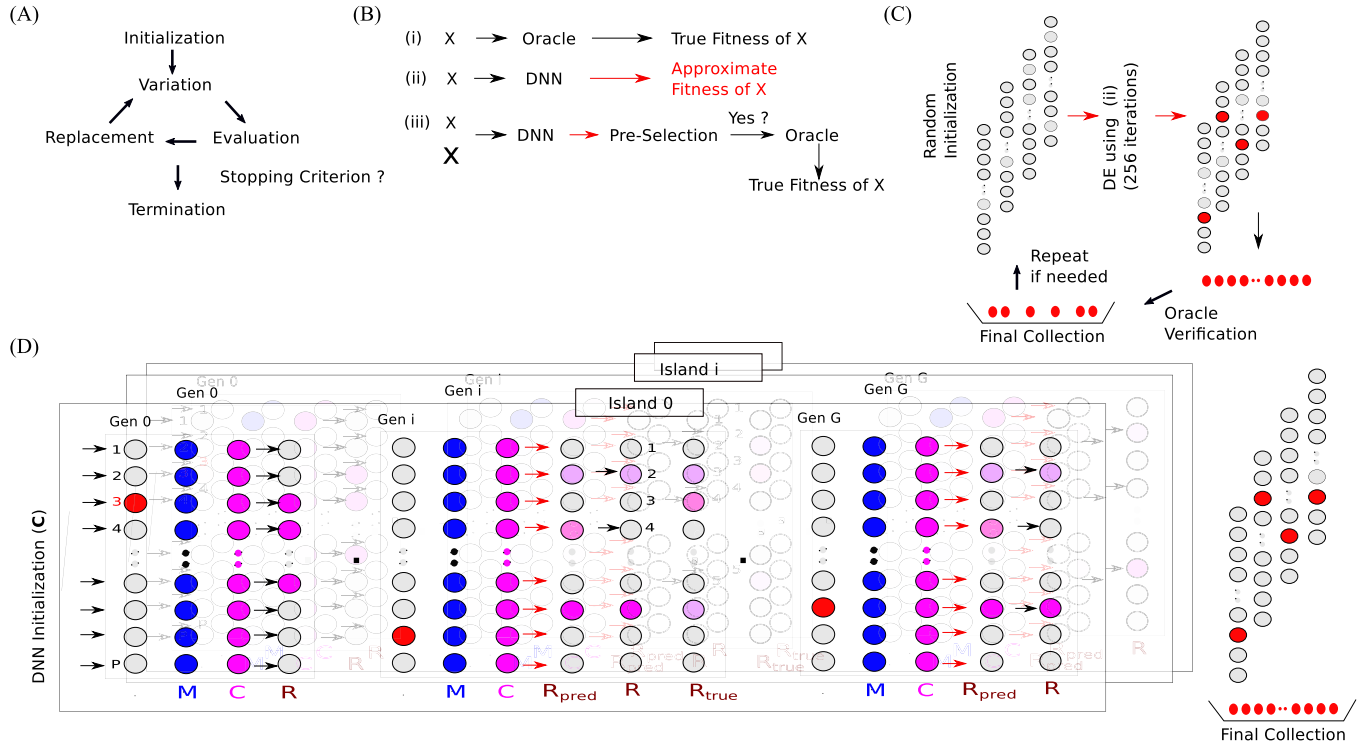

Fig. 1. Overview of DNN-assisted DE. A shows a general description of evolutionary optimization. B the three different ways in which the evaluation step in A can be accomplished. C Global search of the DNN-model space using repeated runs of the DE algorithm using approximate fitness evaluations on multiple islands. D detailed pictorial depiction of the multiple-island DE with initializations borrowed from step C. Filled circles in C, D denote the "individual", a multilayered thin-film design. In each generation, a new population of individuals is created by mutation (M) and crossover (C) and repopulation (R) phases. The merit evaluations can be performed exactly (black arrows) or approximately via the DNN surrogate (red arrows). Red circles denote the best individual of the population at the start/end of the iteration. Blue and purple (and shades of these colors in (b)) represent mutated and crossover variants. The difference of our approach compared to vanilla DE occurs in the repopulation phase where the DNN points towards R_{pred}. Some exact function calls are used to verify and replace the population into R which could be different from the correct case R_{true}. Numbers 1 to 4 show the various possibilities resulting from approximating the repopulation phase.

(1) exactly using a so-called "Oracle" (vanilla DE), (2) approximately, using DNNs under different conditions, and (3) exactly using the oracle but only on a reduced set as determined by the DNN. (2) is used for inversion and (3) is termed DNN-assisted DE. Using this methodology, we can smoothly trade-off speed vs optimality and arrive at guidelines on how to develop the DNN. The assisted DE used here has clear inspirations in surrogate-assisted optimization. Artificial Neural networks, the predecessors of DNNs have been previously explored in the acceleration of evolutionary algorithms [28], [29] but were not found to be as effective. Our approach deviates from traditional surrogate-assisted optimization, in that DNN models are used only for initialization and preselection. To the best of our knowledge, the methodology as well as the focus on optimality of various approaches is the novelty of this contribution. Finally, we incorporated an uncertainty estimation in the loss function of the network and trained it on a smaller dataset and show that such active sampling can further boost the effectiveness of this approach. After this introduction, a detailed description of the proposed algorithm is found in Section II. The results and discussion section (Section III) begins with a description of the DNN model training and testing. The performance of the DE algorithm using various DNNs is then described before concluding the paper.

