

Deep learning the high variability and randomness inside multimode fibers

PENGFEI FAN, TIANRUI ZHAO, AND LEI SU*

School of Engineering and Materials Science, Queen Mary University of London, London E1 4NS, UK

**l.su@qmul.ac.uk*

Abstract: Multimode fibers (MMF) are remarkable high-capacity information channels. However, the MMF transmission is highly sensitive to external perturbations and environmental changes. Here, we show the successful binary image transmission using deep learning through a single MMF subject to dynamic shape variations. As a proof-of-concept experiment, we find that a convolutional neural network has excellent generalization capability with various MMF transmission states to accurately predict unknown information at the other end of the MMF at any of these states. Our results demonstrate that deep learning is a promising solution to address the high variability and randomness challenge of MMF based information channels. This deep-learning approach is the starting point of developing future high-capacity MMF optical systems and devices, and is applicable to optical systems concerning other diffusing media.

1. Introduction

Multimode fibers (MMF) have recently attracted significant renewed interest in applications such as optical communication, imaging and optical trapping [1-12]. This is mainly due to the extremely high information-transmission capacity offered by the thousands of fiber modes simultaneously transmitting in the MMF [13-15]. For example, individual spatial modes or mode groups in a single MMF were proposed as separate information channels. This can be combined with other optical communication multiplexing technologies such as wavelength division multiplexing (WDM), to break the transmission limit of a single optical fiber. Another attractive application is an ultrathin single MMF endoscope based on the large number of fiber modes to perform high-resolution in vivo imaging. At the same time, however, MMF transmission is highly sensitive to external perturbations and environmental changes [16-18], resulting in MMF transmission channels being highly variable and random. For optical communications over MMFs, such changes lead to mode coupling among different guided modes or mode groups in an MMF, resulting in cross talks among different transmitting channels and a high bit error rate. For a single MMF endoscope, despite great research efforts made to overcome the high variability and randomness inside MMFs, any geometric change to the MMF leads to completely different transmission matrices (TM). Whilst the TM-based image reconstruction approach has excellent capability to reconstruct a complex image, a TM is only applicable to the transmission state in which it is calibrated. This unavoidably fails at the image recovery if the MMF transmission channel has changed. This largely limits the practical application of MMFs and hinders the full exploitation of their information capacity.

Deep learning is a booming class of machine learning methods that allows computational models structured of multiple processing layers to automatically discover the intricate representations of data with multiple levels of abstraction [19, 20]. Recently, the convolutional neural network (CNN), one type of the deep neural network, has been successfully applied to address optical research problems such as microscopy resolution enhancement [21, 22], optical experiment design [23], optical imaging [24-27] and optical communications [28]. Interestingly, Lyu et al earlier demonstrated image transmission through an opaque wall via deep learning [29]. This implies that similar approaches may be applied to transmit images or information through an MMF. Different from other diffusing

media, the MMF transmission properties are highly variable according to the fiber's geometry. After that, several groups, including us, demonstrated the use of deep learning for output pattern prediction [30] and input MNIST digits reconstruction through an MMF [31-34]. It was suggested in [31], an encoder-decoder 'U-net' architecture learns the environment-introduced instability in a 1-km-long MMF.

In the paper, different from other recent studies, we attempt to use deep learning to overcome the long-standing high variability and randomness challenge of MMF by introducing strong MMF geometric shape variations, in order to turn MMFs into practical imaging components or communication devices. The proof-of-concept studies consist of MMF transmission simulations and experiments using CNN, including: i) MMF image transmission using output speckle patterns calculated by an experimentally measured TM; ii) image retrieval through a stationary MMF subject to different degrees of bending, and iii) image retrieval through an MMF subject to continuous shape variations (CSV).

2. Methods

2.1. Experimental setup

We use a simple digital micromirror device (DMD) based single MMF transmission system [35, 36] to demonstrate the idea, as shown in Fig. 1. The DMD can be controlled to display any binary pattern, simulating a 2D object imaging from the DMD through the MMF to the camera. The laser beam (632.8nm, 17mW, 25-LHP-925-230, Melles Griot) is expanded by Lens 1 (L1, Mounted Rochester Aspheric Lens, focal length = 11.00 mm, NA = 0.30, A397TM-A, Thorlabs) and Lens 2 (L2, Bi-Convex Lens, focal length = 100.0 mm, LB1187-A-ML, Thorlabs) and is then projected onto a DMD (1024×768 micro-mirrors, mirror size 13.68μm×13.68μm, mirror tilt +/- 12 degrees, operating at 20Hz, maximum frequency 9.8kHz, Discovery 1100, Texas Instruments). Driven by an interface board (ALP-1, ViALUX), the DMD can display any arbitrary binary pattern by turning 'ON' and 'OFF' individual DMD micromirrors. The incident laser beam can be modulated by the displayed DMD pattern and is subsequently coupled into the proximal end of a multimode fiber (MMF, 50μm-core, 35cm-long, FG050UGA, Thorlabs) through a tube lens (TL1, AC254-200-A-ML, focal length = 200 mm, Thorlabs) and a microscope objective (OL1 Nikon CFI Plan Achro 40×, 0.65 NA, Working Distance: WD = 0.56 mm, tube lens focal length: 200 mm). At the other end of the MMF, a microscope objective (OL2, Olympus 20× Plan Achro, 0.4 NA, 1.2 mm working distance, tube lens focal length: 180 mm) and a tube lens (TL2, AC254-200-A-ML, focal length = 200 mm, Thorlabs) are used to expand the output beam from the MMF distal end. The output is then sent to a CMOS camera (2048×2048 pixels, 6.5μm×6.5μm pixel sizes, operating at 20fps, maximum frame rate 100fps, C1140-22CU, Flash4.0, Hamamatsu).

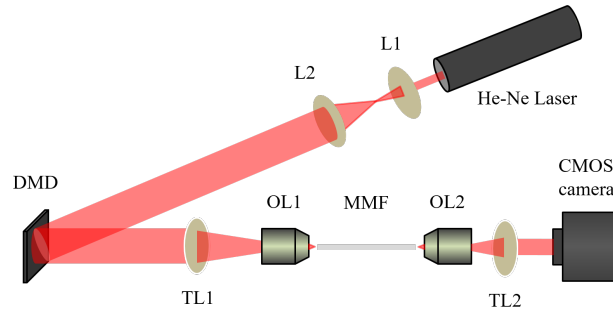


Fig. 1. The experimental setup used to obtain the data pairs for training and testing the neural network. L1, L2: Bi-Convex Lenses; DMD: Digital Micromirror Device; TL1, TL2: Tube Lens; OL1, OL2: Objective Lens; MMF: Multimode Fiber.

