SCIENTIFIC REPORTS

OPEN

Plasmonic colours predicted by deep learning

Joshua Baxter[1,2], Antonino Calà Lesina[1,2,3], Jean-Michel Guay[1,2,4], Arnaud Weck[1,2,4], Pierre Berini[1,2,3] & Lora Ramunno[1,2]

Received: 12 March 2019
Accepted: 17 May 2019
Published online: 30 May 2019

Picosecond laser pulses have been used as a surface colouring technique for noble metals, where the colours result from plasmonic resonances in the metallic nanoparticles created and redeposited on the surface by ablation and deposition processes. This technology provides two datasets which we use to train artificial neural networks, data from the experiment itself (laser parameters vs. colours) and data from the corresponding numerical simulations (geometric parameters vs. colours). We apply deep learning to predict the colour in both cases. We also propose a method for the solution of the inverse problem – wherein the geometric parameters and the laser parameters are predicted from colour – using an iterative multivariable inverse design method.

Deep learning (DL) refers to a sub-field of Machine Learning and Artificial Intelligence where multi-layered artificial neural networks, or deep neural networks (DNNs), are used for high accuracy predictions and classifications. Deep learning has been used in speech recognition, image classification, vision, and pattern extraction, to name a few examples[1]. Its application in photonics is recent. For example, DL has been used to classify scanning electron microscope (SEM) images of nanostructures[2], to increase the resolution of microscopy[3], and for the design of nanophotonic structures[4–10]. DL models can be used to find patterns in complicated datasets and are particularly useful when large pre-calculated or pre-measured datasets are available for training. Both simulations and experiments in nanophotonics can provide the large datasets required for DL. For example, geometric parameter sweeps are used in plasmonics to optimize the design of nanostructures, while parameters are tuned in experiments to produce the best nanofabrication results. The exploration of the parameter space based on trials is time-consuming. Using data from parameter sweeps, one can train a DL model to predict the experimental or simulation outcome for a new set of input parameters, thus avoiding the need to run many trials of the actual experiment or computation. Further, in an industrial setting where the environment is not well controlled, environmental parameters can be tracked and can become additional inputs of a deep learning application.

An application which may benefit from DL is the plasmonic colouration of metal surfaces via laser machining. By controlling different laser parameters such as scanning speed, hatch spacing, burst number, etc., laser-machined surfaces (Fig. 1(a)) with a large range of colours (Fig. 1(b)) were created[11,12]. Finite-difference time-domain (FDTD) simulations were performed based on the geometries inspired by the experimental SEM images and the measured nanoparticle distributions, and these simulations were able to explain experimental trends. This includes finding that the colour transition from blue to yellow occurs when the inter-particle spacing for nanoparticles of radius $R = 35$ nm is progressively increased. The nanoparticle distribution is modelled as a periodic hexagonal lattice of single-sized nanoparticles embedded into a metal surface (Fig. 1(c)), and the output of the simulation is a reflection spectrum from which the colour is rendered to the Red-Green-Blue (RGB) colour scale[13] (Fig. 1(d)). Both experiments and simulations contain challenges which can be addressed by DL. On the experimental side, the colours generated by a new set of laser parameters must be determined empirically through trial and error. On the computational side, the requirement for accuracy means that a large-scale FDTD simulation has to be run in order to find the colour corresponding to a new set of geometric parameters, and this is time-consuming.

In this paper we apply DL to the direct prediction of new colours using both the experimental and simulation data. We predict colours from laser parameters by leveraging the dataset linking laser parameters to real, physical colours. We also predict colours from nanoparticle geometries by using the dataset linking the simulation

[1]University of Ottawa, Department of Physics, Ottawa, K1N 6N5, Canada. [2]University of Ottawa, Centre for Research in Photonics, Ottawa, K1N 6N5, Canada. [3]University of Ottawa, School of Electrical Engineering and Computer Science, Ottawa, K1N 6N5, Canada. [4]University of Ottawa, Department of Mechanical Engineering, Ottawa, K1N 6N5, Canada. Correspondence and requests for materials should be addressed to J.B. (email: jbaxt089@uottawa.ca) or A.C.L. (email: antonino.calalesina@uottawa.ca)