II. DNN-Assisted DE

Evolutionary optimizations are a broad class of algorithms inspired by biology and the general flow is summarized in Figure 1A. The purpose of this section is point out how our algorithm differs from the conventional DE and not to describe it in detail. Figure 1B shows three ways to perform the evaluation phase: (1) exactly, (2) approximately and (3) exactly over a reduced set. While approach (2) could lead to inaccurate optima, the worst outcome of using approach (3) is impeded optimization. Figure 1C shows the procedure to run an optimization entirely using approximate fitness evaluation. Multiple rounds of optimization are run (since approximate fitness evaluation is cheap, this step is not time consuming) where a verification step is run to discard designs that may have been wrongly obtained by approximate evaluation.

Figure 1D shows the detailed pictorial representation of assisted DE optimization. Multiple islands with their own population are iterated in parallel with the possibility of periodic migration of individuals between islands. We consider a single island model for the following description. If we denote the j-th individual at the beginning of the i-th iteration as p_j^i, then the set of designs $p_j^0,\ 0 \leq j \leq P-1$ is the initial population. Before beginning the optimizations, a set $F_j^0,\ 0 \leq j \leq P-1$

of real numbers called the fitness vector is generated by calling the merit function on this initial population. The fitness vector is updated at the end of every iteration and at the end of the i-th iteration is denoted as F_j^i.

Each iteration begins with the mutation (M) phase followed by a crossover (C) phase which results in another set of P individuals called mutants (j-th mutant during the i-th iteration is denoted by m_j^i). Another set of real numbers called the mutant fitness vector is calculated at every iteration by calling the merit function on the set of mutants which is denoted as f_j^i, $0 \leq j \leq P-1$. The process of generating the j-th mutant involves an intermediate individual generated by the mutation phase as described by: $o_j^i = a_j + r_{mut} * (b_j - c_j)$ and $a_j, b_j, c_j \in \{p_0^{i-1}, p_1^{i-1}, \ldots, p_{P-1}^{i-1}\} - \{p_j^{i-1}\}$, where the a, b, c individuals are randomly drawn for each individual and r_{mut} is a hyperparameter of the optimization called the mutation radius.

Finally, the j-th mutant is obtained using p_j^{i-1} and o_j^i in the crossover phase as:

$$m_j^i[d] = \begin{cases} o_j^i[d], & \text{if } p >= p_{cross} \\ p_j^{i-1}[d] & \text{otherwise} \end{cases}, \quad 0 \leq d \leq 15. \tag{1}$$

Crossover is an element-wise operation on the individual (which is a 16-dimensional vector) and involves drawing a random number p for each of the elements. This randomly drawn number is compared with p_{cross} (another hyperparameter of the optimization called the crossover probability) to generate the mutant. The iteration ends with a repopulation (R) phase that results in the next generation of individuals and can be summarized as:

$$\begin{cases} p_j^i = m_j^i, F_j^i = f_j^i & \text{if } f_j^i > F_j^{i-1} \\ p_j^i = p_j^{i-1}, F_j^i = F_j^{i-1} & \text{otherwise} \end{cases}, \quad 0 \leq j \leq P-1, \tag{2}$$

where the updation of the fitness vector is also shown. Note that it is not necessary to store the fitness vectors at each iteration. For N iterations, this results in $(N+1) * P$ function calls.

In the repopulation phase of the proposed algorithm, the exact computation of the mutant fitness vector f_j^i for all the P members is replaced by exact computation only on a subset of the mutants. To determine the subset for which exact computation is neccesary, two auxiliary fitness vectors g (analogous to f) and G (analogous to F) are calculated using the DNN. The modified repopulation phase that results in the next generation of individuals can be summarized as Eq. (3) shown at the bottom of this page, where the updation of the fitness vector is also shown. The last difference is that several mutation and crossover phases can occur in parallel in the same iteration. The final mutant population in a particular iteration can then be selected by comparing the various intermediate fitness vectors.

There are several potential consequences of this approximation phase and all these possibilities are noted in Figure 1D. and can be summarized as: (1) $g_j^i < G_j^{i-1}, f_j^i < F_j^{i-1}$: one function call is correctly saved; (2) $g_j^i > G_j^{i-1}, f_j^i > F_j^{i-1}$: optimization has advanced correctly; (3) $g_j^i < G_j^{i-1}, f_j^i > F_j^{i-1}$: optimization has been impeded, no unnecessary function calls; and, (4) $g_j^i > G_j^{i-1}, f_j^i < F_j^{i-1}$: optimization not impeded, but unnecessary function call made.

III. Results and Discussion

The open-source python program TMM from Byrnes [30] (with some modifications that allow it to calculate the spectrum in parallel) is used as the "oracle". DNNs are implemented with the Keras library [31] running on MXnet backend [32]. The DE component was written in python from scratch with separate CPU and GPU versions and we ensured that unnecessary data transfers between the CPU and the GPU are avoided. The source code for the implementation, datasets and saved models have been made available [33]. The following results are obtained by running our implementation on a workstation with a Intel$^{\text{TM}}$ i9–7920X CPU with 3 NVIDIA$^{\text{TM}}$ GeForce GTX 1080 GPU cards with 12 GB memory. For the evaluation of the spectrum of 100,000 geometries, the DNN (M1) took 479 ms. The exact calculation using the TMM python package with python based multiprocessing library parallelism using 16 cores for 100,000 geometries took 6 minutes and 19 s and about 1 hour 5 minutes for a single-threaded version.