2.2. MMF TM measurement

In Fig. 1, we set 2×2 micromirror pixels as a single macro pixel in the modulation to enhance the difference between the “ON” and the “OFF” pixels. The input DMD modulation pattern consists of 36×36 ($N = 1296$) macro pixels and the output pixels captured by CMOS camera is 96×96 ($M = 9216$). N is determined by the maximum number of modes supported in the MMF to ensure sufficient degrees of freedom. M is determined by the MMF core diameter, NA and the magnification of the objective lens. We generate $P = 8000$ random binary patterned input fields, with an “ON” to “OFF” pixel ratio of 50:50. The response of a single macro pixel is therefore measured $P/2$ times over the cycle. The DMD operates at 250 Hz. The system stability is monitored by calculating correlation coefficients of outputs for the same input over time and is above 99%. The input and output data are processed and the TM is calculated with the *prVBEM* (phase retrieval Variational Bayes Expectation Maximization) algorithm [37], which requires $M/N \geq 2$ in order to retrieve phase information successfully. Typically, the parallel calculations require approximately 5 hours for a TM of size 9216×1296 , implemented under 100 to 200 submitted parallel jobs using high performance computing facilities (a group of Intel Xeon E5645 and AMD Opteron 6234).

The performance of the TM is evaluated with the correlation coefficient (*corr2/corr* functions in MATLAB) between TM calculated outputs and experimentally measured outputs for the same input DMD patterns (random binary pattern at 50:50 “ON” to “OFF” ratio). By evaluating over 8000 input DMD patterns, we obtain an average correlation coefficient of over 98% for the TM used in our simulation.

2.3. Binary DMD input patterns preparation

To initially prepare the raw label training data for the deep neural network, we download images of handwritten digits from the MNIST handwritten digits database [38]. The raw images consisting of 28×28 pixels are converted to images with 36×36 pixels to match the input DMD pattern requirement. Since the on-board rotation of micromirrors can only realize binary intensity modulation, the 36×36 pixel images are then converted into binary images (implemented using *im.convert* function within the Python Imaging Library) to yield the final digits training labels for the deep neural network. Since we use 2×2 DMD micromirror pixels as a macro-pixel, the binary images with 36×36 pixels are used to modulate 72×72 pixels on the DMD.

3. Results and discussion

3.1. Simulation with measured TM

As the first step to verify our idea, we use TM calculated output speckle images for the deep neural network (TM-trained CNN) training and prediction [37, 39]. Firstly we calculate the MMF TM using 8000 input-output pairs collected from the experimental system. The measured TM is verified experimentally to have an average accuracy of 98% for output speckle pattern calculation. We then use the processed binary images of handwritten digits downloaded from the MNIST handwritten digits database as the DMD input pattern. The binary processing is to convert the handwritten digit into 36×36 -pixel image format that directly matches the fiber mode count and eigenchannels for information delivery as suggested in [35]. As shown in Fig. 2, the MMF output speckle images are calculated using TM and the binary handwritten digits. We calculate 7800 speckle images corresponding to different input binary images for the digits ‘0-9’ (780 speckle images for each digit). Out of these speckle images, 7020 (90%) speckle images and their corresponding binary DMD input patterns are randomly selected as our training and validating image dataset to train and validate a deep convolutional neural network (details in Appendix), as shown in Fig. 3(a). This training procedure only needs to be performed once, and the CNN is then fixed. The remaining 780 (10%) speckle images and their corresponding binary DMD input patterns are used for testing the final network model (Fig. 3(b)). To further quantify the performance of

the final trained CNN, we calculate another 780 output speckle images for handwritten letters downloaded from a different EMNIST Letters database [40].

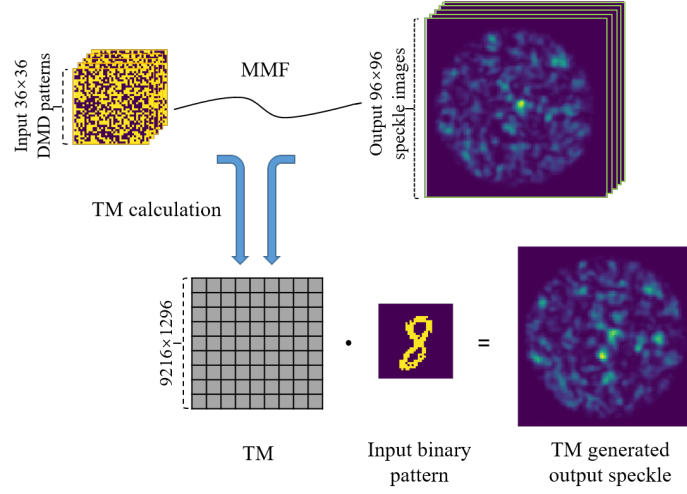


Fig. 2. Schematic of speckle generation using TM: the measured TM is used to calculate the corresponding output speckles for input digits and letters.

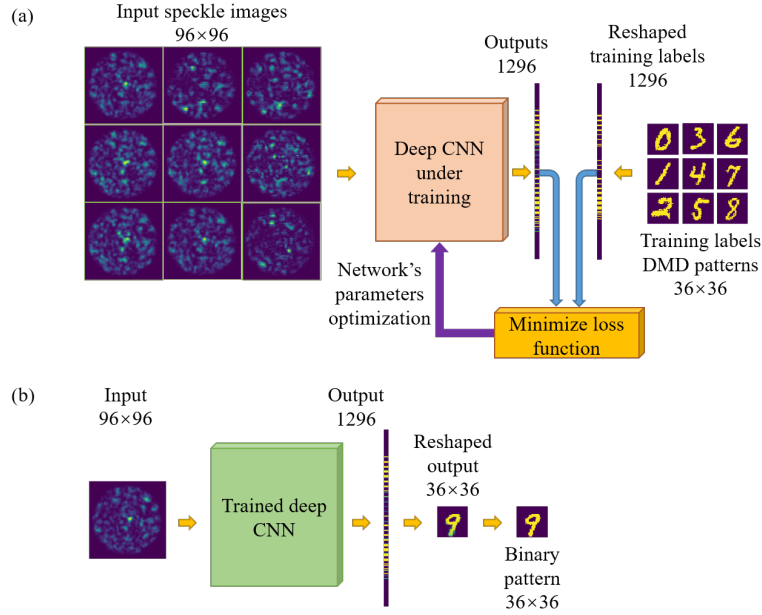


Fig. 3. Schematics of the deep neural network trained for imaging through an MMF. (a) The input is composed of a set of speckle images with 96×96 pixels generated by the TM (or captured by the camera), and the training labels are their corresponding 36×36 -pixel handwritten digits patterned on a DMD. The deep neural network is trained by optimizing various parameters, which minimize the loss function between the network's 1296-pixel output and the corresponding reshaped 1296-pixel handwritten digits training labels. See training details in Appendix; (b) After the training phase is complete, the network is blindly given a 96×96 -pixel input image and rapidly outputs a 36×36 -pixel image, reshaped from 1296 pixels. The output image is then binarily processed to get a binary pattern.

