Title

Compensating the cell-induced light scattering effect in light-based bioprinting using deep learning

Journal

Biofabrication, 14(1)

ISSN

1758-5082

Authors

Guan, Jiaao
You, Shangting
Xiang, Yi
et al.

Publication Date

2022

DOI

10.1088/1758-5090/ac3b92

Peer reviewed

HHS Public Access

Author manuscript

Biofabrication. Author manuscript; available in PMC 2022 December 03.

Published in final edited form as:

Biofabrication. ; 14(1): . doi:10.1088/1758-5090/ac3b92.

Compensating the cell-induced light scattering effect in light-based bioprinting using deep learning

Jiaao Guan[1], **Shangting You**[2], **Yi Xiang**[2], **Jacob Schimelman**[2], **Jeffrey Alido**[2], **Xinyue Ma**[3], **Min Tang**[2], **Shaochen Chen**[2,*]

[1]Department of Electrical and Computer Engineering, University of California San Diego, 9500 Gilman Dr., La Jolla, CA, 92093, USA

[2]Department of NanoEngineering, University of California San Diego, 9500 Gilman Dr., La Jolla, CA, 92093, USA

[3]Division of Biological Science, University of California San Diego, 9500 Gilman Dr., La Jolla, CA, 92093, USA

Abstract

Digital light processing (DLP)-based 3D printing technology has the advantages of speed and precision comparing with other 3D printing technologies like extrusion-based 3D printing. Therefore, it is a promising biomaterial fabrication technique for tissue engineering and regenerative medicine. When printing cell-laden biomaterials, one challenge of DLP-based bioprinting is the light scattering effect of the cells in the bioink, and therefore induce unpredictable effects on the photopolymerization process. In consequence, the DLP-based bioprinting requires extra trial-and-error efforts for parameters optimization for each specific printable structure to compensate the scattering effects induced by cells, which is often difficult and time-consuming for a machine operator. Such trial-and-error style optimization for each different structure is also very wasteful for those expensive biomaterials and cell lines. Here, we use machine learning to learn from a few trial sample printings and automatically provide printer the optimal parameters to compensate the cell-induced scattering effects. We employ a deep learning method with a learning-based data augmentation which only requires a small amount of training data. After learning from the data, the algorithm can automatically generate the printer parameters to compensate the scattering effects. Our method shows strong improvement in the intra-layer printing resolution for bioprinting, which can be further extended to solve the light scattering problems in multilayer 3D bioprinting processes.

Keywords

3D bioprinting; cell printing; digital light processing; machine learning; deep learning; neural network; genetic algorithm

* chen168@eng.ucsd.edu .

Introduction

Three-dimensional (3D) bioprinting is one of the most important tools for tissue engineering, drug development, and regenerative medicine, due to its excellent ability to build 3D biomimetic tissue constructs and promising potential for printing patient-specific 3D tissues or organs.[1–4] There have been many different bioprinting systems like extrusion-based, inkjet-based, and light-based[5–8]. Because light can be precisely manipulated to induce material polymerization and solidification in micro- and even nano-scale,[9–11] light-based systems have been the most promising method for high-resolution biofabrication. Among the different 3D printing methods, the digital light processing (DLP) 3D printing method, which uses a digital micromirror device (DMD) to control the light pattern and photopolymerize the entire layer of the exposed region in the bioink, is getting more popular thanks to its fine resolution (a few micro meters) and high printing speed (a few seconds to a few minutes printing time) [7,8,12–15].

For 3D bioprinting, the printing solution (bioink) typically consists of the hydrogel prepolymer biomaterial, the photo-initiator, and the cells. The cells in the bioink will induce a strong scattering effect of the incident light during the photopolymerization process, which may disturb the light pattern and thus result in a undesired print [16,17]. Such scattering arises from refractive index mismatch between the cytoplasm and the external hydrogel environment[18,19], from the Mie scattering caused by the nucleus and organelle, and from the Rayleigh scattering caused by the macro-molecule.[20] The scattering effect will scatter light from the target location, causing reduced light exposure at target location while adding exposure to the surroundings. Therefore, the undesired surrounding location may be polymerized, and some target location may fail to polymerize due to the reduced light exposure (Fig. 3(a)).