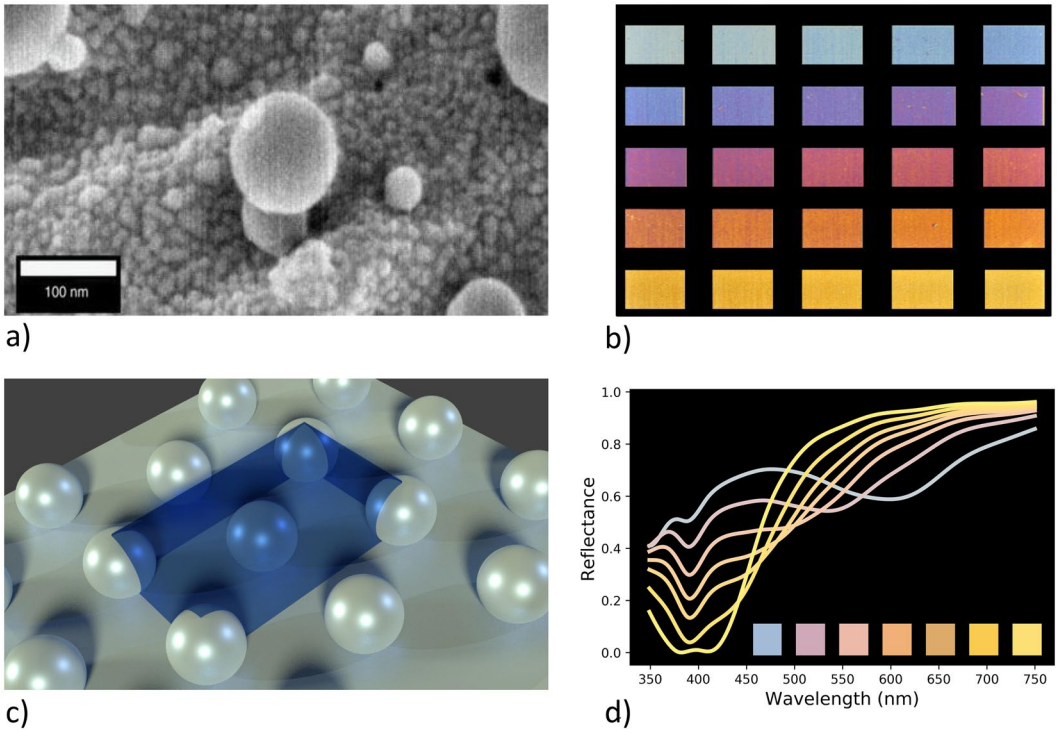

Figure 1. (**a**) High-magnification SEM image of a laser-machined silver surface. (**b**) Colour palette obtained by the laser-machining of silver surfaces[11]. (**c**) Simplified model of the silver surface used in FDTD simulations (where the blue box is the unit cell). (**d**) Simulated reflectance spectra as a function of inter-nanoparticle distance. (**a**,**b**) Are used and modified (stretched and removed labels from **b**) with permission under the Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/).

geometries to computed colours. We then apply DNNs to aid in the solution of the inverse design problem, whereby inputting colours, the model returns appropriate geometric or laser parameters. In solving the inverse design problem, we use an iterative multivariable inverse design method in which the inverse solution is found by iteratively solving for individual outputs. To the best of our knowledge, this is the first time this method has been reported. Traditional optimization and inverse design techniques (e.g. genetic algorithms or gradient-based methods) have been applied to photonics applications[14]. These methods would be time consuming for the large-scale simulations, and costly to implement in the lab. This is why we highlight the importance of using the data that has already been accumulated to train efficient DL models.

Direct Prediction of Colours

Using simulation data. In this section we use DNNs for the prediction of colours from nanoparticle geometries using the FDTD dataset. The FDTD simulations were run to calculate the reflectance spectra from single-sized silver nanoparticle distributions embedded into flat silver surfaces.

Statistical analysis of SEM images of laser-coloured silver surfaces show a bimodal distribution of nanoparticle sizes, which are referred to as medium nanoparticles ($10 < R_m < 75$ nm) with an average radius of ≈35 nm, and small nanoparticles ($R_s < 10$ nm)[11,12]. Simulations have shown that the effect of the small nanoparticles averages out, thus they do not contribute significantly to the colours observed in experiments[11,12]. Indeed, the experimental palette (see Fig. 1(b)) can be reproduced with single-sized nanoparticles of radius 35 nm (see Fig. 1(d)). For this reason, we use the single-sized nanoparticle data for this study.

The particles are arranged in a hexagonal lattice in the x-z plane as shown in Fig. 2, where only a unit cell is simulated and periodic boundaries are placed in the x- and z- directions. We excited the system with a broadband plane wave pulsed signal which allows us to capture the full visible spectrum (broadband simulation) with a single run of the code. The incoming pulse was z-polarized and propagated in the y-direction, which is truncated by Convolutional Perfectly Matched Layers (CPMLs). Simulations are run with particle radii R ranging from 5 nm to 100 nm. The embedding depth R_{emb} represents the percentage of the radius embedded into the silver surface, and it ranges from 30 to 100%, where an embedding of 100% means that we simulate a hemisphere. The inter-particle spacing D_{cc} describes the center-to-center spacing between the nanoparticles. To remove correlation between spacing and radius, we also define an edge-to-edge spacing $D_{ee} = D_{cc} - 2R$ which ranges from 5 nm to 80 nm. The simulation domain was discretized using a uniform space step and a cubic Yee cell[15] ($dx = dy = dz$). We used a space-step of $dx = 0.125$ nm for $R = 5$ to 10 nm, $dx = 0.25$ nm for $R = 15$ to 20 nm, $dx = 0.5$ nm for $R = 25$ to 45 nm and $dx = 1$ nm for $R = 50$ to 100 nm. The total number of Yee cells depends on the space step and geometry but is on the order of 10^8. We run convergence tests to ensure sufficient accuracy[16]. The optical parameters of silver were modelled by the Drude + 2 Critical Points model[17,18]. To simulate the metal substrate as a half plane, we used a

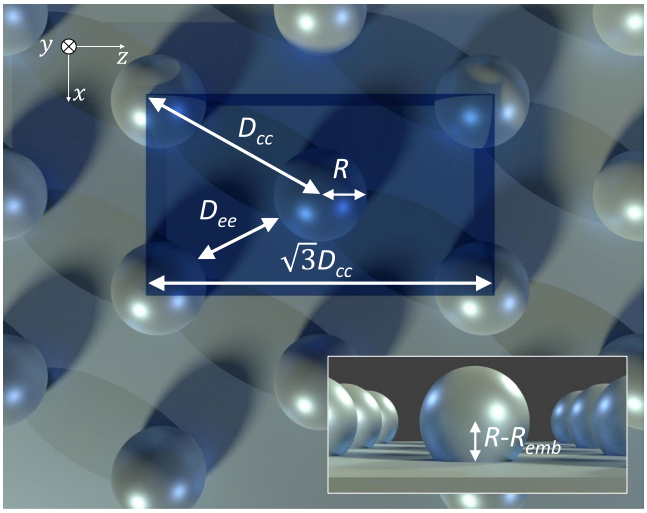

Figure 2. Layout of the nanoparticle geometry in the *x*-*z* plane. The blue tinted box is the unit cell used in the FDTD simulations.