A. DNN Architecture, Training, and Testing

We extensively experimented with the number of neurons in each layer, different activation functions, batch normalization and dropout layers, replacement of fully connected layers with convolutional layers and various training methodologies (SGD, ADAM, N-ADAM etc) [31]. The number of trainable parameters was found to correlate most strongly with the accuracy. RELU and Swish [34] activations performed nearly equally well and were substantially better than other activations. Batch normalization and dropout were not found to be beneficial. ADAM and N-ADAM training procedures were found to significantly outperform SGD. Preliminary experiments with convolutional layers showed some promise. We could reduce the number of trainable parameters with minor degradation in accuracy; however, convolutional layers took more time to train.

The final choice of the model was a feedforward neural network with fully connected layers as seen in Figure 2 with H hidden layers each with N number of neurons (the notation N x H used henceforth). Swish activation [34] was used on all but the last layer where sigmoid activation was used to bound the output between 0 and 1. The structure is an air-clad 16-layered

$$\begin{cases} \begin{cases} p_j^i = m_j^i, F_j^i = f_j^i & \text{if } f_j^i > F_j^{i-1} \\ p_j^i = p_j^{i-1}, F_j^i = F_j^{i-1} & \text{otherwise} \end{cases} & \text{if } g_j^i > G_j^{i-1} \\ p_j^i = p_j^{i-1}, F_j^i = F_j^{i-1} & \text{otherwise} \end{cases}, \quad 0 \leq j \leq P-1 \tag{3}$$

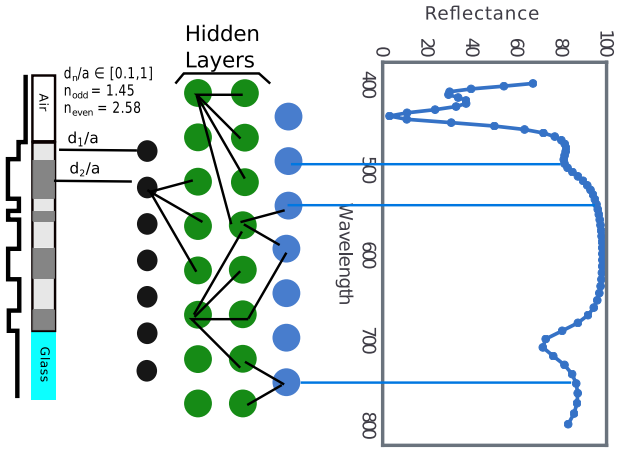

Fig. 2. Overview of the DNN architecture. The neuron color is chosen to denote the activation function (black = linear, green = "Swish", blue = Sigmoid). Representative diagram not showing all connections.

thin-film made of alternating layers of silica [35] and titania [36] on a semi-infinite substrate of refractive index 1.52. The maximum and minimum thickness of each layer are $d_{\min}$ and $d_{\max}$ respectively and $d_{\min} = 0.1\ d_{\max}$. The reflectance spectrum at normal incidence of any design X at 64 points equally spaced in frequency between frequencies corresponding to a minumum and maximum wavelengths of 400 nm and 800 nm respectively.

We first studied the influence of problem complexity (determined by the domain constraints), model complexity (N and H) and training data size on the training (Figure 3) of the DNN. Traditionally, the mean squared error (MSE) [4] and the mean relative error (MRE) [14] have been used to quantify accuracy. We found that MSE is a good loss function to train the network but peak absolute error is a better indicator of the generalization ability of the trained DNN. For the experiment in Figures 3A, a 128 x 3 network is trained on datasets with fixed minimum and maximum layer dimensions of 10 nm and 100 nm. Only the size of the dataset is varied (50k stands for 50,000 samples). It can be seen that 100k is about the optimal size and that further increase in size shows diminishing returns. For the experiment in Figures 3B, the dataset size is fixed at 200k, dimension range is fixed at (10, 100) and the number of hidden layers is varied in the DNN. Once again, it is seen that for a fixed dataset size, simply increasing the model complexity leads to diminishing returns. Finally, the experiment in Figures 3C, shows that expanding the constraints imposed on design parameters increases the VC dimension of the problem and it becomes increasingly difficult to train a DNN. The summary observation is that extensive experimentation is needed in order to identify the correct model size and training dataset sizes and that different networks should be compared keeping the same problem complexity and dataset sizes.

Are the models with the least training error the best ones? This can only be answered with cost/benefit thinking as models with reduced error are harder to develop and require extensive experimentation. For the remaining set of experiments, we consider six possible models listed in Table I. These possibilities coarsely represent the cost/benefit tradeoff involved in DNN development. Note that the generation of 400k sized dataset takes nearly 2 hours on our system in comparison to about 12 minutes for the 40k sized dataset. The models were all trained on datasets sampled in the (10 nm, 100 nm) range for each layer thickness. Two sampling techniques were considered: (1) latin hypercube sampling (this ensures maximum separation between the training data points), (2) latin hypercube sampling on a larger hypercube followed by clipping leading to many datapoints lying exactly on the boundary of the constraint box. Modern DL platforms can handle very large networks but larger dataset sizes continue to be harder to obtain for many problems. Thus, we do not consider the combinations with small models and large datasets.

