Research Article Vol. 4, No. 1 / 15 January 2021 / OSA Continuum 87

Successive training of a generative adversarial network for the design of an optical cloak

ANDRE-PIERRE BLANCHARD-DIONNE* AND OLIVIER J. F. MARTIN

MicroTechnique Institute, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland
andre-pierre.blanchard-dionne@epfl.ch

Abstract: At the nanoscale level, optical properties of materials depend greatly on their shape. Finding the right geometry for a specific property remains a fastidious and long task, even with the help of modelling tools. In this work, we overcome this challenge by using artificial intelligence to guide a reverse engineering method. We present an optimization algorithm based on a deep convolution generative adversarial network for the design a 2-dimensional optical cloak. The optical cloak consists in a shell of uniform and isotropical dielectric material, and the cloaking is achieved via the geometry of this shell. We use a feedback loop from the solutions of this generative network to successively retrain it and improve its ability to predict and find optimal geometries. This generative method allows to find a global solution to the optimization problem without any prior knowledge of good cloaking geometries.

1. Introduction

The theoretical idea of an optical cloak was first suggested in 2006 by JF Pendry in a paper of transformation optics [1]. It introduced the idea that an electromagnetic wavefront could be bent inside a shell of carefully chosen dielectric and magnetic materials and remain unchanged upon its exit, thus concealing an object inside of it. Cloaking of objects was later demonstrated experimentally [2] using metamaterials at microwave frequencies. The proposed material of the shell consisted of a series of split-ring resonators of different dimensions to achieve a non-uniform anisotropic dielectric permittivity and magnetic permeability. Recently, similar cloaking behaviour was found to exist in shell designs of more simple all-dielectric isotropical and uniform material [3–6]. The working principle for those shells resides in their geometry which can lead to the bending of the wavefront around the object to hide. The optimal geometries in those reported cases were found using topology optimization methods.

In this paper we use a deep learning generative algorithm to accomplish the optimization of the geometry of an all-dielectric isotropic shell for cloaking. Generative networks have recently been demonstrated as useful tools for finding global solutions to inverse engineering problems [7–9] and have found applications in nanophotonics for the design of metasurfaces [10–14], nanostructures [15,16], metagratings [17], thermal emitters [18], photonic crystals [19] and power splitters [20]. We suggest to use a generative adversarial network [21] in a feedback loop to find an optimal configuration for cloaking. The procedure is done in a few steps: First, several shell geometries are randomly created and are simulated to obtain their scattering coefficient (see Fig. 1(a)). We then train a forward network (FN) to predict the scattered fields amplitudes from each geometry. A deep convolutional generative adversarial network (DCGAN) coupled to the FN is then used to generate new solutions which are aimed at minimizing the scattered fields. We then implement the feedback loop to reuse the generated solutions to improve both the FN and the DCGAN for sufficient iterations until a satisfactory solution is found or the generative algorithm doesn't improve anymore.

#413394 https://doi.org/10.1364/OSAC.413394
Journal © 2021 Received 27 Oct 2020; accepted 5 Dec 2020; published 24 Dec 2020