There have been a few methods on improving the 3D printing fidelity to mitigate the scattering effect of the turbid bioink. Enhancing the material absorption by adding light absorbing species (e.g. food dye) is the most common practice to mitigate the scattering effects. [21,22] In more light-absorbing materials, light, including the scattered photons, will travel a shorter distance. However, the improvement on fabrication resolution is limited, and it also leads to a slower printing speed. A prolonged printing time can significantly reduce the cell viability. You et al. introduced a flashing photopolymerization technique to avoid the scattering effect caused by the hydrogel polymer, however, the scattering caused by cells cannot be resolved by this method[17]. Recently, machine learning algorithms were used to improve 3D printing fidelity by compensating the scattering effect, which shows a promising approach to address this cell-induced scattering problem [16].

The machine learning algorithms and especially the deep learning algorithms, branching from machine learning based on the use of deep neural network (NN), have been demonstrated for applications in many different fields[23,24]. Machine learning in general is a computer algorithm that learns patterns or rules from the given data or from interacting with a responsive environment without any prior knowledge. The deep learning algorithm uses various NN models to more effectively extract and store information. Researchers have been applying machine learning algorithms to improve the dimensional accuracy in

traditional 3D printing [25–28]. Machine learning has also been applied for 3D printing in-situ monitoring and correction [29,30]. In our previous work, NN-based deep learning method was introduced to learn the shape transformation between output structure and input design when using light scattering material for printing. Three hundred trial printings were printed on the actual 3D printer using the light scattering material, and the microscopic images of the printed structures together with their corresponding input digital masks were used to train the NN. The structure images were cropped and resized to match the location and resolution of the input masks. After training, to compensate the scattering effect, the NN was able to generate a digital mask that differs from the original designed pattern. Compare to the conventional method of using a mask that is identical to the designed pattern, using the NN generated mask helps the printer to print the pattern with higher fidelity[16].

While the previous work was applied on a photopolymer material mixed with glass microbeads mimicking a generic class of scattering payload in the printing materials, in this paper, we apply the deep learning method to cell-loaded bioprinting and show the capability of improving bioprinting quality. We 3D print different structures using various predesigned digital masks, take microscopic images of the structures, and use the mask-structure image pairs as our data set to train the NNs. The trained algorithm can generate a deformed mask for any given target structure to compensate the scattering effect of the cell-loaded bioink. Furthermore, we further improve the previous deep learning method with an additional learning step, which learns parameters from a 3D printer simulator. This simulator serves as a data augmentation tool, which allows us to greatly reduce the required training samples by 10 folds. We show that using only 32 trial printings, which got augmented to 4000 data pairs, is sufficient to train our NN. This reduction of training data requirement is very significant in bioprinting due to the high cost of biomaterials and bioreagent (such as growth factor), limited supply of cells (such as stem cells and primary cells), and the long cell culture time (weeks of culture time).

After we compared our optimized prints guided by machine learning with the conventional printing result, we can see that our deep learning method can indeed improve the printing fidelity for the highly scattering cell-loaded material.

Methods

3D printing method

Our samples are printed with a custom DLP-based 3D printer (Fig. 1). A 385 nm wavelength light source first projects the light onto a DMD chip, which contains an array of 2560 by 1600 micro-mirrors. The on-off state of each individual micro-mirror is controlled by flipping the mirror angle, and a pattern will then appear on the micro mirror array. A grayscale pattern can be displayed by controlling the duty ration of the flipping of the mirrors. The patterned light reflected from the DMD is guided by a series of lenses and projects onto the holder with the prepolymer solution. Polymerization occurs at the exposed region, and a solid thin layer of the structure forms. The motorized stage then lifts the solidified structure, normally by tens or hundreds of microns, and leaves space for the solution to refill and then start the next layer of printing. The process is repeated for all the

cross sections of an object model in order to print a 3D object. Our study is focused on the printing process of an individual layer.