Examples of the prediction results are shown in Fig. 4 and Fig. 5. The prediction accuracy, defined as the percentage of correctly predicted pixels within one 36×36 pixels input image, is used to assess the prediction performance. For the 780 testing speckle images not used in the training process, the average prediction accuracy between the predicted binary images and the ground truth is 98.74%. For the 780 handwritten letters testing images from the other database, the network output demonstrates 95.22% average accuracy over 780 test images. The results demonstrate that the ‘0-9’ digits trained CNN can be used to predict letter DMD input patterns with significantly different shapes.































Predicted patterns										
Predicted binary patterns										
Ground truth										
Accuracy	98.37%	99.77%	98.30%	99.31%	98.69%	98.77%	98.77%	99.15%	98.69%	99.15%

Fig. 4. TM-trained CNN output images of predicted digits.































Predicted patterns										
Predicted binary patterns										
Ground truth										
Accuracy	97.38%	97.22%	97.15%	97.07%	97.07%	96.99%	96.99%	96.91%	96.91%	96.60%

Fig. 5. TM-trained CNN output images of predicted letters.

3.2. Image retrieval through stationary MMFs with different geometric shapes

In the second step, we use the experimentally obtained output speckle images to verify the neural network prediction performance. We experimentally collect input and output pairs to train the CNN instead of using the TM calculated output. Using the experimental configuration shown in Fig. 1, we modulate the DMD input patterns using the binary MNIST handwritten digits database and the EMNIST Letters database, and collect corresponding output speckle images with the CMOS camera. The above input-output-pair collection is repeated for three different geometric states of the MMF, named as G1, G2 and G3 (Fig. 6). The significant difference in light transmission between these three states is ascertained by evaluating the correlation coefficients of output speckles collected at G1, G2 and G3 for the same input DMD pattern. As shown in Fig. 6, the correlation coefficients between any of these states are below 60%.

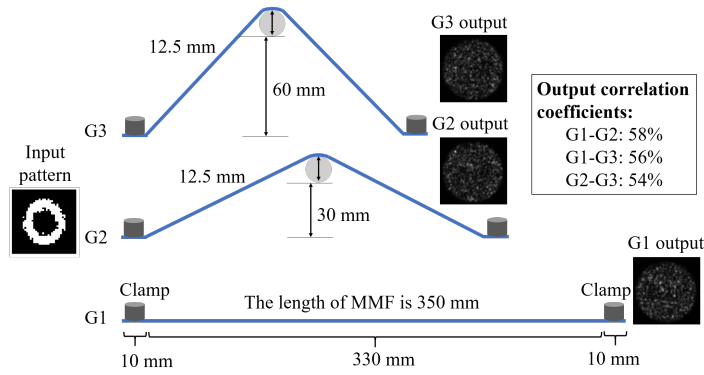


Fig. 6. MMF configurations: G1, G2 and G3 are three geometric states for the MMF under test (50 μ m-core, 350mm-long, FG050UGA, Thorlabs). The correlation coefficients between outputs from the same input at G1, G2 and G3 demonstrate the significant difference in transmission properties at these three states.

To train and test the deep CNN, we experimentally collect three different output speckle pattern datasets at these three different MMF geometries, G1, G2 and G3 respectively. For these geometries, the same DMD input dataset, including approximately 40,000 unique ‘0-9’ MNIST handwritten digits and 780 unique EMNIST letters, is used. We use approximately 90% of the experimentally collected input-output pairs from the handwritten digits database to train the deep CNN, and the remaining 10% handwritten digits input-output pairs and all handwritten letter input-output pairs for testing. Firstly, we separately train and test G1, G2 and G3 using their own datasets. The prediction results by using the CNNs trained by the data collected under the same MMF geometric states are excellent. The prediction examples for G3 are shown in Fig. 7 with an average accuracy of 97.06% over 3622 testing handwritten digit speckle images, and an average accuracy of 92.32% for 780 handwritten letters from the different EMNIST letter database (Fig. 8). This proves our simulation results in Fig. 4 and Fig. 5 experimentally. The trained CNN can successfully predict digits and letters significantly different in shape under the same MMF geometric state (i.e. the same TM). However, as expected, the trained CNN at one MMF geometric state fails to predict the corresponding inputs of output speckle images at other states. Figure 9(a) is an example using G1&G2-trained CNN to predict the input DMD pattern based on the speckle images collected at G3 and the predicted pattern significantly mismatches the ground truth, where the average prediction accuracy is 86.27% over 3622 testing handwritten figures. This accuracy is over 10% worse than that achieved by G3-trained CNN. Since all the input patterns have large proportions of black edges, this 10% difference means significant mismatching at the center areas of input patterns.

Predicted patterns										
Predicted binary patterns										
Ground truth										
Accuracy	97.84%	99.76%	97.76%	98.61%	98.84%	98.38%	98.38%	99.30%	98.14%	98.84%

Fig. 7. G3-trained CNN output images of predicted digits.

Predicted patterns										
Predicted binary patterns										
Ground truth										
Accuracy	95.60%	95.29%	95.17%	95.06%	94.98%	94.52%	94.29%	92.21%	94.14%	93.60%

Fig. 8. G3-trained CNN output images of predicted letters.

(a)										
Predicted patterns										
Predicted binary patterns										
Ground truth										
Accuracy	75.00%	90.35%	80.94%	85.11%	88.35%	91.74%	84.41%	84.72%	82.25%	86.96%
(b)										
Predicted patterns										
Predicted binary patterns										
Ground truth										
Accuracy	95.37%	99.15%	92.97%	96.15%	96.06%	96.29%	97.67%	97.37%	95.14%	96.94%

Fig. 9. (a) The output images of G1&G2-trained CNN on the G3 test dataset; (b) The output images of mixed G1&G2&G3-trained CNN on the G3 test dataset. Note that in (a) even at accuracy over 90%, the shape of the predicted result is still far from its ground truth. Using the digit '5' as an example, the predicted '5' in (a) is completely unrecognizable at 91.74% accuracy, compared to the predicted result in (b).

The following experiment leads to a key finding of our work. We use the combination of data collected at G1, G2 and G3 states to train a CNN jointly. Subsequently, the newly trained CNN is used to predict the input patterns for both the digit and letter testing speckle images collected at the G1, G2 and G3 states, respectively. As depicted in Fig. 9(b), our mixed G1&G2&G3-trained CNN provides outstanding prediction performance for testing speckle images measured at G1, G2 and G3 states. The average prediction accuracies of G1&G2&G3-trained CNN are: 96.96% for 4097 G1 state testing data, 96.31% for 3801 G2 state testing data, and 96.05% for 3622 G3 state testing data. Although slightly lower, these are in accordance with the prediction accuracies achieved by individually trained CNNs for G1, G2 and G3 (97.88%, 97.36%, and 97.06%, respectively) and are highly accurate predictions. The results demonstrate that the proposed deep neural network not only has a great performance to predict the input patterns at a certain MMF geometric state, but also exhibits a significant generalization capacity for different MMF geometric states (i.e. different TMs).