OSA CONTINUUM

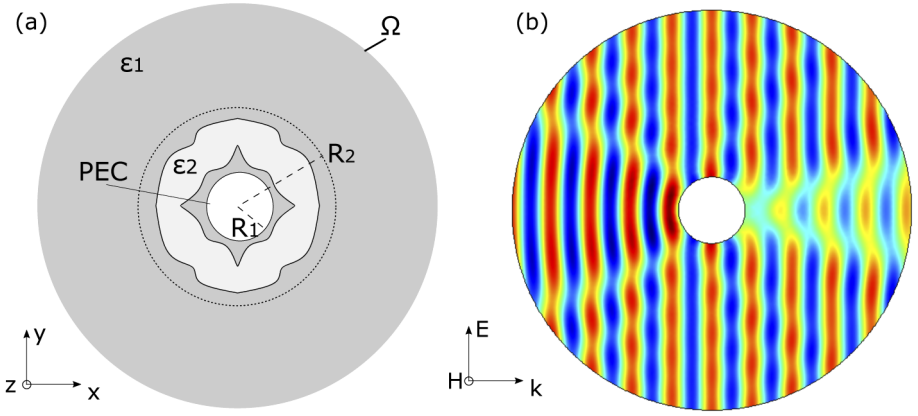

Fig. 1. (a) Schematic of the simulation domain. The object at r=R_1 is represented by a perfect electrical conductor (PEC). The shell consists of a medium of permittivity $\epsilon_2 = 2$ inside the region $R_1 < r < R_2$. The scattered fields are calculated at the boundary of the domain Ω (b) Total transverse fields H_z when no cloaks are included in the design.

2. Methods

2.1. FEM simulations

In order to obtain the scattered field's amplitude from a given cloak geometry, finite-element simulations using COMSOL's RF module are used. The simulation domain is presented in Fig. 1 and has an exterior circular boundary with a radius of 12 μm. Another circular boundary of perfect electrical conductor (PEC) is located at R_1 =1 μm and constitutes the object to conceal. Multiple polygons of dielectric constant $\varepsilon_2 = 2$ are included in the region $R_1 < r < R_2$ to form the cloaking shell. A background electromagnetic field defined by $\vec{E} = E_0 e^{ik_0 x}\hat{y}$ is used and the scattered fields are solved with scattering boundary conditions at Ω. We then integrate the relative Poynting vector at the exterior boundary of the simulation to obtain the metric to minimize :

$$\Psi = \frac{1}{2}\int_{\Omega} \overrightarrow{E}_{rel} \times \overrightarrow{H}^{*}_{rel}\, dC, \tag{1}$$

where $\overrightarrow{E}_{rel} = \overrightarrow{E}_{total} - \overrightarrow{E}_{background}$.

We use an object of radius R_1=1 μm and a shell of R_2=3 μm for the simulations, and the wavelength is set at $\lambda = R_2/2.5 = 1.2$ μm. A total of 13000 simulations were performed with randomly generated shell geometries to form the initial dataset. The geometries were generated using the union of randomly generated curves inside the defined shell radius. We use a shell which is symmetrical in x- and y- directions according to the symmetry of the problem and also to assure continuity of the shell around the object. Data is available in the figshare Ref. [22].

2.2. Neural networks

The first part of the model consists in a forward predictive model of the scattering coefficient Ψ as a function of the shell geometry. The input image consists in 64×64 binary images where the region of dielectric constant ϵ_2 is represented by 1s and ϵ_1 by 0s. Since the shells are symmetrical in -x and -y, the images are taken for only one quadrant of the shell. Given the 1.5 μm span of this region, that leads to a pixel of dielectric material measuring about 23 nm. A convolution network is then used, which consists in 4 convolution block layers and 2 dense layers that takes a (64×64) image data and creates a single digit output. The dataset was divided into a training set, a validation set and a test set using a ratio of 70 – 15 – 15%. The network is trained with an

OSA CONTINUUM

Adam optimizer [23] with a learning rate of $1 \cdot 10^{-4}$ and the mean squared error is used as the loss function.

Each convolution block consists in a convolution layer (Conv2D), a batch normalization layer (BatchNormalization) and a leaky rectifier linear unit (LeakyReLU) activation function. The convolution operation optimizes a number of filters to be convolved with the inputs, whether it is the input image or the subsequent output of each layer. The number of filters of the four layers are respectively [30,32,24,18] and their size are [20,30,40,50]. Both the dense layers have 75 units and use LeakyReLU activation function and L2 regularization. All hyperparameters were optimized using a grid search methodology. This network is represented in Fig. 2 and is available in the Refs. [24].