A computer software controls the light exposure on the target region. The software controls the light source power, the exposure duration, the DMD mirror array pattern, and local light exposure dose on each micro mirror. A digital mask with grayscale pixel values is used to represent the DMD pattern and the local exposure dose. In this study, the light source power and the exposure duration are set to be constant. We are only controlling a square region in the center of the DMD array, which is represented by a 512-by-512-pixel mask. The rest of the DMD array is set to a fixed state that is designed to localize the center region. In this setup, our 3D printing system can be abstracted as a nonlinear time-invariant system, where the input of the system is a 512×512 grayscale image representing the digital mask, and the output is a 512×512 binary image where 0 and 1 represent void and solid region, respectively (Fig. 3(a)). The size is chosen as multiples of 8 (a byte) for efficient CPU and GPU processing, and it is also chosen to not exceed our 8GB GPU memory during the NN training process.

Our prepolymer printing solution is composed of 5% (v/v) gelatin methacryloyl (GelMA) in phosphate-buffered saline (PBS) solution, 1% (w/v) lithium phenyl-2,4,6-trimethylbenzoylphosphinate (LAP) as the photoinitiator, and 10 million/mL C2C12 mouse myoblast cells as the scattering load. The source of our C2C12 cell line was purchased from American Type Culture Collection.

The 3D printed structures are imaged with a fluorescent microscope. Due to the transparent nature of the GelMA polymer, it is hard to detect the printed structures' contour under a bright field microscope. Hence, we apply fluorescent staining to the material in order to obtain high quality images distinguishing the printed and unprinted part. We obtained the Fluorescein (FAM) NHS ester, 6-isomer from Lumiprobe (MD, USA). The FAM-labeled GelMA was synthesized in accordance to the manufacturer's general NHS ester conjugation protocol. We always wash away the residual solution after printing to avoid the false positive detection of the fluorescent signal in the residual solution. Thanks to the FAM label, the brightness of the fluorescence can directly translate to the density of the structure being polymerized. The fluorescent structure image is cropped, rotated, and resized to match the size and location of the input mask. We use the intensity on the fluorescent image as a measure of the polymerization completeness, and we take a threshold on the fluorescent image to obtain a binary image representing whether each part of the structure is properly polymerized (true, white) or not (false, black) (Fig. 3(a)).

The goal of our method is to find an optimal design mask for any printable structure, which represent the light exposure dose on every pixel location with a grayscale value, that help compensate the scattering effect of the cell-loaded material during 3D printing. Our machine learning algorithm is composed of two learning steps, the simulator calibration step for generating augmentation data, and the NN training step to generate desired grayscale masks (Fig. 2). The overall workflow of our algorithm is to first acquire image data of real 3D printed structures with a variety of sample masks. Then we use these trial data and apply generic algorithm to learn the parameters of a mathematical simulator to simulate the

3D printer with scattering effects from the cell loading. With the calibrated simulator, we generate thousands of simulated data and train a specially designed deep neural network to learn the optimal mask choice for any desired target structure. For simplicity, all our experiments were done with a single layer of printing instead of multilayer 3D structures. It is worth noting that our method can easily extend to 3D cases by upgrading the simulator and NN design, and the overall learning process remains the same. The design region is fixed to 512×512 pixels in terms of the mask size with 2.96 micrometers per pixel, which is about 1.5×1.5 mm^2 in physical size of the printing region.

Trial data acquisition

The first stage of our algorithm is to acquire real 3D printed structures (Fig. 2(a)). The goal of this stage is to collect data for analyzing the inner-layer scattering effects of cell loading. We first generate 32 predesigned masks as demonstrated in Fig. 3a with MATLAB code. These sample masks have three different types of randomized feature shapes, including random grayscale valued checkerboard shapes, randomly positioned and randomly sized rectangles, and randomly positioned variety of circular shapes. The design purpose of these masks is to provide various smooth and sharp features as represented by the circles and rectangles.