The six models were tested on a separately generated dataset of size 100k and Figures 4 summarizes their testing performance. The test and the train errors (MSE) are seen to track well in all cases as seen in Figures 4A with a hint of divergence seen for M5 and M6. This shows that a small amount of overfitting is expected when dataset sizes are smaller than ideal. With L2 regularization of about 1e-5 added, we observed that this overfitting could be reduced slightly without unduly affecting convergence or training time. Figures 4(B–E) indicate that MSE is a more optimistic picture than the peak error; MSE is narrowly distributed while the peak error shows a flatter distribution with very large values possible. A surprising result in seen when we compare Figures 4C and D. Although, the MSE for model M1 is far better than that of M3, both are seen to have nearly the same number of data points where their prediction leads to a large peak error. Figure 4E shows that model M4 shows a much reduced rate of high peak errors in comparison to M1 which occurs because the training dataset and testing dataset were sampled on the same distribution unlike that of M1. Figure 5 shows the predicted spectrum of all 6 models on the same design for a longpass filter. A surprising observation is that even the coarser models seem to approximate the "shape" of the spectrum quite well.

B. Evolutionary Search in Model Space

The performance of a design search is measured by the use of merit functions. In this subsection and the rest of the paper, the mean reflectance over the 400 to 800 nm spectral windown is used as the merit function:

$$\frac{\sum_{\lambda=1}^{64} R(\lambda_i)}{64}, \tag{4}$$

where λ_i denotes the i-th sampling point between the wavelengths of 400 nm and 800 nm. The results of the evolutionary search of the model space is reported in Figure 6. The GPU version of the DE algorithm employed 300 islands each with population P of 100 and ran for 256 iteration with a mutation radius r_{mut} of 0.8 and crossover probability p_{cro} of 0.7 using only a single GPU. In comparing the achieved reflectances, we note that reports in literature have established that for normal incidence, two material systems provide the best designs for ARC. The optimal reflectance achievable is known to depend on the highest and the lowest refractive indices available, the optical thickness

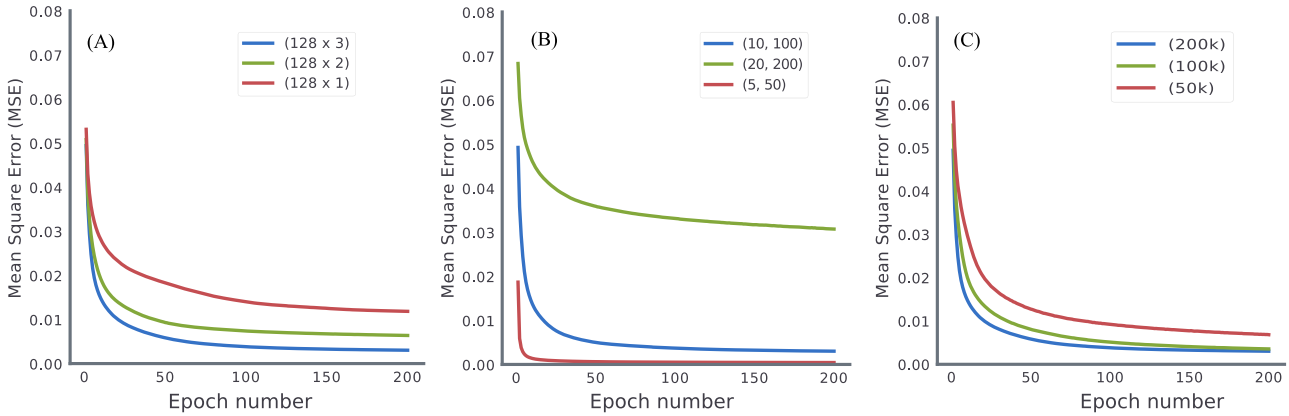

Fig. 3. Training error (MSE) progression over 200 epochs for various combinations of network architecture, training dataset size and layer size range. A: Varying network (128 neurons in 1, 2 and 3 hidden layers), identical size range (10 nm, 100 nm) and training dataset size of 200k. B: Identical network (128 neurons in 3 hidden layers) and size range (10 nm, 100 nm), varying training dataset size 50k, 100k and 200k. C: Identical network (128 neurons in 3 hidden layers) and training dataset size of 200k, but varying size ranges (5 nm, 50 nm), (10 nm, 100 nm) and (20 nm, 200 nm).