100 nm thick silver layer, at the end of which there was zero transmittance; this layer was terminated by a CPML. The length of the y-dimension was chosen such that the reflectance is computed in the far-field region, which is at least 100 nm from the top of the nanoparticles. The reflectance spectra were computed over the optical wavelength range of 350–750 nm and were converted to RGB values. Simulations were performed using an in-house 3D-FDTD parallel code[16] running on the IBM BlueGene/Q supercomputer (64 k cores)[19] operated by SOSCIP at Scinet, University of Toronto, Canada.

For the direct prediction DL application, the input is the simulation geometries which are parameterized by nanoparticle radius R, embedding depth R_{emb}, and edge-to-edge nanoparticle spacing D_{ee}. The output is one of the colour values R, G, or B. Our total simulation dataset consists of 770 datapoints.

To optimize the use of DL, we need first to find the most ideal DNN architecture: the number hidden layers (HLs) and the number of nodes in each HL that best models our dataset. N-fold Cross Validation (CV)[20] is a process where the data is separated into N subsets. A DNN is trained using N-1 of the subsets. The Nth subset is used for testing and the error is recorded. This process is repeated N times so that each subset is used for testing. The error of a given DNN architecture is the average error from the N DNNs on the test sets.

Ten-fold CV is applied to DNNs with HL number ranging from 1 to 10, and nodes per HL ranging from 4 to 72. We simplify this process by only making DNNs with equal numbers of nodes per HL. Seeing as each DNN outputs only one value (be it R, G, or B), this process is repeated for each colour component. To find an ideal DNN architecture we decided to find one with a minimum sum of normalized Mean Square Errors (MSEs):

$$\min\left(\frac{R_{MSE}}{\min(R_{MSE})}+\frac{B_{MSE}}{\min(B_{MSE})}+\frac{G_{MSE}}{\min(G_{MSE})}\right) \tag{1}$$

where R_{MSE}, G_{MSE}, and B_{MSE} are the mean squared error for the R, G and B values respectively. We also want a DNN that uses a minimal number of HLs and nodes, in order to avoid over-fitting and long training times.

Because of the large number of DNNs trained in the CV process, we made use of the SOSCIP GPU cluster[21] where we could train 4 DNNs consecutively on a single node consisting of 4 GPUs. The DNN codes were written using TensorFlow[22] and the weights of the DNNs were optimized using an Adam Optimizer[23].

Figure 3 shows the normalized MSE sum as a function of nodes per HL for all of HLs considered. We see that the error plateaus for five or more HLs, after 40 nodes per HL. Beyond this point, the DNNs all have similar accuracies. Because of this, the five HL DNN with 40 nodes per HL, as sketched in Fig. 4(a), was chosen to model the simulation dataset.

A node in a hidden layer will receive as inputs the outputs from a previous layer, as shown in Fig. 4(b). The node performs a linear transformation on the inputs:

$$f(\mathbf{x}^{(n-1)}) = b_j^{(n)} + \sum_i a_{i,j}^{(n)} x_i^{(n-1)} \tag{2}$$

where $a_{i,j}^{(n)}$ is the weight of the jth node in the nth hidden layer applied to the ith input, and $b_j^{(n)}$ is the bias of the jth node in the nth hidden layer. The output of this node is given by the rectified linear activation function:

$$x_j^{(n)} = \max(f(\mathbf{x}^{(n-1)}),\ 0) \tag{3}$$

The training process involves adjusting the weights $a_{i,j}^{(n)}$ and the biases $b_j^{(n)}$ to minimize the error on the training set. This DNN was trained using 90% of the available data. The remaining randomly selected 10% not used in the training process was used for testing.

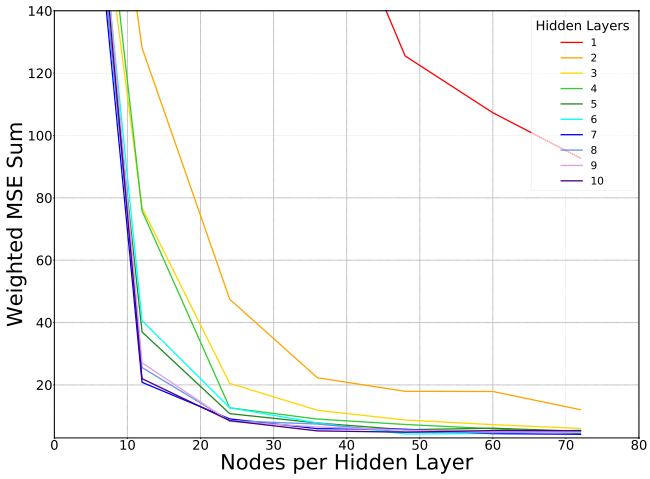

Figure 3. Weighted sum of the MSE for different DNN architectures for the simulation dataset.

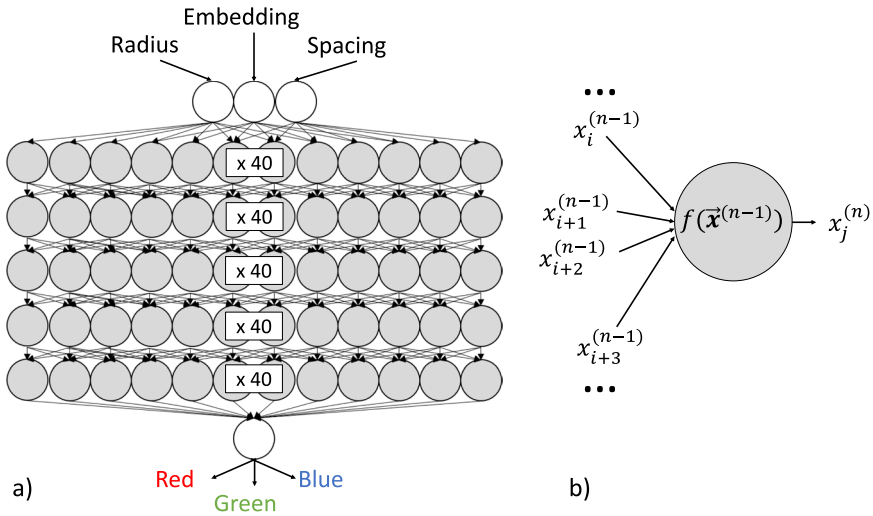

Figure 4. (**a**) Sketch of the DNN used on the simulation dataset. (**b**) The jth hidden layer node of the nth hidden layer.