After we designed the sample masks, we use them as input into our DMD 3D printer. The grayscale value on each pixel of the mask image represents the percentage of light exposure dose of the 3D printer, which is controlled by the duty ration of the flipping of the DMD mirrors. A full valued pixel, which is 255 in 8-bit unsigned integer representation, represents a 100 percent exposure dose. The maximum light exposure dose is also controlled by the exposure time, light source power. For simplicity, we use a constant light source power and manually fix the exposure time to 20 seconds, and the maximum light intensity (at 255 grayscale value on the mask) is measured to be 20.8 mW/cm^2.

The printed structures are imaged using a fluorescence microscope. The intensity of fluorescence can translate to the completeness of polymerization. After a thresholding operation, we obtain the binary image with 0 and 1 representing void and polymerized state respectively. The processed structure images as well as their corresponding masks are then used as input to train our algorithm (Fig. 3(a)).

Simulator calibration

The data augmentation stage is the first learning step that takes the real printing data as input and generates various virtual data as output. The input data, including the sample masks and the postprocessed printed structure images, came from the previous data acquisition stage. The input data is used to calibrate a mathematical simulator that models the local and global deformation caused by the scattering effect of the cell-loaded material during the printing process. We have developed the simulation function according to the interpretation to the physical 3D printing process (Eq. 1).

$$P = Sim(M; c_0, c_1, c_2, c_3, c_4, \sigma, T) = Threshold_T\left(Gauss_\sigma\left(ReLU\left(c_1X^2 + c_2Y^2 + c_3X + c_4Y + c_0\right) \times M\right)\right) \quad (1)$$

The simulator takes a grayscale mask image M of size 512×512 as the input variable, and outputs a binary image P of the same size that shows the polymerization condition of each pixel point on the given region. A true or 1 represents fully polymerized unit, while a false or 0 represents an under-polymerized or unpolymerized unit. The mask is first element-wise multiplied by a basic light absorption matrix $ReLU(c_1X^2 + c_2Y^2 + c_3X + c_4Y + c_0)$. This matrix is composed of a linear combination of X and Y as well as their quadratics, where X is the first coordinate of each point on the mask, and Y is the second coordinate. Both X and Y are of the same dimension as M. The basic light absorption matrix serves to mimic the light emitting and absorbing process of the DMD 3D printer under a mask with no patterns on it. When it multiplies M pixelwise, the result should represent the light energy absorbed on each patterned pixel without considering the scattering effect. The basic light absorption matrix also provides the simulator the ability to mimic certain locational variant characteristics of the 3D printer, for example the light energy could be slightly stronger at the center of the exposed region compared to the side. The rectifier linear unit (ReLU) function zeros out all the negative values, since we know the light absorption is physically non-negative. After that, we have a 2D Gaussian kernel $Gauss_\sigma(\cdot)$ with standard deviation σ, that simulates the light scattering effect of the cell-loaded material. Finally, the $Threshold_T(\cdot)$ function sets a threshold T that decides under what degree of light exposure should the material be considered as being successfully polymerized. After all the function processes in the simulator, it will calculate a binary map of polymerized versus under- or unpolymerized for each unit area.

The calibration of this simulator is done by optimizing the seven parameters in Eq. 1 with genetic algorithm, which is a commonly used non-gradient optimization method in many fields[31]. The optimization objective is set to be the mean squared distance between the real printed structure and the simulator output, averaging among all 32 trial data pairs. Optimization was implemented with MATLAB and the global optimization toolbox[32].

After calibration, we apply a new set of 4,000 masks to the simulator and obtain their corresponding simulated structures. These new masks are designed with the same scheme as the sample masks with a greater variety feature numbers and sizes as well as four additional feature types, vertical lines with different spacing, horizontal ones, the combination of two, and 2D vasculature shapes. After we get all the simulation results, we proceed to use this data to train the NNs in the final stage.