3.3. Deep learning an MMF subject to continuous shape variations

Inspired by the results above, we design an experiment to test if a deep CNN can be trained to predict input DMD patterns, using output speckle images captured when the MMF is subject

to continuous shape variations applied by a human at a random speed. Using the experimental setup in Fig. 1, we insert a steel bar (diameter 12.5 mm, length 200 mm) under the MMF. The steel bar can be translated within a range of 30 mm in the middle of the MMF, as illustrated in Fig. 11. We ensure that the MMF under test is in tight contact with the steel bar, so that the strain applied to the MMF by moving the steel bar is strong enough for significant TM variations [11]. At the same time, the strain on the MMF cannot be too strong; otherwise it may overcome the clamping friction, resulting in the fiber-end movements and misalignment. Throughout the data collection process (approximately 110 minutes, including both data uploading time and input-output pair measurement time), the steel bar is manually translated along both MMF axial directions within the 30mm range at a random speed within 0-30 mm/s. This ensures continuous and random-speed shape changes applied to the MMF, simulating the situation of manual MMF manipulations. To show significant MMF transmission property variations during the steel-bar translation, we calculate correlation coefficients of the MMF output speckle images for the same input DMD pattern between six different steel-bar locations, H1-H6, which are equally distanced along the 30mm translation range with approximately 5mm between any two adjacent locations. We show in Fig. 10 that the output speckle images for the same input DMD pattern at H1-H6 are significantly different from each other with calculated correlation coefficients around 50%. Table 1 gives a summary of the correlation coefficient evaluations of different MMF transmission states. Note that for our experiment system, the strain required to reach a correlation coefficient lower than 45% is likely to overcome the clamping friction and to disrupt the alignment at MMF ends.

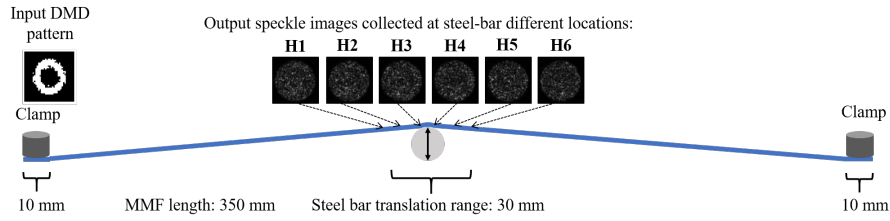


Fig. 10. Experimental configuration to apply continuous shape variations to the MMF under testing. The output speckle images collected at the MMF distal end for the same input DMD pattern are shown when the steel bar is at the locations of H1-H6.

Table 1. The correlation coefficient between the output speckle images for the same input DMD pattern at steel bar locations: H1 to H6 when the MMF is subject to continuous shape variations^a

	H1	H2	H3	H4	H5	H6
H1	100%	71%	60%	56%	51%	52%
H2	71%	100%	72%	56%	50%	52%
H3	60%	72%	100%	67%	50%	54%
H4	56%	56%	67%	100%	52%	53%
H5	51%	50%	50%	52%	100%	51%
H6	52%	52%	54%	53%	51%	100%

^aH1-H6 correspond to the six different steel-bar locations.

Note that in Fig. 10, six correlation coefficients at six locations of the steel bar are listed in order to show the de-correlation as a result of bending. There are much more bending states than the listed six locations. In fact, when the steel bar is manually moved continuously and

randomly at a varying speed across the 30mm region on the fiber over the 1-2 hours data acquisition, the number of bending states is a great many as well as the number of decorrelated states. This proof-of-concept experiment is sufficient to demonstrate the potential of this scheme.

We acquire 39,641 speckle images for input DMD patterns modulated with different ‘0-9’ MNIST handwritten digits. Out of these images, 35,676 speckle input images and their corresponding binary DMD labels are randomly selected to be used as our training dataset, and the remaining 3965 speckle inputs and their corresponding patterns form our test images to blindly quantify the average performance of the network. It is worth noting that the handwritten digit input DMD pattern dataset is only used once for all the shapes incurred during the steel-bar translation over the 30mm MMF. This is different from the three different MMF geometric states experiment in Fig. 6, where the whole input DMD pattern dataset is repeatedly used for three different states. This increases the difficulty in predictions, as there are less training data and all input DMD patterns in the handwritten dataset are different from each other. Figure 11 illustrates the prediction results of CNN trained by data collected when the MMF is subject to continuous shape variations. It is clear that the shapes, tilting angles and pixel-level details of the predicted digits resemble the actual input DMD patterns very well. The average prediction accuracy is 96.48% over 3965 testing images.































Predicted patterns										
Predicted binary patterns										
Ground truth										
Accuracy	97.15%	99.69%	97.53%	98.77%	98.77%	98.46%	98.69%	98.92%	98.46%	99.00%

Fig. 11. CSV-trained CNN output images of predicted digits.

In order to understand the influence of the size of training dataset on the performance of the deep neural network, we randomly choose different sizes of datasets from the training dataset collected when the MMF is subject to CSV, as shown in Fig. 12. As expected, the increase of training data pairs results in stronger prediction ability. The training time of the deep neural networks for different dataset sizes is summarized in Table 2 (with the implementation configuration detailed in Appendix).

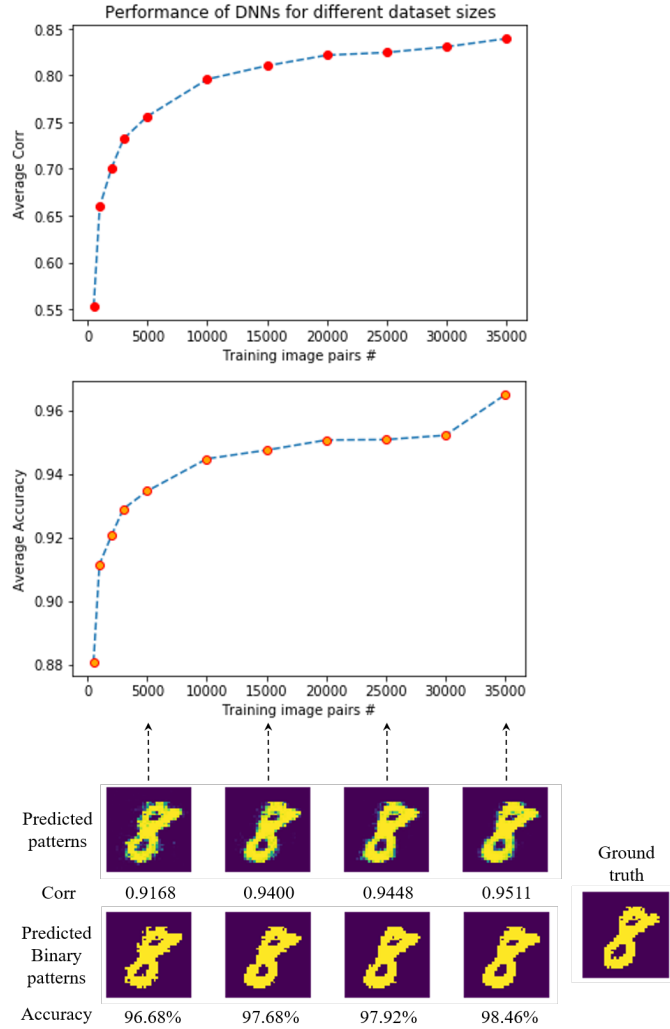


Fig. 12. The network performance at different sizes of randomly chosen training datasets (500, 1000, 2000, 3000, 5000, 10000, 15000, 20000, 25000, 30000 and 35000), tested using CSV-trained CNN and on the associated testing dataset whose size is 3,965. Insets below show the predicted patterns and their corresponding predicted binary patterns at different stages of the training. The correlation coefficient is calculated between the predicted pattern and the ground truth. On the other hand, the accuracy is calculated between the predicted binary pattern and the ground truth.