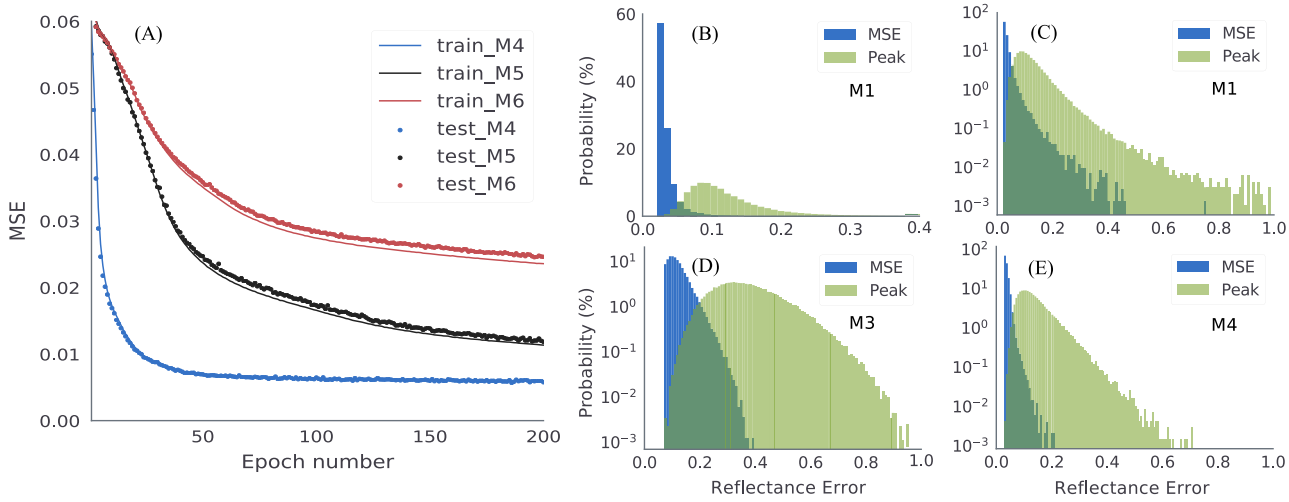

Fig. 4. DNN performance on the same unseen test data (size 100,000) for the 6 models shown in Table I. A shows the progression with training epochs of the training and the test error for M4, M5 and M6. The curves for M1 were nearly identical to that of M4, M2 to that of M5 and M3 to that of M6. B–E are histograms of the mean and peak absolute errors for models M1, M1, M3 and M4 respectively. In C–E the percentage probability is seen on a logartithmic scale.

TABLE I
MODEL SUMMARY. TRAIN SIZE S_t: LARGE (L) = 400K, SMALL (S) = 40K. MODEL SIZE S_m: LARGE (L) IS 128 X 3, SMALL (S) IS 128 X 1. BIAS: LH + BW IS LATIN HYPERCUBE SAMPLING (WITH CLIPPING), U IS UNIFORM RANDOM SAMPLING. THE TRAINING USED 1 GPU AND TRAIN TIME FOR ONE EPOCH t_T IS REPORTED

Name	S_t	S_m	Bias	MSE	Peak Err	t_T
M1	L	L	U	0.0012	0.23	28s
M2	S	L	U	0.0099	0.46	19s
M3	S	S	U	0.0183	0.63	15s
M4	L	L	LH + BW	0.0013	0.25	28s
M5	S	L	LH + BW	0.0131	0.50	19s
M6	S	S	LH + BW	0.0240	0.66	15s

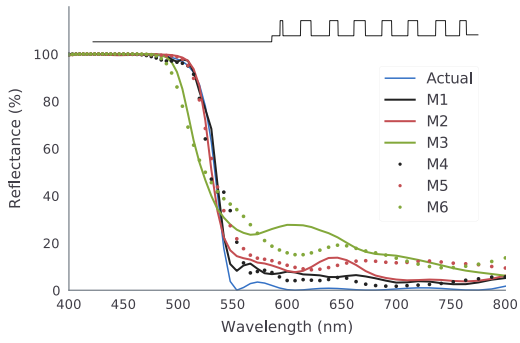

Fig. 5. Comparison of the prediction of all six models on an unseen design against the exact spectrum.

as well as the number of layers [17]–[20]. For the silica/titania system, it has been reported that the best achievable mean reflectance may be expressed as: $R_m = 0.1029 \times d_{opt}^{-0.9758}$, where R_m is expressed in percentage and d_{opt} is the optical thickness expressed in microns. In our case we consider designs where

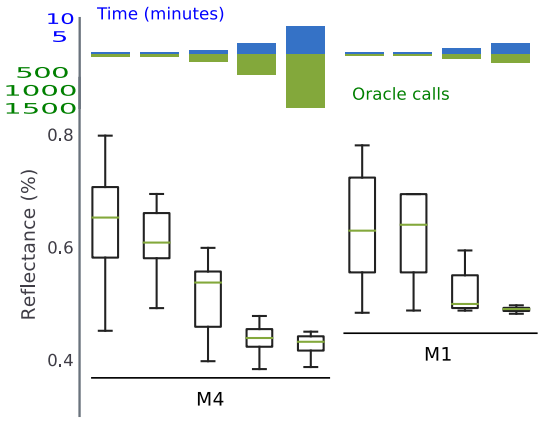

Fig. 6. Evolutionary search in the model space of M1 and M4 models. 20 results were collected and the mean reflectance of each of these designs, the time taken and the number of verfication calls are shown.

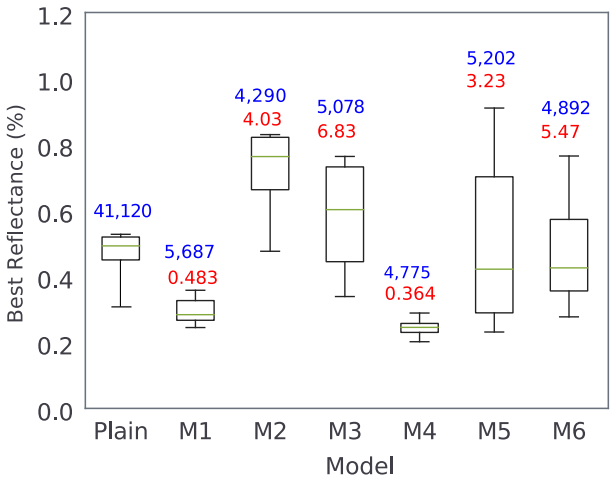

Fig. 7. DNN-Assisted DE Performance for a ten-island run with 160 individuals without intermigration using all the six models is compared against that of a similiar vanilla DE. The best reflectance values of the ten islands obtained after 256 iterations are shown. The blue text over boxes represent the number of function calls made (per island) and red text represents the best fitness of the initializing population.