Neural network training

The next step of our machine learning method is the NNs training, which is the key part of our algorithm. We are looking to train a NN that can automatically calculate the appropriate mask for any potential target structure. We are supplying the 4,000 simulated data pairs from the data augmentation stage to train the NN.

Our NN method is composed of two U-Net-like NNs which we call the master NN and the slave NN [16,33]. The slave NN learns the transformation of our physical 3D printer, which is similar to the function of the simulator in the previous stage. The major difference is that the slave NN is a differentiable function, which provides gradient information to support the training of the master NN. The master NN serves to learn the inverse transformation of the 3D printer. For any given target structure, the master NN will suggest a deformed mask that could allow the printer to print out this target structure under the highly scattering condition.

The network architecture of the master NN is shown in Fig. 4. This architecture reproduces the U-net style encoder-decoder architecture with some adaptations [33,34]. The network consists of 14 building blocks. The network takes a 512×512 single-channel image as the input. The first 7 blocks each has a convolution layer with stride 2 to down-sample the images and features, and the convolution layer is followed by a batch normalization layer and a ReLU function[35,36]. The other 7 blocks each has a deconvolution layer with stride 2 to up-sample the features and ensure the output resolution is the same as input. The deconvolution layers are again followed with batch normalization and ReLU, except the last layer uses the hyperbolic tangent function (Tanh) instead of the ReLU. The U-Net style skip connections copy the feature map from the first six block outputs to the last six blocks' input features respectively. These skip connections help the network to learn the local details in the earlier feature maps while retaining the global information extracted from the later features. The slave NN has almost the same architecture as the master NN except the final output layer of the slave NN has 2 channels for the pixel-wise classification output, which outputs the binary structure, instead of the single channel regression output of the master NN.

To train the NNs, we randomly divide the 4,000 data pairs into 3,600 pairs of training data and 400 as testing data. The testing data is used to verify the convergence of the networks. Since we already have the simulator generated data as the augmented data, we are not applying other data augmentation techniques. The training process is achieved by backpropagation of the following loss function.

$$\begin{aligned} Loss &= E_y\left[L_{Crossentropy}\left(Slave(Master(y)), y\right)\right] + \lambda_1 \\ &* E_{x,y}\left[L_{L1}(Master(y), x)\right] + \\ &\lambda_2 * E_{x,y}\left[L_{Crossentropy}\left(Slave(x), y\right)\right] + \lambda_3 * \ Sparsity_y(Master(y)) \end{aligned} \tag{2}$$

In Eq. 2, the loss function is the sum of four loss terms, the slave supported master loss, the data supported master loss, the data supported slave loss, and the sparsity loss. The details of the first three loss terms can be found in the Appendix section of [16]. The x here stands for the grayscale mask, which is rescaled to a range between −1 and 1, and y stands for the 3D printer output structure, which is represented in a binary class map with two channels. E represents the expectation operator. The $L_{Crossentropy}$ calculates the cross-entropy loss between the two arguments, and L_{L1} is the $L1$ loss function or the least absolute deviations. $Master$ is the master NN forward pass, and the $Slave$ is the slave NN forward pass. The $Sparsity$ is a sparsity loss term that sums the pixel grayscale values of a given region. In this case, the sparsity loss sums up the pixel values in the master NN generated mask, where it

is outside of the target structure region of y. This term ensures that the NN generated mask does not give unwanted exposure far away from the target structure region. λ_n is the tradeoff coefficient for different loss terms. We are setting the tradeoff coefficients λ_1, λ_2, λ_3 to be 1, 1, and 0.05 respectively. We also decay the λ_1 term with factor 0.98 on every training iteration, so that the master NN relies on the training data initially, and it will gradually rely more on the slave NN gradients in the later epochs when the slave NN becomes more trained. This will improve the generalizability of the master NN by promoting it to spend more time on learning the 3D printer transformation instead of repeatedly looking at the training data.