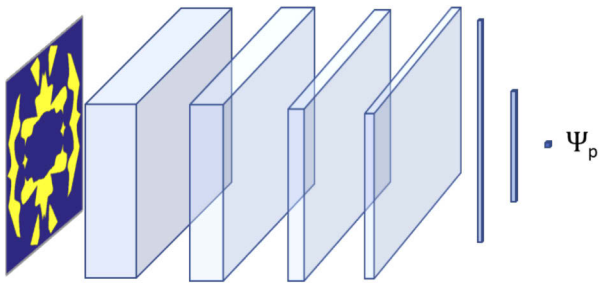

Fig. 2. Forward predictive network. The input is the image of a cloaking shell configuration. The network consists in 4 convolution layers and 2 dense layers, and the output is the scattering coefficient Ψ

The generative adversarial network consists of two different networks, a generator and a discriminator, which work in opposition to create new shell geometries similar to those of the dataset. The generator is a transposed convolution network, which takes a random noise vector of dimension 200 as input and creates a new "fake" image of a cloaking shell as output. The discriminator is a convolution network and takes an image and gives it a probability that it is "real", meaning that it came from the initial datatset.

The generator consists in a series of 3 transposed convolution layers (Conv2DTranspose), which patches the input with zeros in between each of its row and columns, then applies convolution operations with a series of filters. This operation thus upsamples its input by a factor of a certain ratio, which is 2 for our case. This layer is followed by BatchNormalization and LeakyReLU activation layers. The number of filters for these layers are [64,32,1] of size [5,5,5]. The discriminator is a convolution network with 3 block layers of convolutions, LeakyReLU and dropout layers (DropOut) of coefficient 0.3. The number of filters is [32,32,16], and the size of the filters are [5,5,5]. All hyperparameters were optimized using a grid search methodology. Both networks are represented in Fig. 3(a) and their details can be found in the online Ref. [24].

The adversarial play between the two networks is accomplished via their loss function, which is computed using the binary cross-entropy function :

$$L(I) = -\frac{1}{N}\sum_{i}^{N} y_i log(p(y_i)) + (1 - y_i) log(1 - p(y_i)), \tag{2}$$

where I is the dataset of images, N is the number of images in the dataset, y_i is the label of the image, and $p(y_i)$ is the probability output. For every batch of the image dataset, a batch of the same size of fake images is created using the generator. All those images are given to the discriminator, who gives them a probability $p(y_i)$ of being real. The loss is then calculated using Eq. (2), with the label value y_i equal to 1 for the real images and 0 for the fake images in the case of the discriminator loss, and 1 for the fake images in the case of the generator loss. This

OSA CONTINUUM

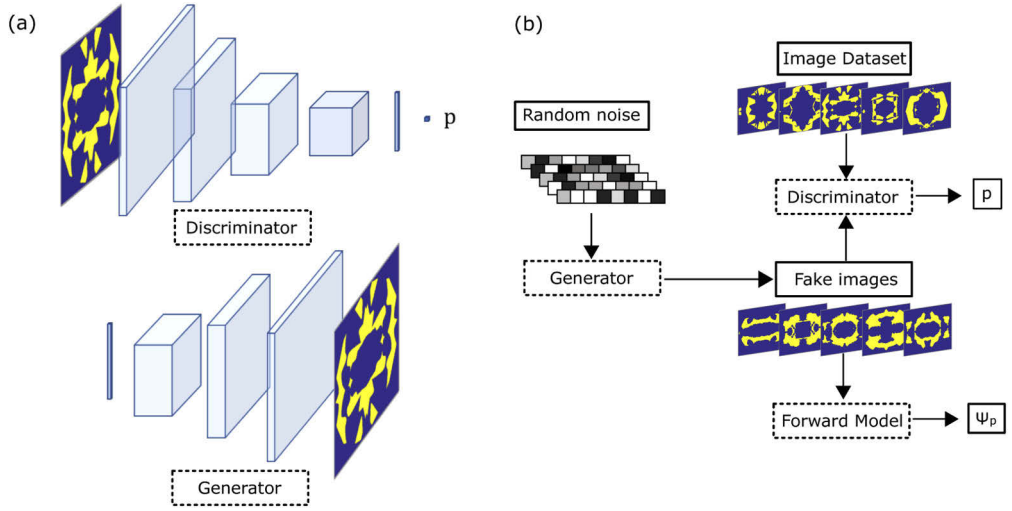

Fig. 3. (a) Discriminator and Generator networks. The Discriminator consists of 4 convolutions layers and one dense layer, with the output being the probability of the image being "real" (from the dataset) or "false" (from the generator). The Generator consists of 3 Transposed convolution layers and the output is a binary image of a shell configuration. (b) Block diagram of the DCGAN. The neural networks are represented with dashed lines, while data are represented with full lines. Output value $p(y)$ represents the probability of an image being tagged as real, while output Ψ represents the scattering metric of the shell represented by the image.