Table 2. Training time of different dataset sizes

Size of training dataset (including validation set)	Training time per epoch (sec)	Total training time (200 epoch)
5000 patches	14	47min
10000 patches	28	1hr, 34min
15000 patches	46	2hr, 33min
20000 patches	67	3hr, 43min
25000 patches	80	4hr, 27min
30000 patches	91	5hr, 03min

It is worth mentioning that all experiments conducted in this work are based on a shorter MMF. Although the changes to the MMF transmission induced by environment changes is relatively weaker and less abrupt compared to geometric shape variations studied in this work, in a longer fiber, such gradual environment factors can accumulate and lead to a strong effect, as suggested in [31].

4. Conclusion

In conclusion, our findings suggest that the high variability and randomness inside MMFs can be overcome by training a deep neural network with the possible variations that may occur to a certain MMF based optical system. Particularly, we demonstrate experimentally that distorted images (DMD input patterns) through an MMF subject to continuous shape variations can be recovered successfully by an appropriately trained deep CNN without knowing the exact MMF geometrical state or the TM. Our work paves the way for fully exploring the high-information capacity of MMF channels, and will lead to future new MMF-based optical systems for applications such as imaging and communications.

Appendix

Deep convolutional neural network architecture

The detailed architecture for the proposed deep neural network is depicted in Fig. 13. To infer the relationship between the output MMF speckle images and the modulated DMD patterns, the feature map dimension (i.e., the number of channels) used for the three convolutional layers is 5, 10 and 24 respectively, as illustrated in Fig. 13. This is empirically determined to balance the trade-off between the network complexity and prediction output time. The size of kernels (filters) used throughout the network's convolutional layers is 3×3 elements. The input image is mapped into five output feature maps by the first convolutional layer, which is followed by a batch normalization (BN) layer [41] and a dropout layer [42] (detailed below in this section). The first convolutional layer maps the original 96×96 pixels speckle image into five 94×94 -pixel feature maps. The second convolutional layer maps these five 94×94 -pixel feature maps to ten 46×46 -pixel output feature maps, as detailed in Fig. 13. This approximately quarters the size of the feature maps by utilizing a two-pixel stride convolution, while a single pixel stride convolution is performed in the other two convolutional layers in the network. The third convolutional layer maps these ten 46×46 pixels feature maps into 24 44×44 -pixel feature maps.

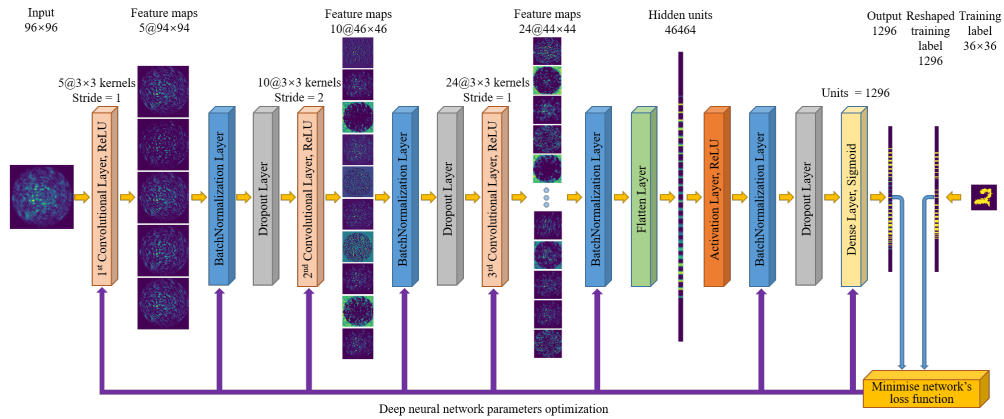


Fig. 13. The proposed deep neural network architecture.

Each convolutional layer consists of a convolution operation (determined by the learnable kernels and their biases) and an activation function (here ReLU is used), as shown in Fig. 13.

The ReLU is an activation function and performs the calculation $\text{ReLU}(x) = \max(0, x)$. The mathematic expression for each convolutional layer can be summarized as:

$$X_{out} = \text{ReLU}(X_{in} * W + B), \quad (1)$$

where X_{in} refers to the input to the convolutional layer, $*$ denotes the convolution operation, W denotes an ensemble of learnable convolution kernels, B refers to the bias, X_{out} is the output from the convolutional layer. The output feature maps from the convolutional layer in the network are expressed as:

$$g_i = \sum_j f_j * w_{i,j} + \beta_i \Omega, \quad (2)$$

where $w_{i,j}$ is a learnable 2D kernel (i.e., the (i,j) -th kernel of W) applied to the j -th input feature map f_j , β_i is a learnable bias term, g_i is the i -th $M \times M$ -pixel output feature map, and Ω is an $M \times M$ -pixel all-ones matrix. Note that the maximum values of i and j correspond to the numbers of the output and input feature maps respectively.

The three convolutional layers allow better data representations by extracting multiple high-level features from images. These extracted high-level features can be combined cheaply by adding a fully-connected layer to learn a non-linear function in the feature space effectively. To be specific, another BN layer is followed by a flatten layer. This flatten layer yields $24 \times 44 \times 44 = 46,464$ one-dimensional flatten units, which are subsequently activated by a ReLU activation layer. After the last BN layer and the last dropout layer, we utilize a dense layer (fully connecting the neighboring layers by a weighting value), which makes the model end-to-end trainable. The activation function used in the dense layer of our deep neural network is a sigmoid function. It has a characteristic ‘S’-shaped curve bounding the output to the 0-1 range (which matches our binary labels retrieval task), defined by the formula:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}. \quad (3)$$

BN layers and dropout layers are added as elemental building blocks in our deep neural network to enhance the network performance. BN layers accelerate the learning convergence, and dropout layers prevent the neural network from overfitting.

A BN layer conducts an affine transformation with the following equation:

$$y = \gamma x + \beta, \quad (4)$$

where γ and β are a pair of parameters learned for each activation in feature maps to scale and shift the normalized value. The experimental results of the four structures provided in Fig. 14 show that the BN layers enhance the network performance.