In terms of training implementation, we used the PyTorch framework GPU version 1.2 with Adam solver for the back-propagation process [37,38]. We set the batch size to 10, learning rate to 10^{-5}, gradient decay factor to 0.9, and the squared gradient decay factor to 0.999. A Gaussian noise term is added to the input x to avoid overfitting of slave NN, while the training error of slave NN itself could help avoid the master NN from overfitting[39]. The initialization of model weights is done by sampling from normal distribution with zero mean and 0.02 standard deviation. The training was executed on a desktop computer with Intel i5–7500 CPU and GTX 1070Ti GPU. We trained the networks for 200 epochs for about 26 hours. After the network is trained, it will only require a few seconds on a CPU only machine to execute a forward pass of the master NN, while the GPU enabled machine can process the forward pass in less than a second.

Results

3D printing with NN-generated masks

To verify our trained NN, we set several testing designs that are unseen from the training data (Fig. 5(a)) (Fig. 6). The test structure designs mainly demonstrate various concave and convex sharp features as well as the ring shape that could potentially be used to print biological tissue models. We then input these target structures one-by-one to the master NN and get their corresponding NN-calculated masks as the output (Fig. 2(c)) (Fig. 5(b)).

The NN-calculated masks are indeed different from the target structure. We can interpret that the NN-calculated mask tends to “stretch out” at the protruded sharp regions and “shrink” at the denting regions (Fig. 5(e)). The overall behavior of how the NN tries to compensate the scattering effect is similar to our previous research[16]. It is important to note that these kind of mask designs are usually not possible even for an experienced expert in 3D printing.

After we obtain all the NN-calculated masks, we apply each mask to the DMD 3D printer and take microscopic fluorescent images as well as postprocess the images into binary representation, in the same way as we did in preparing the trial data (Fig. 5(c–d)). The resulting binary images matches nicely to the input target structures.

Printing quality comparison

To better demonstrate the power of our method, we compare the printing results between using the NN-calculated masks and using the traditional identical masks. Traditionally, an operator of the DMD 3D printer usually sets the DMD mask identical to the target

design and only changes the overall light exposure dose by tuning the light power and print time. Therefore, we use the 100%, 75%, 70%, and 50% grayscale mask values to mimic the overall light exposure changes of the traditional tuning. Later experiment shows that grayscale value of lower than 50% will hardly print any obvious structures, therefore, we choose 70% as the lower bound of intensity dose (Fig. 6). The printing results of these identical masks are then used as benchmark printing samples for comparison with the printing result of NN-calculated masks.

From the results in Fig. 6, we can easily see that the NN-calculated masks perform better than the traditional masks across all the testing target structures we had. In the first three rows, we can see that the NN results greatly reserves the sharp features of the target, while the benchmark results always lost the sharpness to the rounded smooth features. For the ring shapes in the bottom two rows, we find that although similar quality has shown on the bigger rings across the different masks, the smaller rings can only be properly printed with the NN-calculated masks. The ring shape with varying diameters shows an example that different patterns would require different manual tuning of the printer settings, since the varying rings under the same printing condition with identical masks would never show the same printing quality. With our machine learning method, we can see that different rings have a more consistent quality. Comparing the smooth ring patterns with the patterns with sharp features, we can see that the smooth features are easier to print even with the traditional identical mask method, while the more complex features can only be properly printed with NN masks to reserve the fine features. The size of the overall patterns also effects the print quality. From the various ring shapes, we can find that the large features are always easier to print, and the small features are very challenging to print under the high scattering effect even with the help of NN (Fig. 6).

Discussion

As 3D bioprinting attracts more and more industrial applications, key challenges emerge: how to improve printing fidelity when cell-induced light scattering effects dominate? How to minimize the traditional trial and error operation in optimizing the printing parameters? As shown in our results, the NN-based deep learning method demonstrate great success on advising the 3D bioprinter by providing the optimal optical masks to improve the printing fidelity when scattering cells present.