the layer dimensions is in the range (10 nm, 100 nm) and thus we can conclude that the exact optima must lie in the 0.1% to 0.2% range. Models M1 and M4 approach the global optima closely, but the remaining models fared much worse (the best fitness obtained was as follows: M2 - 4.03, M5 - 3.23, M3 - 6.83 and M6 - 5.47). From previously published reports [17]–[20], it is apparent that optimal ARC designs are lying at the boundary of the design space. As M4 is trained on boundary weighted dataset, it performs best in initializations in comparison to M1.

C. Performance of Assisted-DE

From this experiment, it appears that the best strategy is to build bigger DNNs and train them on as large a dataset is possible. A different picture emergers in Figure 7, where we compare the performance of assisted DE with the vanilla DE. The results are judged on three metrics: (1) how many exact function calls are used; (2) the best fitness achieved and (3) the variance in the best fitness. DE is a stochastic algorithm and hence ten runs are considered to show the range of possibilities. The important observation is that the assisted DE performs better than plain DE as well as DNN acting alone for some cases. At first glance, this appears to be due to the initializations, but, even an inferior model, the M6, seems to do nearly as well as M4, albeit with a larger variance. The assisted DE is seen to result in a drastic reduction in the number of exact function calls. Nearly all the versions resulted in only 10% exact function calls and ran 5 to 6 times faster.

Assisted DE using coarser models M6 and M3 are seen to have a large variance. In Figure 8A, we show a technique to deal with the large variance seen in assisted DE with coarser models. Unlike the previous runs, where the population count was 160, we reduced the population count to 80 and ran DE on 4 islands simultaneously. At the 128th and 192nd iteration mark, the best individuals in the islands were transmigrated in a round-robin fashion among these four islands. This procedure was repeated 8 different times and the best fitness achieved each time and the number of exact function calls are noted in Figure 8B. Although, using only 20% of the function calls of the plain version and an inferior DNN model, we see that we achieve comparable performance to it. Mutation radius and crossover probability are the hyperparameters of the assisted DE. In Figures 8C and 8D, we studied how the choice of these parameters influence the performance. It is seen that a mutation radius of 0.4 and crossover probability of 0.4 works best for both M4 and M6 models.

D. Active Sampling

We finally investigate the possibility of improved performance with smaller datasets. A DNN is first trained on a smaller dataset of size 50k with a modified loss function. The modified loss function between the true output $\hat{y}$ and the prediction of the DNN y is:

$$L(y,\hat{y}) = \frac{\sum_{i=1}^{B}\left(\frac{(y-\hat{y})^2}{2\sigma_i^2} + \frac{1}{2}\log\sigma_i^2\right)}{B} + \lambda\sum_{j=1}^{K}|w_k|^2, \tag{5}$$

where B is the batch size and K is the total number of weights of the DNN w and λ is the regularization constant. σ_i is a parameter that the DNN learns for each of the data point in an unsupervised manner. During training, the DNN can minimize loss by two ways: (1) make the mean error as small as possible, (2) make the variance large. The logartithmic term punishes large variance values and hence the DNN after training tends to chose a high variance only for cases where it cannot match the desired output.

A 128 x 3 network was first trained on a 50k dataset and showed a median variance of about 6.8%. Subsequently, a second dataset of 50k was added to this choosing only samples which had predicted variances larger than 6.8%. The network weights were then reset, the loss function was changed back to the simpler form without variances and was trained on this 100k dataset for 256 epochs with a batch size of 64. This DNN