To avoid overfitting, dropout layers are essential in the building blocks of the network. In our experiment, the dropout rate was empirically determined as 0.4 for good performance. The results in Fig. 14 suggest that removing the dropout layers deteriorates the network performance significantly.

We have investigated the effect of the network architecture (the number and size of filters, the depth of network, the components of multi-branch network, etc.) on the prediction performance in the preliminary experiments, which show that the existing modest network architecture is sufficient. Additionally, we have also implemented a more sophisticated encoder-decoder ‘U-net’ architecture with dense blocks inspired by [27, 43], which does not solve the domain adaptation issue mentioned in Section 3.2 in the main text but is very time-

consuming. The deep neural network architecture discussed-above provides three major benefits: first, the inverse problem becomes a learnable operation with supervised learning; second, the fewer convolutional operations with less parameters enable faster training/predicting and outstanding representation power of the network; and third, the appropriate use of BN layers and dropout layers maximizes the capability of the network, speeds up the convergence and prevents the network from overfitting. All these lead to the excellent performance of the deep neural network and its significant generalization to the complex light propagation process inside MMFs.

Data pre-processing

For experimentally collected speckle-image training samples, the intensity of some transmitted light would inevitably overstep the dynamic range of the camera sensor resulting in saturated data for the training and testing phase of the deep neural network. Saturated data elimination is performed on each group dataset to further refine the data validity. Here, those images are removed from the dataset if they fall into the following two categories: i) No pixel in the image has an intensity value greater than 10,000; or ii) one or more pixels in the image have intensity values greater than 65,000. Thresholds for data elimination are empirically determined by the intensity dynamic range (0-65,535) of the camera (C1140-22CU, Flash4.0, Hamamatsu) to provide the optimal balance between high learning quality and full information preservation. The percentage of removed saturated data is usually less than 5% of the collected dataset.

After data pre-processing, this intensity-only speckle image consisting of 96×96 pixels and its corresponding binary pattern with 36×36 pixels (reshaped to 1296 pixels), form an input-label pair, which is used for the network's training and testing.

Network training

The designed network architecture determines the total number of its parameters. The network training phase is to optimize these parameters. The details (including parameters) of our deep neural network are provided in Table 3. The total number of parameters in the network is 60,407,346, consisting of 60,314,340 trainable params and 93,006 non-trainable params. The non-trainable weights, namely the mean and the variance vectors, are maintained by BN layers and updated via layer updates but not through the back propagation [41].

Table 3. Details of the deep neural network

Layer (type)	Output Shape	Param #
Conv2D_1	(94, 94, 5)	50
BN_1	(94, 94, 5)	20
Dropout_1	(94, 94, 5)	0
Conv2D_2	(46, 46, 10)	460
BN_2	(46, 46, 10)	40
Dropout_2	(46, 46, 10)	0
Conv2D_3	(44, 44, 24)	2184
BN_3	(44, 44, 24)	96
Flatten_1	46,464	0
Activation_1	46,464	0
BN_4	46,464	185,856
Dropout_3	46,464	0
Dense_1	1296	60,218,640
Total params: 60,407,346		

Trainable params: 60,314,340

Non-trainable params: 93,006

As shown in Fig. 13, we use sigmoid as the activation function to predict the final binary patterns in the output layer of the network. In fact, it is useful to consider a sigmoid output layer with Binary Cross Entropy cost rather than other quadratic cost functions (such as MSE). It has been proved that the quadratic cost functions will slow down the training phase during error backward propagation, while cross entropy overcomes this issue by dynamically controlling the speed of learning with the output changing errors [44].

The Binary Cross Entropy cost function, however, led to a disappointing performance in a preliminary experiment as a result of the imbalanced classification problem [45], where the classes of the data are not represented equally. In our case, the handwritten digits dataset is an imbalanced dataset and the ratio of positive targets (i.e., ‘1’s in the binary pattern) to all pixels is only 10% to 20%. That is to say, one handwritten digit just occupies a small part of the entire 36×36 pixels. The initial model is highly possible to predict negative targets regardless of the data. Therefore, we introduce adaptively weighted binary cross entropy as the loss function for training the network to deal with highly imbalanced data.

The usual cross-entropy cost can be expressed as:

$$L = - \frac{1}{n_{bs}} \sum_{i=1}^{n_{bs}} \sum_{j=1}^N (t_{i,j} \log (\text{sigmoid}(\mathbf{I}_i, j)) + (1-t_{i,j}) \log (1-\text{sigmoid}(\mathbf{I}_i, j))) \quad (5)$$

where $t_{i,j}$ and (\mathbf{I}_i, j) define the expected target (label) and the logit of the predicted output on the j -th pixel value of the DMD pattern corresponding to the training image \mathbf{I}_i , i.e. $\mathbf{t}_i = [t_{i,1}, \dots, t_{i,N}]$. N is the macro pixels of DMD pattern, and n_{bs} denotes the batch size. In implementation, targets and logits must have the same type and shape.

We use a value called *pos_weight*, which tunes the trade-off between the two metrics for classification tasks, namely ‘recall’ and ‘precision’, by up- or down-weighting the cost of a positive error relative to a negative error. This can be seen from the fact that *pos_weight* is introduced as a multiplicative coefficient for the positive targets term in the loss expression:

$$L_{pw} = - \frac{1}{n_{bs}} \sum_{i=1}^{n_{bs}} \sum_{j=1}^N (t_{i,j} \log (\text{sigmoid}(\mathbf{I}_i, j)) \text{pos_weight} + (1-t_{i,j}) \log (1-\text{sigmoid}(\mathbf{I}_i, j))) \quad (6)$$

The argument *pos_weight* needs to be tuned. For example, if we have the ratio of positives to negatives, p/n , we set *pos_weight* to be n/p , such that both categories contribute equally in total. In our implementation, the mean value of the ratio in all binary patterns from each experiment is calculated separately according to the datasets, and is assigned to *pos_weight*. The rebalanced classes provide a rational basis to determine the threshold of binary processing. This threshold used in our implementation is 0.5.

Following network’s loss function minimization, the error between the output pattern and its corresponding ground truth is backpropagated through the network, and a stochastic optimization method called the *AdaDelta* optimization [46] is used to optimize the network’s parameters. In our implementation, the learning rate parameter is empirically set as 0.1, and the total batch is split to mini-batches with 32 patches each. The entries of kernels (with 3×3 elements) used in convolutional layers are initialized by using *glorot_uniform* [47], specifically,

$$w_{i,j} : \text{Uniform}\left(-\frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in} + n_{out}}}\right), \quad (7)$$

where n_{in} and n_{out} are the number of input and output channels, respectively. All the bias terms (for instance β_j) are initialized as 0.