One major benefit of our method is the high data efficiency. Common machine learning algorithms would require a large amount of data (i.e. printed samples) in order to get reliable performance. In our algorithm, we are only using 32 actual printing samples as training data, which is an extremely small number compared to the common machine learning data sets that have thousands or even hundreds of thousands of samples. Even with such a small number of trial data, our trained NN showed its power to greatly improve the printing quality. Considering that the 3D bioprinting is normally very expensive to produce many samples due to the cost of the cells, bioinks, and cell cultures, our method is more practically applicable than the normal data heavy deep learning methods. If we are accessible to more printing data, our method can still be applied, and the resulting accuracy could be improved

according to the central limit theorem considering the random selection of initial masks and the potential production and imaging noises.

The way we utilize the limited amount of data is to introduce a simple equation, being calibrated with the trial data, to simulate the process of the 3D printer and the scattering effect. Although different from the common data augmentation methods[40], our approach of calibrated simulations can be seen as a heuristic way of data augmentation. By incorporating the understanding of the physical processes of the DMD 3D printing, our simulator only considers seven parameters. Compare to the basic image manipulation type of augmentations like flipping or rotation, our augmentation method shows many more different features by introducing a great variety of new masks than the sample masks in the simulation, which prevents the further overfitting that might be caused with simple augmentations[40].

Beside the calibrated simulation, our slave part of the NN can also be considered a kind of data augmentation, since it predicts the output structure for the input of master NN generated mask which can be potentially unseen from the training data. The slave NN can also be interpreted as a noise term that will gradually decrease with the training process, since the randomly initialized slave NN can produce very wrong prediction of the output structure at the beginning and improves through training. Adding noise to the NN training process is a known technique to improve the generalization performance of the trained NN[41]. The benefit of the slave NN has been experimentally studied in our previous paper[16].

Although our current experiment is on a single-layer basis, our method can easily be extended to apply on multilayer 3D structures. The only changes in the algorithm will be that the 2D convolution layers in the NNs will be replaced by 3D convolutions, and the simulation function will need to add an extra dimension. The difficulty we foresee is the way to properly image or scan the printed 3D structure. Also, the size of the data we are going to process will be a lot larger if we want to keep the resolution of each dimension. The huge data size could cause trouble in NN training.

Our current experiment is working on a custom DMD based 3D printer, and we are only printing with one specific material and cell composition. However, our method could easily adapt to a different printer or material composition with a new set of sample prints. Our requirement of 32 data samples is relatively easy to obtain comparing to some other deep learning methods that relies on big data[42]. Further online training to improve the performance and the generalizability across different settings would be an interesting future improvement.

Conclusion

Our deep learning method with learning-based data augmentation greatly improves the fidelity of bioprinting with as few as 32 sample prints to train the learning system. Our experiment shows that using the grayscale masks generated from our trained NN, we can print fine detailed structures surpassing the traditional manual tuning method with identical masks. Our method allows the use of a very small amount of trial data, which is usually not

possible for the common deep learning applications that relies on big dataset. Furthermore, our method could easily be applied on different materials or different printer settings with a new set of sample prints. Considering the high cost of cells, reagent, and bioinks, our deep learning method provides a powerful solution for bioprinting with reduced cost, high fidelity, and shorter time to product, paving the way for future large scale organ printing.

Acknowledgements

This work was supported in part by National Institutes of Health (R21AR074763, R33HD090662, R21HD100132) and National Science Foundation (1907434, 1937653). Part of the work is performed at San Diego Nanotechnology Infrastructure (SDNI) of UCSD, a member of the National Nanotechnology Coordinated Infrastructure (NNCI), which is supported by NSF (Grant ECCS-1542148). This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1650112.