way, the discriminator is trained to differentiate the real and fake images, while the generator is trained to fool the discriminator by feeding it more and more realistic images which look like those of the dataset.

To this branched network we add an additional path, which will calculate the scattering metric of the new fake images using the forward model. This way, we can add this value to the loss function of the generator so the new images not only aim to mimick the ones of the dataset but also to optimize cloaking. Schematic of the full DCGAN network is represented in Fig. 3(b). The total loss function of the generator is thus given by :

$$L_t = \alpha_g L_g + \alpha_f L_f, \tag{3}$$

where L_t is the total loss, L_g is the generator loss, L_f is the forward model loss and α_g, α_f are weighting coefficients. Care must be taken in order to choose weighting coefficients that balance cloaking optimization and generator efficiency.

One important detail of the previous network is the implementation of a rounding function on the fake images obtained from the DCGAN in order to have binary image input for the forward model. Since a rounding function's derivative is equal to zero, its direct use will lead to a vanishing gradient and the forward model wouldn't contribute to the training of the generator. We thus use a parameterized sigmoid as an approximation of the round function with a continuous derivative given by:

$$\sigma = \frac{1}{1 + e^{-b(x-a)}} \qquad d\sigma/dx = b\sigma(1-\sigma) \qquad . \tag{4}$$

Taking $a = 0.5$ and $b = 10$, this will give a good approximation of the rounding function for values which are included in between [0,1]. Parameter b is a factor to control the slope of the sigmoid.

OSA CONTINUUM

2.3. Feedback loop

DCGAN networks are effective at quickly generating and evaluating potential configurations for the cloacking shell. The addition of the forward network guides this generative process towards the optimization goal, in this case to minimize scattering. This method is limited though since the forward model isn't perfect at predicting the output to minimize, especially for configurations that differ considerably from those of the dataset. Some solutions might thus not be as optimal as the forward model predicts, and the model might also pass on a good configuration by wrongly predicting its output.

For this reason, an iterative process of retraining the forward network is suggested for improving the solution search. The method is depicted in Fig. 4. The best solutions from the DCGAN according to the predicted scattering coefficient Ψ_p are taken and simulated using FEM in order to obtain the real scattering coefficient Ψ_r. Those solutions are then added to the dataset and the forward model is retrained. We reinitialized the weights of the forward model everytime in order to avoid validation biases incoming from training examples reshuffled in the validation set. The new forward model is added in the DCGAN and a new solutions search is initiated, using the new and improved dataset. The generator is also reinitialized to avoid an early convergence towards a local minimum.