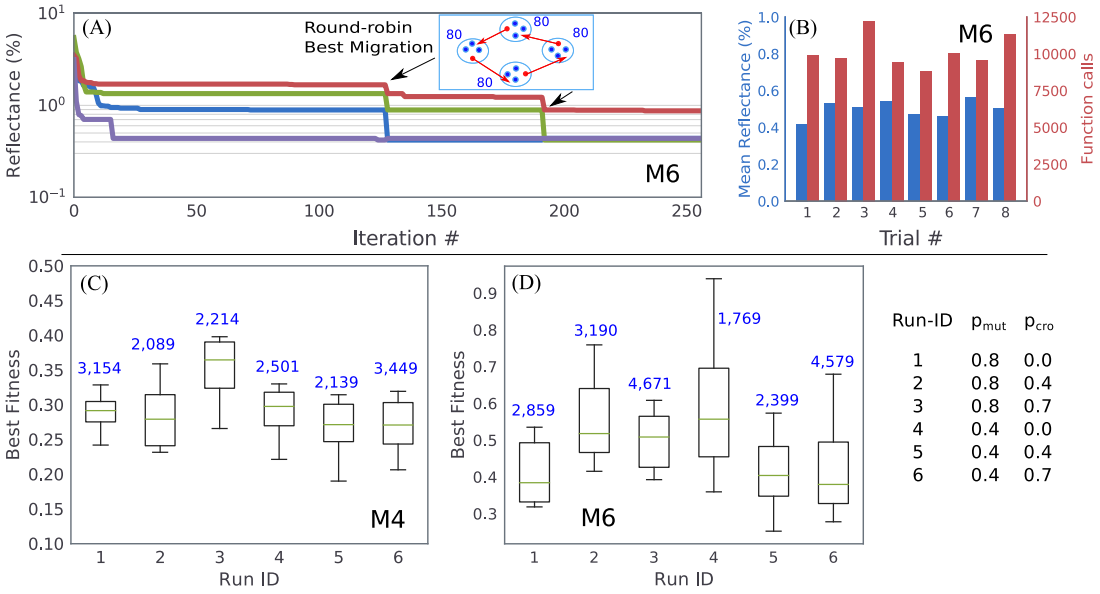

Fig. 8. DNN-Assisted DE Performance (using the M6 model) for a 4-island run with 80 individuals each with two round-robin best individual migration steps. A fitness evolution as a function of algorithm iteration number. B summary of eight runs of the DE showing the function calls and best fitness achieved in each run. C and D compare the performance when the mutation radius and the crossover probability are changed for the assisted-DE (M4 and M6) models for a 10-island 80 population run without intermigration. Blue text indicates the number of function calls (per island).

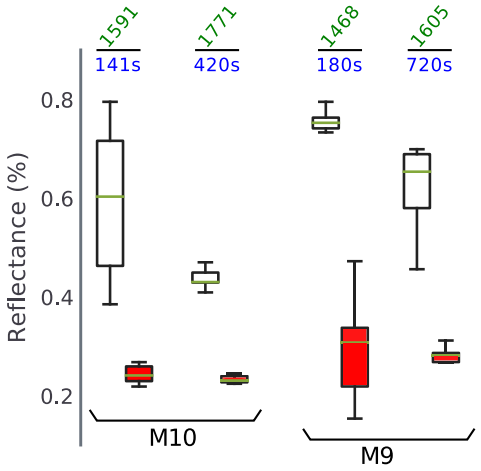

Fig. 9. Comparison of the performance of a model trained with an actively sampled dataset M10 (size 100k) with another model M9 trained on the same sized dataset. The white boxes are for DNN acting alone generating initializations and the red boxes are assisted-DE using these initializations. The green text is the number of function calls and the blue is the initialization time. All runs took about 16 minutes in the assisted-DE phase.

was called M10. For comparison purposes, another 128 x 3 DNN, trained on a passively sampled 100k dataset was created (called M9). Note that M9 and M10 require the same number of function calls for dataset generation. M10 takes additional time for the first round of training. Figure 9 shows the performance comparison of M9 and M10. In standalone mode, M10 seems to provide improved designs faster in comparison with M9 for comparable search times. However, in assisted-DE mode, M10 performs much better than M9 and even beats M4 and vanilla DE. The variance observed is very small.

IV. Conclusion

We have presented promising results that are obtained by pairing DNNs with evolutionary algorithms. It is anticipated that other kinds of evolutionary algorithms like genetic algorithms may also work well. Even with DE, many variants exist which could be explored to provide further performance gains. In terms of active sampling, even the single iteration reported here was found to be beneficial. It is likely that multiple stages may even make DNNs attractive for one-off optimizations. A fruitful extension is to study if these approaches can be extended to the inverse design of two and three dimensional designs. In summary, evolutionary Optimization is ubiquitous in photonics design. Our report shows that it is possible to bridge this with recent work in DNN based inverse design.

References

[1] Richard Ernest Bellman, *Dynamic Programming*. New York, New York, USA: Dover Publications Inc, 2003.

[2] J. Langhabel, J. Wolff, and Raphael Holca-Lamarre, "Learning to optimise: Using Bayesian deep learning for transfer learning in optimisation," in *Proc. Workshop Bayesian Deep Learn.*, Barcelona, Spain, 2016, pp. 1–3.

[3] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[4] D. Liu, Y. Tan, E. Khoram, and Z. Yu, "Training deep neural networks for the inverse design of nanophotonic structures," *ACS Photonics*, vol. 5, pp. 1365–1369, 2018.

[5] Z. Liu, D. Zhu, S. P. Rodrigues, K.-T. Lee, and W. Cai, "Generative model for the inverse design of metasurfaces," *Nano Lett.*, vol. 18, pp. 6570–6576, 2018.

[6] T. Zhang *et al.*, "Spectrum prediction and inverse design for plasmonic waveguide system based on artificial neural networks," *Arxiv Phys. Opt., 2018, arXiv:1805*, pp. 1–7.

[7] I. Malkiel *et al.*, "Plasmonic nanostructure design and characterization via Deep Learning," *Light: Sci. Appl.*, vol. 7, no. 60, pp. 1–8, 2018. [Online]. Available: http://dx.doi.org/10.1038/s41377-018-0060-7