Figure 5(a) shows a palette of some of the colours used in the test set, where the calculated colours from the FDTD simulations are shown on the left and the DNN predicted colours are shown on the right for the same geometric parameters. To the right of the colour pair in Fig. 5(a) we report the corresponding ΔE value[24] which is dimensionless and provides a quantitative measure of the difference between the two colours:

$$\Delta E = \sqrt{(L_1 - L_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2} \tag{4}$$

where L_i, a_i, and b_i (i = 1, 2) are the colour coordinates in an alternate colour space, calculated from the RGB values[13]. For reference, a ΔE of 1 is the threshold of a colour difference that is just perceptible to the human eye, and 6–7 (as a subjective threshold) is often deemed acceptable in commercial practice. In the test set of 77 colours, only 8 surpass a ΔE of 7. For example, the penultimate block in Fig. 5(a) yields $\Delta E = 7.73$, yet the colour difference is only somewhat noticeable. The mean ΔE of the entire test set is 2.61. The colour palette and corresponding ΔE demonstrate the predictive accuracy of our model.

Figure 5(b) compares the RGB values of FDTD vs DNN for the entire test set. Each circle represents a single datapoint, where the x-axis is the R, G, or B value of the FDTD calculation, and the y-axis is the corresponding R, G, or B value predicted by the DNN. The colour of the circle is either red, green or blue, corresponding to which RGB component that datapoint represents. The ideal linear model ($y = x$) is also displayed for convenience as a solid line. The mean percent error on the test set for R, G, and B are 0.48%, 0.67%, and 3.31% respectively, showing that the DNN chosen predicts all three value with very good accuracy. The larger error for the blue is likely due to the relative lack of representation of low blue values in the dataset. The DNN was not given enough training data with blue values below 0.6 where we clearly see the most error in Fig. 5(b). We can see from the plot that numerical values are well predicted from the model.

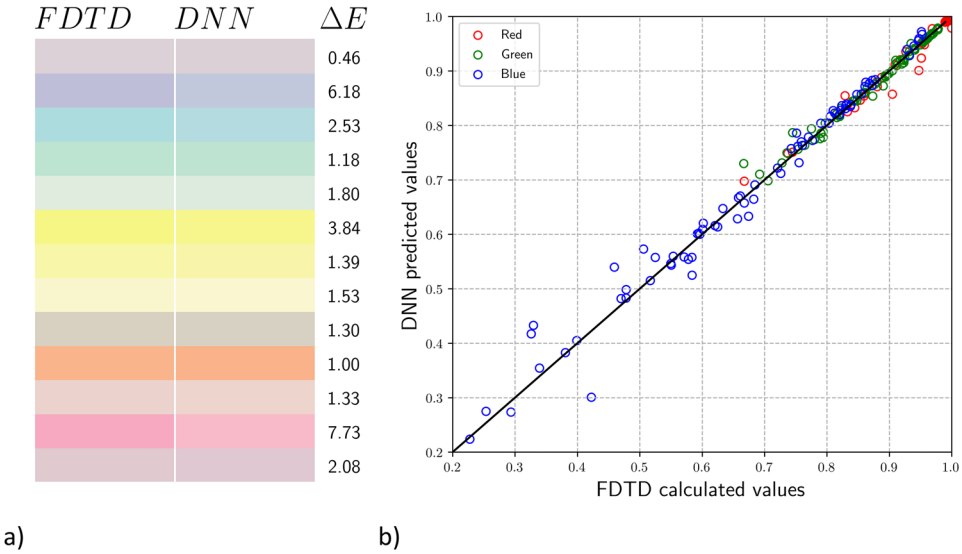

Figure 5. FDTD simulation colours vs. DNN prediction. (**a**) Comparison between FDTD simulation colours (left) and DNN predicted colours (right) on the test set with corresponding ΔE values. (**b**) Comparison between RGB values from the simulation results (x-axis) and the DNN predictions (y-axis) along with the ideal linear model ($y = x$). The colour of each circle represents whether the point represents an R (red), G (green), or B (blue) datapoint.

It is worth noting that this model can only be used for interpolation and very small extrapolations. Large extrapolations can lead to nonphysical results. Over the range spanned by our data, this model can predict the colours with high accuracy (less than 4% error) without having to run additional simulations.

Using laser machining data. In this section we repeat the above analysis for the laser parameter dataset. This dataset consisted originally of 10 laser parameters and 1841 datapoints. We eliminated 6 laser parameters due to lack of variation and effect on colour, which left 4 laser parameters under consideration: laser scanning speed, hatch spacing, number of laser pulses per burst, and laser fluence. Physically these parameters affect the total accumulated fluence on the metal surface. This was found empirically to be the key factor in colour production, so that different laser parameters with the same total accumulated fluence produced very similar colours[11]. While eliminating laser parameters, we also removed the corresponding datapoints to ensure that the training process was not affected. The number of datapoints in our updated set is 1745.

We use the process outlined in the previous section to find the ideal DNN for this dataset. We choose our model to be a DNN with 3 HLs and 60 nodes per hidden layer.