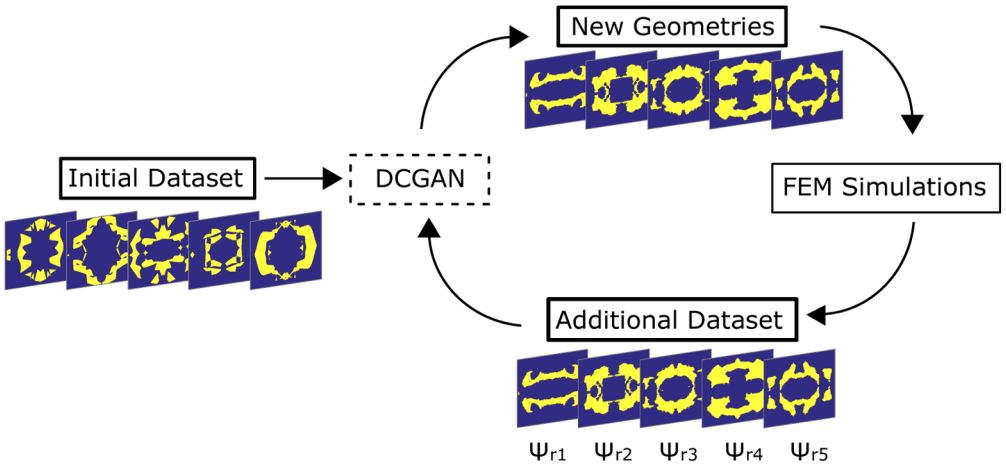

Fig. 4. Feedback loop training of the DCGAN. After each training of the DCGAN for 60 epochs, the 1000 best configurations are taken and simulated using FEM. The new ground truth data is used to retrain the forward model and the DCGAN is retrained with the new dataset.

This way, the forward model is improved by correcting any wrong prediction attributed to specific configurations which were deemed optimal. Furthermore, the dataset is improved by including good configurations, which will in turn lead the generator towards suggesting more favorable geometries.

3. Results and discussion

3.1. Training of the DCGAN

Training of the DCGAN usually requires a fine tuning of the parameters for each of the neural networks, whether it is the generator, the discriminator or the forward model. Both generator and discriminator are involved in a zero-sum game one with the other, since positive outcome for one means negative outcome for the other. They usually reach a Nash equilibrium and oscillate out of phase [25]. The generator and discriminator need not to overpower one another, or else the

OSA CONTINUUM

generated configuration might be random and noisy, and as they differ considerably from those of the dataset the performance of the forward model will decrease significantly.

Adding an additional loss for the generator with the forward model creates a perturbation of this equilibrium, since the generator not only updates its weights to fool the discriminator but also to minimize the scattering metric Ψ. This leads, as can be seen in Fig. 5(a), to a generator reaching an equilibrium slightly above that of the discriminator, meaning that the generator might not perform fully at suggesting configuration resembling those of the dataset. Choosing the weighting parameters α_f and α_g of Eq. (3) carefully as to keep the difference between generator and discriminator loss as small as possible is thus crucial for obtaining adequate solutions. The coefficient α_g was set to 1 and the parameter $\alpha_f = 5/<\Psi>$ for each iteration of the method in order to adapt for lower loss with improved dataset.

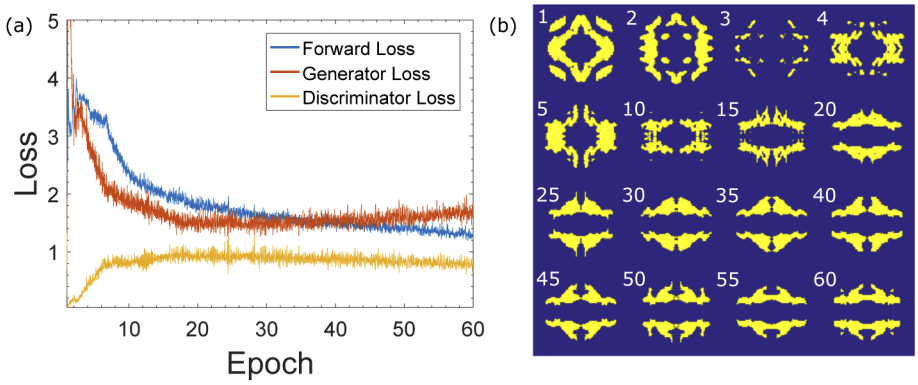

Fig. 5. (a) Forward, Generator and Discriminator Loss during training of the DCGAN. (b) Evolution of the generated images of the DCGAN for a specific noise vector during the training of the 5^{th} iteration of the feedback loop.

Since the dataset doesn't contain similar type of images, the outputs of the generator remains dynamic and keeps evolving during the training even once the equilibrium between discriminator and generator has been reached. Figure 5(b) plots an exemple of the output image for a specific noise vector during certain training epochs of the DCGAN (during the 5^{th} iteration of the feedback loop). Even if the image converges towards a certain shape at epoch 20, little variations are still created for subsequent epochs. For this reason, a solution search is made after each epoch to maximize the number of different potential configurations.