[8] P. R. Wiecha, N. Mallet, and G. Larrieu, "Pushing the limits of optical information storage using deep learning," *Nature Nanotechnology*, vol. 14, pp. 237–244, 2019.
[9] K. Yao, R. Unni, and Y. Zheng, "Intelligent nanophotonics: Merging photonics and artificial intelligence at the nanoscale," *Nanophotonics*, vol. 8, no. 3, pp. 1–28, 2019.
[10] W. Ma, F. Cheng, and Y. Liu, "Deep-learning-enabled on-demand design of chiral metamaterials," *ACS Nano*, vol. 12, pp. 6326–6334, 2018.
[11] S. Inampudi and H. Mosallaei, "Neural network based design of metagratings," *Appl. Phys. Lett.*, vol. 112, pp. 1–5, 2018, Art. no. 241102.
[12] M. H. Tahersima *et al.*, "Deep neural network inverse design of integrated nanophotonic devices," 2018, *arXiv:1809.0355*, pp. 1–8.
[13] A. Gandhi and C. E. Png, "Modal classification in optical waveguides using deep learning," *J. Modern Opt.*, vol. 66, no. 5, pp. 557–561, 2019.
[14] J. Peurifoy *et al.*, "Nanophotonic particle simulation and inverse design using artificial neural networks," *Sci. Adv.*, vol. 4, pp. 1–8, 2018, Art. no. eaar4206.
[15] R. Schwartz-ziv and N. Tishby, "Opening the black box of deep neural networks via information," 2017, *arXiv:1703.00810*, pp. 1–19.
[16] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation, University of Cambridge, Cambridge, U.K., 2016.
[17] U. B. Schallenberg, "Antireflection design concepts with equivalent layers," *Appl. Energy*, vol. 45, no. 7, pp. 1507–1514, 2006.
[18] A. V. Tikhonravov and J. A. Dobrowolski, "Quasi-optimal synthesis for antireflection coatings: a new method," *Appl. Opt.*, vol. 32, no. 22, pp. 4265–4275, 1993.
[19] J. A. Dobrowolski, A. V. Tikhonravov, M. K. Trubetskov, B. T. Sullivan, and P. G. Verly, "Optimal single-band normal-incidence antireflection coatings," *Appl. Opt.*, vol. 35, no. 4, pp. 644–658, 1996.
[20] A. V. Tikhonravov, "Some theoretical aspects of thin-film optics and their applications," *Appl. Opt.*, vol. 32, no. 28, pp. 5417–5426, 1993.
[21] S. W. Anzengruber, E. Klann, R. Ramlau, and D. Tonova, "Numerical methods for the design of gradient-index optical coatings," *Appl. Opt.*, vol. 51, no. 34, pp. 8277–8295, 2012.
[22] Y. Zhao, F. Chen, Q. Shen, and L. Zhang, "Optimal design of graded refractive index profile for broadband omnidirectional antireflection coatings using genetic programming," *Progr. Electromagnetics Res.*, vol. 145, pp. 39–48, 2014.
[23] J.-M. Yang and C.-Y. Kao, "Efficient evolutionary algorithm for the thin-film synthesis of inhomogeneous optical coatings," *Appl. Opt.*, vol. 40, no. 19, pp. 3256–3267, 2001.
[24] M. Ebrahimi and M. Ghasemi, "Design and optimization of thin film polarizer at the wavelength of 1540 nm using differential evolution algorithm," *Opt. Quantum Electron.*, vol. 50, no. 4, pp. 1–9, 2018. [Online]. Available: https://doi.org/10.1007/s11082-018-1453-9
[25] V. Janicki, J. Sancho-parramon, and H. Zorc, "Refractive index profile modelling of dielectric inhomogeneous coatings using effective medium theories," *Thin Solid Films*, vol. 516, pp. 3368–3373, 2008.
[26] H. Becker *et al.*, "Design and realization of advanced multi-index systems," *Appl. Opt.*, vol. 53, no. 4, pp. A88–A95, 2014.
[27] R. Storn and K. Price, "Differential evolution a simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, pp. 341–359, 1997.
[28] T. Peter, "Using deep learning as a surrogate model in multi-objective evolutionary algorithms," Ph.D. dissertation, Otto-von-Guericke-Universität, Magdeburg, Germany, 2018.
[29] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Comput.*, vol. 9, pp. 3–12, 2005.
[30] S. J. Byrnes, "Multilayer optical calculations," 2018, *arXiv:1603.02720*, pp. 1–20.
[31] F. Chollet, "Keras: The python deep learning library," *Github Repository*, no. https://github.com/fchollet/keras, 2015.
[32] T. Chen *et al.*, "MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems," 2015, *arXiv:1512.01274*, pp. 1–6.
[33] R. S. Hegde, "DeepMie – Deep learning electromagnetic scattering," *Bitbucket Repository*, no. https://bitbucket.org/rshegde/deepmie-deep-learning-electromagnetic-scatteri ng/overview, 2018.
[34] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.0594*, pp. 1–13.
[35] I. H. Malitson, "Interspecimen comparison of the refractive index of fused silica," *J. Opt. Soc. Amer.*, vol. 55, no. 10, pp. 1205–1209, 1965.
[36] T. Siefke *et al.*, "Materials pushing the application limits of wire grid polarizers further into the deep ultraviolet spectral range," *Adv. Opt. Mater.*, vol. 4, pp. 1780–1786, Jul. 2016.

Ravi S. Hegde received the Bachelor of Engineering degree in electrical engineering from the National Institute of Technology, Mangalore, India. He received the Master of Science in electrical engineering (specialization in photonics technology) from the University of Southern California, Los Angeles, CA, USA and the Doctor of Philosophy in Electrical Engineering from the University of Michigan at Ann Arbor, MI, USA. He was a Research Scientist with the Electronics and Photonics division at the A*STAR Institute of High-Performance Computing in Singapore, since 2009. He is an Assistant Professor with the Department of Electrical Engineering, Indian Institute of Technology, Gandhinagar, India, since 2015. He currently works on analytical and numerical modeling of nanoscale optical structures and devices and their application toward energy harvesting, sensing, and imaging.