We split our data randomly into two sets: 90% for training and 10% for testing. We again display a comparison between some of the DNN predicted colours and those measured in the lab in Fig. 6(a) along with the ΔE for the test set. The error is higher for this dataset in comparison to the simulation dataset. There is a much more complicated relationship between laser parameters and the RGB of the resulting colour. Despite this, the DNN has found enough relationships to be able to predict a colour within a given range of the measured colours, resulting in a mean ΔE of 5.27 for the test set, where 41 of the 175 test datapoints are greater than 7. We can see a few of these in Fig. 6(a), as the second, third, fifth, and last blocks from the top. The first and third, although having high ΔE values, are still subjectively well predicted. The last block however is the result of a large difference (greater than 0.1) in the R, G and B values. Large errors like these are rare but should be expected in a large test set such as this, where there will likely be points that are not well represented in the training data. This means that the DNN is not as well trained in these regions of parameter space. In Fig. 6(b) we display the comparison of RGB values for the entire test set. The mean percent error for R, G and B are 4.29%, 4.88%, and 5.47% respectively. We see that this DNN predicts each colour reasonably well. However we point out that care should be taken when applying DL to the prediction of subjective results – here, even a small error can change the colour perceptively.

Implicit parameters like temperature or humidity deviations in a laboratory environment may also affect the colour, in addition to the explicit parameters such as laser settings considered here. In commercial setting, the time of day may be relevant due to workplace traffic or machinery vibrations. The relationship between these implicit effects and the resulting colour is very complicated. Though not tracked in our work, these are all parameters that can used as additional inputs into a deep learning model.

Inverse Design Prediction from Colours

Using simulation data. In this section we solve the inverse problem: we find the simulation geometries that are required to produce given colours, where the RGB values are now the inputs. Solving the inverse problem is not a trivial task as the solution may not be unique. Multiple geometries may produce the same colour so a DNN cannot be directly trained to predict geometric parameters from a given colour. Methods to get around the uniqueness problem have been suggested, including (1) training the inputs using back-propagation with the

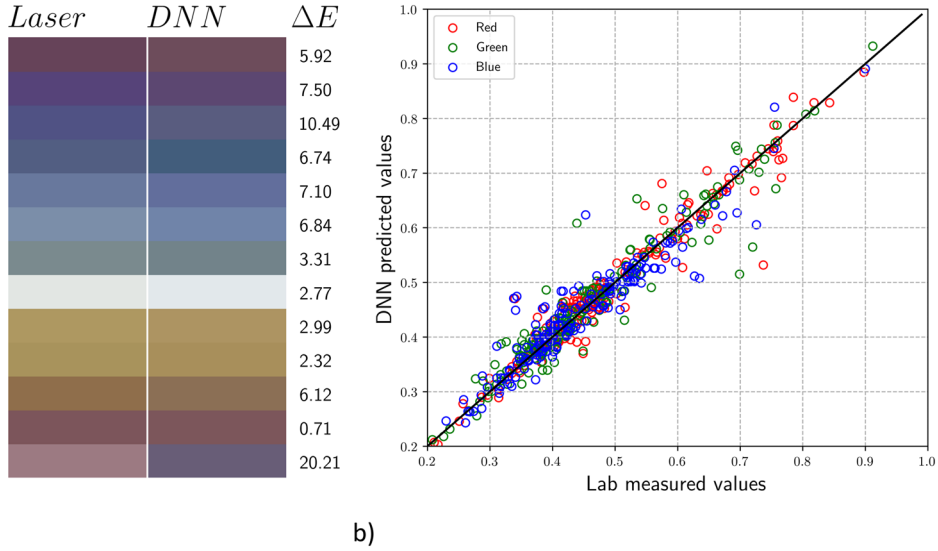

Figure 6. Laser parameter colours vs. DNN prediction. (**a**) Comparison between the measured colours (left) and DNN predicted colours (right) with corresponding ΔE values. (**b**) Comparison between RGB values from the experimental measurements (x-axis) and the DNN predictions (y-axis) along with the ideal linear model ($y = x$). The colour of each circle represents whether the point represents an R (red), G (green), or B (blue) datapoint.

weights of the DNN fixed[7], (2) training the inverse DNN using the direct DNN[8,10], and (3) applying reinforcement learning[9]. Here we introduce a more straightforward iterative multivariable inverse design approach, which to our knowledge remains unreported.

Our approach involves training n neural networks; $\{DNN_1, DNN_2, \ldots, DNN_n\}$, where n is the number of outputs (in this case our geometric parameters), given by $\{O_1, O_2, \ldots, O_n\}$. DNN_1 is trained to calculate output O_1 from the desired inputs (in this case the RGB values) and the remainder of the outputs $\{O_2, \ldots, O_n\}$. This means that the remainder of the outputs act as inputs for DNN_1 alongside the desired RGB values. DNN_2 is trained to calculate O_2 using the desired inputs and the remainder of the outputs $\{O_1, O_3, \ldots, O_n\}$, and so on.

Once all n DNNs are trained, we can calculate the set of outputs in an iterative procedure. We first calculate O_n using the DNN_n with the desired inputs and the remaining outputs (which can be initialized as random numbers or the mean values of the dataset). Next we calculate O_{n-1} using DNN_{n-1} with the desired inputs and the remaining outputs including the updated O_n. This is repeated for all outputs. This process can be repeated until the values of the outputs have relaxed to a final value. This method is very simple in implementation since it requires no special treatment of the DNNs unlike inverse design methods suggested in previous studies[7–10].

In the case of our simulation geometry, $n = 3$. DNN_1 is trained to predict the particle spacing from R, G, B, radius, and embedding. DNN_2 predicts the embedding from R, G, B, radius, and spacing. DNN_3 predicts the radius from R, G, B, embedding, and spacing. After training these three DNNs, we can then calculate the three geometric parameters using the iterative procedure outlined in Fig. 7(a).

First the radius and embedding are set as the mean radius and mean embedding of the dataset, providing a good starting place for the parameter search. We then calculate the spacing using our spacing DNN. The calculated spacing value and mean radius value are used to predict the embedding. This embedding and the spacing are then used to calculate the radius. We then repeat these steps until the geometric parameters relax to their final values. This generally this takes 10–20 iterations, however, this may vary depending on the colours themselves. If the colours are not well represented in the training data, the number of iterations will increase. As sketched in Fig. 7(b), we use 5 HLs with 40 nodes per hidden layer for our backwards DNNs.