3.2. Feedback loop

In order to achieve optimization of the cloaking shell, 11 training iterations of the DCGAN were accomplished. In Fig. 6, 4 random examples of the generator output are presented for each iteration of this feedback loop of the DCGAN. The proposed configurations are very diverse for the first few iterations, but converge towards one optimal configuration starting at iteration 5. This convergence is caused by two effects: on one hand, the forward model forces the generator to suggest optimal configurations, and on the other hand, the dataset is slowly getting more and more populated by those optimal configurations. In order to avoid getting the algorithm trapped in a local minimum, the α_f parameter of Eq. (3) needs to be chosen low enough as to not collapse all the solutions of the generator towards the same configuration.

The strong point of the DCGAN is its ability to generate adequate configuration and to quickly predict the output of a certain configuration. The forward model takes about 0.1 ms for one prediction working on a Kaggle 17 GPU RAM kernel, while one COMSOL simulation takes

OSA CONTINUUM

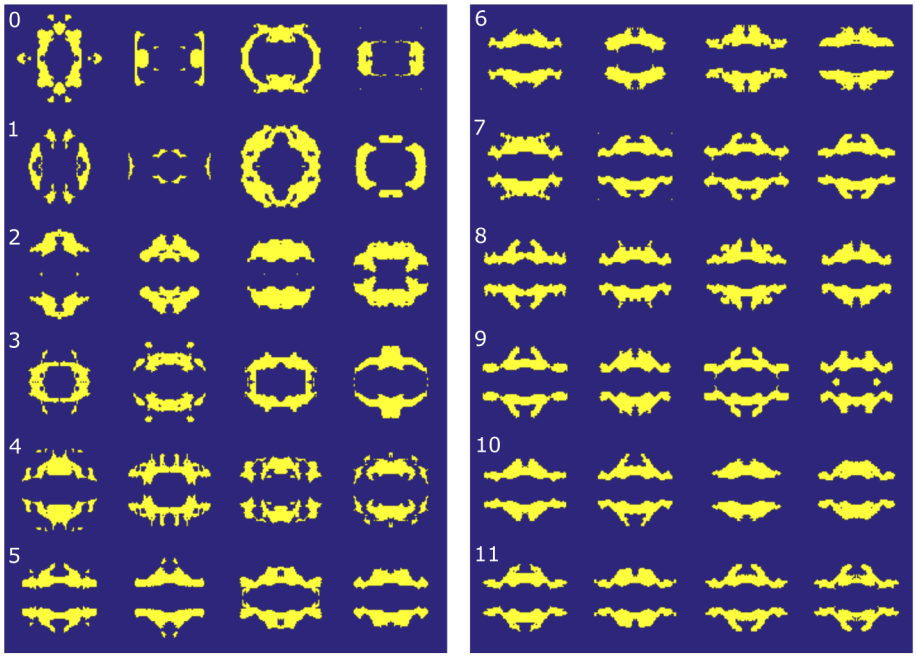

Fig. 6. Examples of 4 generated images at the end of the 60 epochs training of the DCGAN for each iteration of the forward loop.

5-10s, making it roughly 50 000 - 100 000 times faster. We thus take advantage of this feature by testing 128000 configurations at each of the 60 epoch of the training of the DCGAN.

After each training of the DCGAN the 1000 best configurations are taken according to their predicted value of Ψ_p and are simulated in the FEM method in order to obtain their real value Ψ_r. The loop is continued until no more improvements in the Ψ_r is observed. The minimal value of Ψ_r and the average value for each of the 11 iterations are plotted in Fig. 7(a). Both values are normalized with the value of the scattering coefficient of the object with no cloak.