As an example, the training and validating losses of CNNs with different structures trained by data collected when the MMF is subject to continuous shape variations are illustrated in Fig. 14. After 200 epochs, the error between the network outputs and the binary handwritten digits drops to 0.1 and converges steadily, depicted in Fig. 14(a). This suggests that the network is trained sufficiently. The testing results of this CSV-trained CNN are shown in Fig. 11 of the main text. In contrast, the results are completely unsatisfying without the dropout or BN layers, as shown in Figs. 14(b)-14(d). This demonstrates the importance of BN and dropout layers in our CNN.

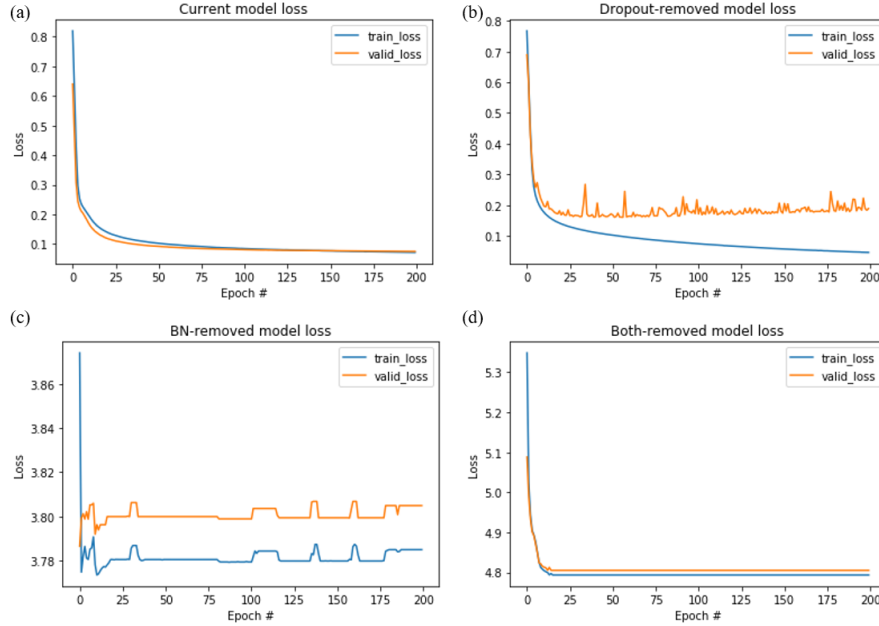


Fig. 14. Training and validation dataset errors as a function of the number of epochs for the deep neural network model in comparison with the other three structures. (a) loss curve of the proposed network's model; (b) loss curve with the dropout layers removed network's model; (c) loss curve with the BN layers removed network's model; and (d) loss curve with both BN and dropout layers removed network's model.

The training time of the deep neural networks for different image datasets mentioned in the main text is summarized in Table 4 (with the implementation configuration detailed in Implementation).

Table 4. Deep neural network training details for different datasets

Models	Number of input-output training patches	Validation set	Training time per epoch (sec)	Total training time (200 epoch)
TM-trained CNN	5265 patches	1755 images	17	56min
G3-trained CNN	24,444 patches	8148 images	92	5hr, 6min
G1&G2&G3-trained CNN	77,754 patches	25,918 images	291	16hr, 10min

CSV-trained CNN	26,757 patches	8919 images	101	5hr, 36min
-----------------	----------------	-------------	-----	------------

Networks performance evaluation

The sufficiently trained CNN is capable of predicting the input DMD pattern, by generating a 36×36 -pixel binary image based on a 96×96 -pixel speckle image captured at the other end of the MMF at any of these bending states. Testing images are used to numerically quantify the performance of our trained CNN models. From the perspective of classification evaluation, the output image of the network is quantified by using the accuracy and the $f1_score$, which are calculated between the predicted binary pattern and the ground truth. On the other hand, the mean squared error (MSE), the correlation coefficient (corr) and the structural similarity index (SSIM) [48] are also calculated between the predicted pattern and the ground truth from the perspective of regression evaluation. Details are listed in Table 5.

Table 5. Average accuracy, $f1_score$, MSE, Corr, SSIM and output runtime using a laptop CPU for different datasets

Models	Test set	Accuracy	F1_score	MSE	Corr	SSIM	Network output runtime (sec)
TM-trained CNN	780	98.74%	91.29%	0.0167	0.9222	0.8478	0.0115
G3-trained CNN	3622	97.06%	86.95%	0.0288	0.8621	0.7305	0.0117
G1&G2&G3-trained CNN	3622	96.05%	79.73%	0.0348	0.7834	0.6644	0.0114
CSV-trained CNN	3965	96.48%	83.61%	0.0313	0.8492	0.6968	0.0116

Implementation

Keras library version 2.1.5 with Python version 3.6.4 was used to implement the program. The architecture of the deep neural network was set up by using TensorFlow framework [49] back-end version 1.2.1 (Google). We utilized Queen Mary's Apocrita HPC facility to train our network's models. The training phase of the network was performed under the Linux Singularity container using a Tesla K80 GPU card (Nvidia) and a 16 Core Xeon E5-2683V3 processor (Intel) with 7.5GB RAM requested for each core. For the network's training stage, ~ 10 times speedup performance was implemented by GPU acceleration, compared to a single CPU. In the prediction phase, a laptop computer was used (Intel Core i7-7600K CPU @ 2.8GHz, 8GB of RAM, running a Microsoft Windows 10 professional operating system). The computing time for training and prediction for different datasets are available in Table 4-5.

Funding

Engineering and Physical Sciences Research Council (EP/L022559/1, EP/L022559/2); Royal Society (RG130230, IE161214).

Acknowledgments

We are thankful to the QMUL Research-IT for providing access to Queen Mary's Apocrita HPC facility. We would like to thank Fei Luo for useful discussions and support.

References

1. Y. Choi, C. Yoon, M. Kim, T. D. Yang, C. Fang-Yen, R. R. Dasari, K. J. Lee, and W. Choi, "Scanner-free and wide-field endoscopic imaging by using a single multimode optical fiber," *Phys. Rev. Lett.* **109**, 203901 (2012).
2. T. Cizmar and K. Dholakia, "Exploiting multimode waveguides for pure fibre-based imaging," *Nat. Commun.* **3**, 1027 (2012).
3. I. T. Leite, S. Turtaev, X. Jiang, M. Siler, A. Cuschieri, P. S. Russell, and T. Cizmar, "Three-dimensional holographic optical manipulation through a high-numerical-aperture soft-glass multimode fibre," *Nat. Photonics.* **12**, 33-+ (2018).