A sample colour palette of our test dataset is shown in Fig. 8(a) where on the left we display a set of desired colours, and on the right we display the colours generated by the output geometric parameters. We see an agreement between the input colour, and that generated by our iterative method. We also see ΔE values within the same range as the direct prediction with a mean of 2.83. The comparison between input and output RGB values are given in Fig. 8(b) where the percent error between the input and output colours are 0.83%, 1.34%, and 2.12% for R, G and B respectively. The slight increase in error between the direct and inverse prediction is expected due to error accumulation between the backward DNNs used to calculate the geometries and the forward DNN used for testing the geometries. The geometries output by the relaxation are not guaranteed to provide the closest possible colour to the input. They depend on the initial inputs (O_1 and O_2 in Fig. 7). Multiple initialized inputs can be used to find multiple possible geometries, and the one which has the least error between input and output colours should be used.

In traditional optimization methods such as gradient based methods and genetic algorithms, solutions are iteratively sought by trying to minimize an error function. Depending on the complexity of the problem, this can require hundreds or thousands of iterations where at least one simulation must be run for each iteration,

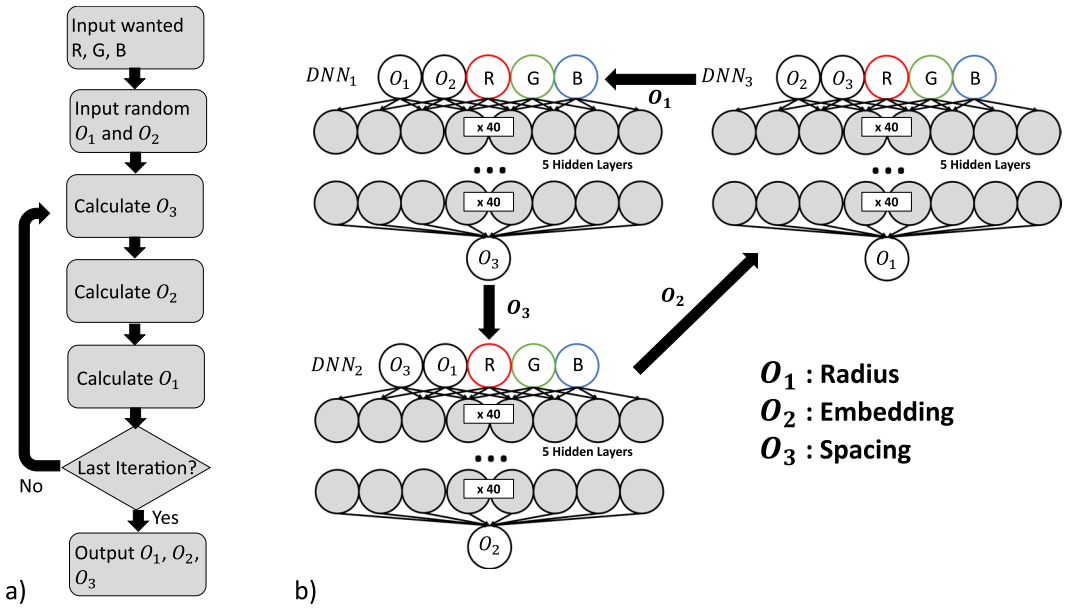

Figure 7. (**a**) Iterative multivariable inverse design algorithm for the prediction of geometric parameters from colours. We input the desired R, G, B and initial (random or mean) values for the radius and embedding. We then iterate through DNN calculations for spacing, embedding, radius until the last iteration. (**b**) DNN calculation iteration procedure.

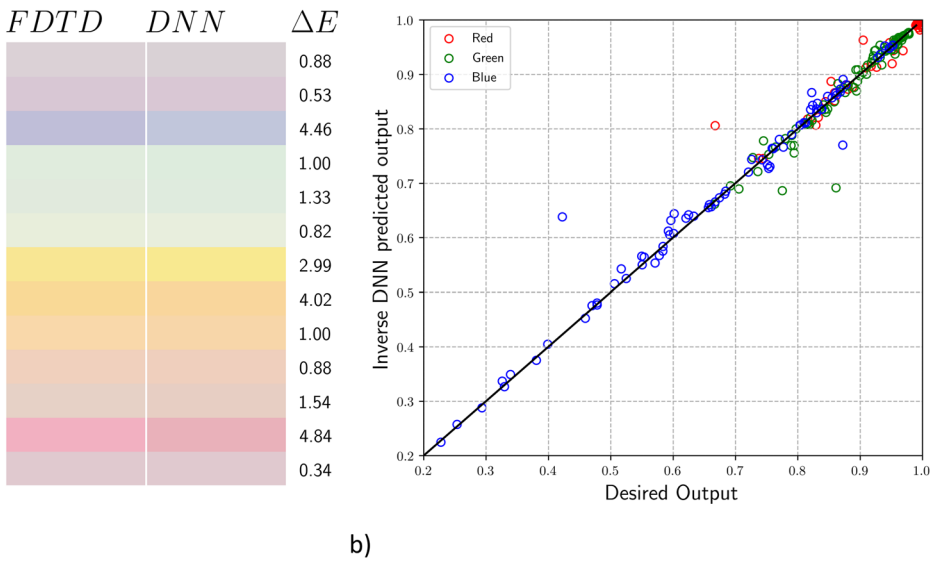

Figure 8. Geometric parameter inverse DNN prediction. (**a**) Comparison between the input test colours (left) and the colours produced by the output geometric parameters (right) with corresponding ΔE values. (**b**) Comparison between input RGB values (x-axis) and the values produced by the output geometric parameters (y-axis) along with the ideal linear model ($y = x$). The colour of each circle represents whether the point represents an R (red), G (green), or B (blue) datapoint.

and all this to obtain only one optimized solution. The multivariable inverse design method also requires a large number of simulations upfront for data acquisition, the number of which may be comparable to what is required for an optimization method. However, once these data are used to train the DNNs, it is then possible to obtain a large number of optimized solutions very rapidly. In this study, we seek to find many optimized solutions satisfying different constraints using an existing simulation dataset. Thus using DNNs – which take less than a second per prediction, versus hours on a super computer for a single simulation iteration – wins over traditional optimization.