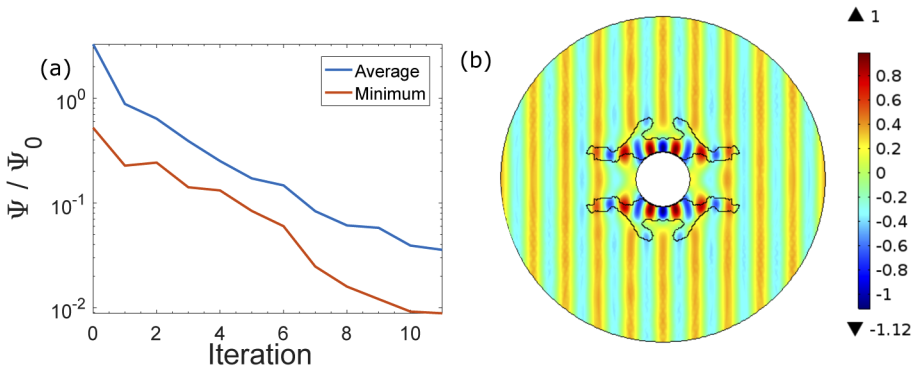

Fig. 7. Average value (blue) and Minimum value (red) of Ψ_r for the 1000 best configurations found by the DCGAN at every iteration. This value was normalized with the scattering coefficient of the object without a cloak. The retraining of the DCGAN was done for 11 iterations until the minimal value of Ψ_r no longer improved. (b) Normalized value of the trasverse H_z field with the optimal configuration.

OSA CONTINUUM

We can observe a continual improvement of the minimal value and the average value of Ψ until it reaches 5.11×10^{-11} W/m. This leads to a cloaking ratio of 0.0089 when comparing to the value 5.77×10^{-9} W/m without the cloak. This performance is comparable to solutions found using topology optimization [3–5] for similar dimensions of the shell and object compared to the wavelength. Restarting the feedback loop from the beginning gave a similar type of geometry, meaning that the algorithm converges towards an optimized solution. Also, the average value of the normalized scattering coefficients at iteration 0 is greater than one, meaning that most of the random design of the initial set are detrimental to cloaking. This demonstrates the ability of the method to reach a global minimum, with no prior initial assumptions on the shape of an optimal cloak.

In Fig. 7(b) the total transverse magnetic field H_z is plotted with the optimal cloak configuration found, which shows almost no distorsion of the wavefront compared to Fig. 1(b). A strong concentration of the fields near the top and bottom regions of the object is observed, which is typical of the bending of the electromagnetic wavefront for cloaking shells.

4. Conclusion

In this paper we have demonstrated the use of deep learning for the optimization of an optical cloak. The suggested algorithm consists in a DCGAN architecture which is trained multiple times in a feedback loop in order to improve the solution search. The total scattered field coefficient, calculated with the Poynting vector at the outside boundary of the simulation domain of a FEM simulation, reached a ratio of 0.0089 of the field scattered without a cloak, which is comparable to results obtained using topology optimization. Since it started from completely random data, this algorithm thus represent an efficient optimization method which can guide the solution towards a global minimum, with no initial assumptions.

Funding. European Research Council (ERC-2015-AdG-695206 Nanofactory); Fonds de recherche du Québec – Nature et technologies (B3 259466).

Disclosures. The authors declare no conflicts of interest.