4. I. N. Papadopoulos, S. Farahi, C. Moser, and D. Psaltis, "Focusing and scanning light through a multimode optical fiber using digital phase conjugation," *Opt. Express* **20**, 10583-10590 (2012).
5. I. N. Papadopoulos, S. Farahi, C. Moser, and D. Psaltis, "High-resolution, lensless endoscope based on digital scanning through a multimode optical fiber," *Biomed. Opt. Express* **4**, 260-270 (2013).
6. B. Redding and H. Cao, "Using a multimode fiber as a high-resolution, low-loss spectrometer," *Opt. Lett.* **37**, 3384-3386 (2012).
7. D. J. Richardson, J. M. Fini, and L. E. Nelson, "Space-division multiplexing in optical fibres," *Nat. Photonics* **7**, 354-362 (2013).
8. O. Tzang, A. M. Caravaca-Aguirre, K. Wagner, and R. Piestun, "Adaptive wavefront shaping for controlling nonlinear multimode interactions in optical fibres," *Nat. Photonics* **12**, 368-+ (2018).
9. L. Raddatz, I. H. White, D. G. Cunningham, and M. C. Nowell, "An experimental and theoretical study of the offset launch technique for the enhancement of the bandwidth of multimode fiber links," *Journal of Lightwave Technology* **16**, 324-331 (1998).
10. K. Krupa, A. Tonello, B. M. Shalaby, M. Fabert, A. Barthelemy, G. Millot, S. Wabnitz, and V. Couderc, "Spatial beam self-cleaning in multimode fibres," *Nat. Photonics* **11**, 237-U299 (2017).
11. M. Ploschner, T. Tyc, and T. Cizmar, "Seeing through chaos in multimode fibres," *Nat. Photonics* **9**, 529-+ (2015).
12. L. G. Wright, D. N. Christodoulides, and F. W. Wise, "Controllable spatiotemporal nonlinear effects in multimode fibres," *Nat. Photonics* **9**, 306-310 (2015).
13. D. Gloge, "Optical Power Flow in Multimode Fibers," *At&T Tech. J.* **51**, 1767-+ (1972).
14. S. Fan and J. M. Kahn, "Principal modes in multimode waveguides," *Opt. Lett.* **30**, 135-137 (2005).
15. H. R. Stuart, "Dispersive multiplexing in multimode optical fiber," *Science* **289**, 281-283 (2000).
16. R. Olshansky, "Mode Coupling Effects in Graded-index Optical Fibers," *Appl. Opt.* **14**, 935-945 (1975).
17. L. Su, K. S. Chiang, and C. Lu, "Microbend-induced mode coupling in a graded-index multimode fiber," *Appl. Opt.* **44**, 7394-7402 (2005).
18. L. G. Wright, Z. W. Liu, D. A. Nolan, M. J. Li, D. N. Christodoulides, and F. W. Wise, "Self-organized instability in graded-index multimode fibres," *Nat. Photonics* **10**, 771-+ (2016).
19. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature* **521**, 436-444 (2015).
20. J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Netw.* **61**, 85-117 (2015).
21. Y. Rivenson, Z. Gorocs, H. Gunaydin, Y. B. Zhang, H. D. Wang, and A. Ozcan, "Deep learning microscopy," *Optica* **4**, 1437-1443 (2017).
22. E. Nehme, L. E. Weiss, T. Michaeli, and Y. Shechtman, "Deep-STORM: super-resolution single-molecule microscopy by deep learning," *Optica* **5**, 458-464 (2018).
23. A. A. Melnikov, H. Poulsen Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel, "Active learning machine learns to create new quantum experiments," *Proc. Natl. Acad. Sci. U S A* **115**, 1221-1226 (2018).
24. S. Li, M. Deng, J. Lee, A. Sinha, and G. Barbastathis, "Imaging through glass diffusers using densely connected convolutional networks," *Optica* **5**, 803-813 (2018).
25. X. Yuan and Y. Pu, "Parallel lensless compressive imaging via deep convolutional neural networks," *Opt. Express* **26**, 1962-1977 (2018).
26. A. Sinha, J. Lee, S. Li, and G. Barbastathis, "Lensless computational imaging through deep learning," *Optica* **4**, 1117-1125 (2017).
27. Y. Z. Li, Y. J. Xue, and L. Tian, "Deep speckle correlation: a deep learning approach toward scalable imaging through scattering media," *Optica* **5**, 1181-1190 (2018).
28. S. Lohani, E. M. Knutson, M. O'Donnell, S. D. Huver, and R. T. Glasser, "On the use of deep neural networks in optical communications," *Appl. Opt.* **57**, 4180-4190 (2018).
29. M. Lyu, H. Wang, G. Li, and G. Situ, "Exploit imaging through opaque wall via deep learning," *arXiv preprint arXiv:1708.07881* (2017).
30. P. Fan, L. Deng, and L. Su, "Light Propagation Prediction through Multimode Optical Fibers with a Deep Neural Network," in *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, (IEEE, 2018), 1080-1084.
31. N. Borhani, E. Kakkava, C. Moser, and D. Psaltis, "Learning to see through multimode fibers," *Optica* **5**, 960-966 (2018).
32. B. Rahmani, D. Loterie, G. Konstantinou, D. Psaltis, and C. Moser, "Multimode optical fiber transmission with a deep learning network," *Light Sci. Appl.* **7**, 69 (2018).
33. O. Moran, P. Caramazza, D. Faccio, and R. Murray-Smith, "Deep, complex, invertible networks for inversion of transmission effects in multimode optical fibres," in *Advances in Neural Information Processing Systems*, 2018), 3283-3294.
34. A. Turpin, I. Vishniakou, and J. D. Seelig, "Light scattering control in transmission and reflection with neural networks," *Opt. Express* **26**, 30911-30929 (2018).
35. L. Deng, J. D. Yan, D. S. Elson, and L. Su, "Characterization of an imaging multimode optical fiber using a digital micro-mirror device based single-beam system," *Opt. Express* **26**, 18436-18447 (2018).
36. T. Zhao, L. Deng, W. Wang, D. S. Elson, and L. Su, "Bayes' theorem-based binary algorithm for fast reference-less calibration of a multimode fiber," *Opt. Express* **26**, 20368-20378 (2018).

37. A. Dremeau, A. Liutkus, D. Martina, O. Katz, C. Schulke, F. Krzakala, S. Gigan, and L. Daudet, "Reference-less measurement of the transmission matrix of a highly scattering material using a DMD and phase retrieval techniques," *Opt. Express* **23**, 11898-11911 (2015).
38. D. Li, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine* **29**, 141-142 (2012).
39. S. M. Popoff, G. Lerosey, R. Carminati, M. Fink, A. C. Boccarda, and S. Gigan, "Measuring the transmission matrix in optics: an approach to the study and control of light propagation in disordered media," *Phys. Rev. Lett.* **104**, 100601 (2010).
40. G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: an extension of mnist to handwritten letters," *arXiv preprint arXiv:1702.05373* (2017).
41. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167* (2015).
42. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.* **15**, 1929-1958 (2014).
43. G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *CVPR*, 2017), 3.
44. M. A. Nielsen, *Neural networks and deep learning* (Determination press USA, 2015), Vol. 25.
45. N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis* **6**, 429-449 (2002).
46. M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701* (2012).
47. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010), 249-256.
48. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.* **13**, 600-612 (2004).
49. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and M. Isard, "Tensorflow: a system for large-scale machine learning," in *OSDI*, 2016), 265-283.