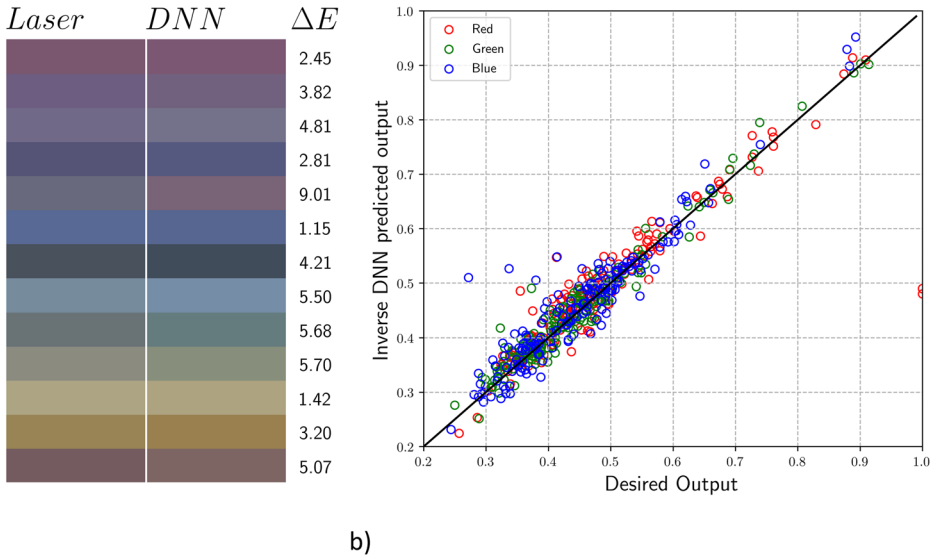

Figure 9. Laser parameter inverse DNN prediction. (**a**) Comparison between the input test colours (left) and the colours produced by the output laser parameters (right) with corresponding ΔE values. (**b**) Comparison between input RGB values (x-axis) and the values produced by the output laser parameters (y-axis) along with the ideal linear model ($y = x$). The colour of each circle represents whether the point represents an R (red), G (green), or B (blue) datapoint.

Using laser machining data. The laser parameter dataset was obtained from the Royal Canadian Mint, who uses this colouring technique to create artwork on silver[11]. The colours for this artwork are chosen carefully using a palette of known experimentally-obtained colours. The laser is then programmed with the corresponding laser parameters to create this colour on the metal surface. This process is restricted to the list of known colours. However, by using DNNs to predict the laser parameters required to produce a given desired colour, this process can be simplified and laser parameters can be predicted for colours that are not within the known list.

We use the iterative multivariable inverse design procedure introduced in the previous section with $n = 4$ for the four laser parameters that we want to predict (the same ones as used for Fig. 6). We train our four backward DNNs with 90% of the data and test it with the other 10%. These DNNs have three HLs and 60 nodes per HL. We use the direct prediction DNN to calculate the colour from the predicted laser parameters. We show some of the colours output in Fig. 9(a) and display the numerical results in Fig. 9(b). The percent error between the input and output colours are 5.50%, 4.57%, and 6.09% for R, G, and B, respectively, and the mean of ΔE is 6.31, which remains less than our limit of 7. Numerically these errors are small and result in the linear pattern displayed in Fig. 9(b). However, these errors are large enough that some colours are switched in the process. For example, the fifth box from the top in the palette of Fig. 9(a) reveals a change from the desired grey to the output purple. Such changes are caused by differences in the RGB values of the order of 0.07. These differences should be expected due to the accumulation of error in experiment, the backward DNNs, and the forward DNN used to test the backward DNNs. Despite the error in some of the colours, the majority of them are accurately predicted.

Conclusion

We have applied deep learning to plasmonic metal colouration through (1) direct prediction of colours and (2) inverse prediction of parameters required for a desired colour, using both simulation and experimental data sets, with good prediction accuracy. The direct prediction of colours from new parameters means that we increase the palette of available colours without needing to run further simulations or experiments. The Royal Canadian Mint has started using laser colouring on silver[11] where the appropriate colours are chosen manually from the database. The inverse prediction can automate this process by finding the laser parameters (that are not necessarily in the database) that best match the desired colour. This method allows for the fixing of certain input parameters, like scanning speed or laser fluence. Fixing the scanning speed is of particular interest, as with higher scanning speeds, a surface can be coloured in less time. The direct prediction can be expanded for industrial applications where environments (temperature, humidity, vibrations, etc) are much less controlled. Environmental factors can be taken into account in the training of the deep neural networks and therefore the output colour of a laser machined surface can be predicted with higher accuracy. Finally, the relaxation technique we developed for the inverse prediction has numerous potential applications in nanophotonics[14]. Since it is much easier to implement than advanced methods such as reinforcement learning, and faster than traditional optimization techniques like gradient based methods, it is more accessible to those new to the field of deep learning.

Data Availability

The datasets generated and analyzed during the current study are available from the corresponding authors on reasonable request.