References

1. J. B. Pendry, D. Schurig, and D. R. Smith, "Controlling electromagnetic fields," Science **312**(5781), 1780–1782 (2006).
2. D. Schurig, J. Mock, B. Justice, S. A. Cummer, J. B. Pendry, A. Starr, and D. R. Smith, "Metamaterial electromagnetic cloak at microwave frequencies," Science **314**(5801), 977–980 (2006).
3. J. Andkjær, N. Asger Mortensen, and O. Sigmund, "Towards all-dielectric, polarization-independent optical cloaks," Appl. Phys. Lett. **100**(10), 101106 (2012).
4. J. Andkjær and O. Sigmund, "Topology optimized low-contrast all-dielectric optical cloak," Appl. Phys. Lett. **98**(2), 021112 (2011).
5. G. Fujii, H. Watanabe, T. Yamada, T. Ueta, and M. Mizuno, "Level set based topology optimization for optical cloaks," Appl. Phys. Lett. **102**(25), 251106 (2013).
6. E. Bor, C. Babayigit, H. Kurt, K. Staliunas, and M. Turduev, "Directional invisibility by genetic optimization," Opt. Lett. **43**(23), 5781–5784 (2018).
7. B. Sanchez-Lengeling and A. Aspuru-Guzik, "Inverse molecular design using machine learning: Generative models for matter engineering," Science **361**(6400), 360–365 (2018).
8. S. So, T. Badloe, J. Noh, J. Bravo-Abad, and J. Rho, "Deep learning enabled inverse design in nanophotonics," Nanophotonics **9**(5), 1041–1057 (2020).
9. K. Yao, R. Unni, and Y. Zheng, "Intelligent nanophotonics: merging photonics and artificial intelligence at the nanoscale," Nanophotonics **8**(3), 339–366 (2019).
10. Z. Liu, D. Zhu, S. P. Rodrigues, K.-T. Lee, and W. Cai, "Generative model for the inverse design of metasurfaces," Nano Lett. **18**(10), 6570–6576 (2018).
11. J. Jiang and J. A. Fan, "Simulator-based training of generative neural networks for the inverse design of metasurfaces," Nanophotonics **9**(5), 1059–1069 (2019).
12. S. An, B. Zheng, M. Y. Shalaginov, H. Tang, H. Li, L. Zhou, J. Ding, A. M. Agarwal, C. Rivero-Baleine, M. Kang, K. A. Richardson, T. Gu, J. Hu, C. Fowler, and H. Zhang, "A freeform dielectric metasurface modeling approach based on deep neural networks," arXiv preprint arXiv:2001.00121 (2020).
13. J. A. Hodge, K. V. Mishra, and A. I. Zaghloul, "Rf metasurface array design using deep convolutional generative adversarial networks," in *IEEE International Symposium on Phased Array Systems and Technology*, (2019).

OSA CONTINUUM

14. W. Ma, F. Cheng, Y. Xu, Q. Wen, and Y. Liu, "Probabilistic representation and inverse design of metamaterials based on a deep generative model with semi-supervised learning strategy," Adv. Mater. **31**(35), 1901111 (2019).
15. S. So and J. Rho, "Designing nanophotonic structures using conditional deep convolutional generative adversarial networks," Nanophotonics **8**(7), 1255–1261 (2019).
16. Y. Kiarashinejad, S. Abdollahramezani, and A. Adibi, "Deep learning approach based on dimensionality reduction for designing electromagnetic nanostructures," npj Comput. Mater. **6**(1), 12 (2020).
17. J. Jiang, D. Sell, S. Hoyer, J. Hickey, J. Yang, and J. A. Fan, "Free-form diffractive metagrating design based on generative adversarial networks," ACS Nano **13**(8), 8872–8878 (2019).
18. Z. A. Kudyshev, A. V. Kildishev, V. M. Shalaev, and A. Boltasseva, "Machine-learning-assisted metasurface design for high-efficiency thermal emitter optimization," Appl. Phys. Rev. **7**(2), 021407 (2020).
19. T. Asano and S. Noda, "Iterative optimization of photonic crystal nanocavity designs by using deep neural networks," Nanophotonics **8**(12), 2243–2256 (2019).
20. Y. Tang, K. Kojima, T. Koike-Akino, Y. Wang, P. Wu, M. Tahersima, D. Jha, K. Parsons, and M. Qi, "Generative deep learning model for a multi-level nano-optic broadband power splitter," in *Optical Fiber Communication Conference*, (Optical Society of America, 2020), pp. Th1A–1.
21. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, (2014), pp. 2672–2680.
22. A.-P. Blanchard-Dionne, "Neural Network Codes," figshare (2020). https://doi.org/10.6084/m9.figshare.13135802.v1.
23. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980 (2014).
24. A.-P. Blanchard-Dionne, "Shell geometry data," figshare (2020). https://doi.org/10.6084/m9.figshare.13135802.v1.
25. I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," arXiv preprint arXiv:1701.00160 (2016).