References

1. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nat.* **521**, 436–444, https://doi.org/10.1038/nature14539 (2015).
2. Modarres, M. H. *et al.* Neural Network for Nanoscience Scanning Electron Microscope Image Recognition. *Sci. Reports* **7**, 13282, https://doi.org/10.1038/s41598-017-13565-z (2017).
3. Rivenson, Y. *et al.* Deep learning microscopy. *Opt.* **4**, 1437–1443, https://doi.org/10.1364/OPTICA.4.001437 (2017).
4. Ma, W., Cheng, F. & Liu, Y. Deep-Learning-Enabled On-Demand Design of Chiral Metamaterials. *ACS Nano* **12**, 6326–6334, https://doi.org/10.1021/acsnano.8b03569 (2018).
5. Malkiel, I. *et al.* Plasmonic nanostructure design and characterization via Deep Learning. *Light. Sci. & Appl.* **7**, 60, https://doi.org/10.1038/s41377-018-0060-7 (2018).
6. Pilozzi, L., Farrelly, F. A., Marcucci, G. & Conti, C. Machine learning inverse problem for topological photonics. *Commun. Phys.* **1**, 57, https://doi.org/10.1038/s42005-018-0058-8 (2018).
7. Peurifoy, J. *et al.* Nanophotonic particle simulation and inverse design using artificial neural networks. *Sci. Adv.* **4**, eaar4206, https://doi.org/10.1126/sciadv.aar4206 (2018).
8. Liu, D., Tan, Y., Khoram, E. & Yu, Z. Training Deep Neural Networks for the Inverse Design of Nanophotonic Structures. *ACS Photonics* **5**, 1365–1369, https://doi.org/10.1021/acsphotonics.7b01377 (2018).
9. Sajedian, I., Badloe, T. & Rho, J. Finding the best design parameters for optical nanostructures using reinforcement learning. *arXiv:1810.10964 [physics]* ArXiv: 1810.10964 (2018).
10. Liu, Z., Zhu, D., Rodrigues, S. P., Lee, K.-T. & Cai, W. Generative Model for the Inverse Design of Metasurfaces. *Nano Lett.* **18**, 6570–6576, https://doi.org/10.1021/acs.nanolett.8b03171 (2018).
11. Guay, J.-M. *et al.* Laser-induced plasmonic colours on metals. *Nat. Commun.* **8**, 16095, https://doi.org/10.1038/ncomms16095 (2017).
12. Guay, J.-M. *et al.* Topography Tuning for Plasmonic Color Enhancement via Picosecond Laser Bursts. *Adv. Opt. Mater.* **6**, 1800189, https://doi.org/10.1002/adom.201800189 (2018).
13. Math|EasyRGB. Available at: https://www.easyrgb.com/en/math.php. (Accessed: 24th October 2018).
14. Molesky, S. *et al.* Inverse design in nanophotonics. *Nat. Photonics* **12**, 659, https://doi.org/10.1038/s41566-018-0246-9 (2018).
15. Yee, K. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas Propag.* **14**, 302–307, https://doi.org/10.1109/TAP.1966.1138693 (1966).
16. Calà Lesina, A., Vaccari, A., Berini, P. & Ramunno, L. On the convergence and accuracy of the FDTD method for nanoplasmonics. *Opt. Express* **23**, 10481–10497, https://doi.org/10.1364/OE.23.010481 (2015).
17. Prokopidis, K. P. & Zografopoulos, D. C. A Unified FDTD/PML Scheme Based on Critical Points for Accurate Studies of Plasmonic Structures. *J. Light. Technol.* **31**, 2467–2476, https://doi.org/10.1109/JLT.2013.2265166 (2013).
18. Vial, A., Laroche, T., Dridi, M. & Le Cunff, L. A new model of dispersion for metals leading to a more accurate modeling of plasmonic structures using the FDTD method. *Appl. Phys. A* **103**, 849–853, https://doi.org/10.1007/s00339-010-6224-9 (2011).
19. BGQ - SciNetWiki. Available at: https://wiki.scinet.utoronto.ca/wiki/index.php/BGQ. (Accessed: 24th October 2018).
20. James, G., Witten, D., Hastie, T. & Tibshirani, R. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics (Springer-Verlag, New York, 2013).
21. SOSCIP GPU - SciNet Users Documentation. Available at: https://docs.scinet.utoronto.ca/index.php/SOSCIP_GPU. (Accessed: 24th October 2018).
22. TensorFlow. Available at: https://www.tensorflow.org/. (Accessed: 24th October 2018).
23. Kingma, D. P. & Ba, J. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* ArXiv: 1412.6980 (2014).
24. Guay, J.-M. *et al.* Passivation of Plasmonic Colors on Bulk Silver by Atomic Layer Deposition of Aluminum Oxide. *Langmuir* **34**, 4998–5010, https://doi.org/10.1021/acs.langmuir.8b00210 (2018).

Acknowledgements

SOSCIP is funded by the Federal Economic Development Agency of Southern Ontario, the Province of Ontario, IBM Canada Ldt., Ontario Centres of Excellence, Mitacs and Ontario academic member institutions. The authors thank SOSCIP for their computational resources and financial support. We acknowledge the computational resources and support from Scinet. We acknowledge financial support from the National Sciences and Engineering Research Council of Canada, and the Canada Research Chairs program. The authors also thank the Royal Canadian Mint for the use of their laser lab and the data acquired from it. We would like to thank Graham Killaire, Meagan Ginn, Guillaume Côté, and Martin Charron for their contributions in creating the colour palettes at the Royal Canadian Mint.

Author Contributions

A.C.L. proposed the application of machine learning for colour prediction and inverse design. J.B. conducted the deep learning study including data structuring, coding the deep neural networks, and developing the inverse design method under the supervision of A.C.L. and L.R. J.B. and A.C.L. created the figures. A.C.L. designed and conducted the FDTD simulations under the supervision of P.B. and L.R., J.-M.G. conducted the experiments and supplied the experimental data under the supervision of A.W. and P.B., J.B., A.C.L. and L.R. wrote the manuscript, and J.-M.G., A.W. and P.B. were involved in the editing process.

Additional Information

Competing Interests: The authors declare no competing interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.

© The Author(s) 2019