References

[1]. Berry DB, You S, Warner J, Frank LR, Chen S, Ward SR, A 3D Tissue-Printing Approach for Validation of Diffusion Tensor Imaging in Skeletal Muscle, Tissue Engineering Part A. 23 (2017) 980–988. 10.1089/ten.tea.2016.0438. [PubMed: 28338417]

[2]. Ma X, Qu X, Zhu W, Li Y-S, Yuan S, Zhang H, Liu J, Wang P, Lai CSE, Zanella F, Feng G-S, Sheikh F, Chien S, Chen S, Deterministically patterned biomimetic human iPSC-derived hepatic model via rapid 3D bioprinting, Proc Natl Acad Sci USA. 113 (2016) 2206–2211. 10.1073/pnas.1524510113. [PubMed: 26858399]

[3]. Hwang HH, You S, Ma X, Kwe L, Victorine G, Lawrence N, Wan X, Shen H, Zhu W, Chen S, High throughput direct 3D bioprinting in multiwell plates, Biofabrication. (2020). 10.1088/1758-5090/ab89ca.

[4]. Zhu W, Tringale KR, Woller SA, You S, Johnson S, Shen H, Schimelman J, Whitney M, Steinauer J, Xu W, Yaksh TL, Nguyen QT, Chen S, Rapid continuous 3D printing of customizable peripheral nerve guidance conduits, Materials Today. 21 (2018) 951–959. 10.1016/j.mattod.2018.04.001. [PubMed: 31156331]

[5]. Seol Y-J, Kang H-W, Lee SJ, Atala A, Yoo JJ, Bioprinting technology and its applications, European Journal of Cardio-Thoracic Surgery. 46 (2014) 342–348. 10.1093/ejcts/ezu148. [PubMed: 25061217]

[6]. Ozbolat IT, Hospodiuk M, Current advances and future perspectives in extrusion-based bioprinting, Biomaterials. 76 (2016) 321–343. 10.1016/j.biomaterials.2015.10.076. [PubMed: 26561931]

[7]. Dababneh AB, Ozbolat IT, Bioprinting Technology: A Current State-of-the-Art Review, Journal of Manufacturing Science and Engineering. 136 (2014). 10.1115/1.4028512.

[8]. Derakhshanfar S, Mbeleck R, Xu K, Zhang X, Zhong W, Xing M, 3D bioprinting for biomedical devices and tissue engineering: A review of recent trends and advances, Bioactive Materials. 3 (2018) 144–156. 10.1016/j.bioactmat.2017.11.008. [PubMed: 29744452]

[9]. You S, Li J, Zhu W, Yu C, Mei D, Chen S, Nanoscale 3D printing of hydrogels for cellular tissue engineering, J. Mater. Chem. B. 6 (2018) 2187–2197. 10.1039/C8TB00301G. [PubMed: 30319779]

[10]. You S, Zhu W, Wang P, Chen S, Projection Printing of Ultrathin Structures with Nanoscale Thickness Control, ACS Applied Materials & Interfaces. 11 (2019) 16059–16064. 10.1021/acsami.9b02728. [PubMed: 30964636]

[11]. You S, Miller K, Chen S, Chapter 1. Microstereolithography, in: Cho D-W (Ed.), Biomaterials Science Series, Royal Society of Chemistry, Cambridge, 2019: pp. 1–21. 10.1039/9781788012683-00001.

[12]. Yu C, Schimelman J, Wang P, Miller KL, Ma X, You S, Guan J, Sun B, Zhu W, Chen S, Photopolymerizable Biomaterials and Light-Based 3D Printing Strategies for Biomedical Applications, Chem. Rev. 120 (2020) 10695–10743. 10.1021/acs.chemrev.9b00810. [PubMed: 32323975]

[13]. Zhu W, Ma X, Gou M, Mei D, Zhang K, Chen S, 3D printing of functional biomaterials for tissue engineering, Current Opinion in Biotechnology. 40 (2016) 103–112. 10.1016/j.copbio.2016.03.014. [PubMed: 27043763]

[14]. Lin M, Firoozi N, Tsai C-T, Wallace MB, Kang Y, 3D-printed flexible polymer stents for potential applications in inoperable esophageal malignancies, Acta Biomaterialia. 83 (2019) 119–129. 10.1016/j.actbio.2018.10.035. [PubMed: 30366130]

[15]. Lin M, Vatani M, Choi J-W, Dilibal S, Engeberg ED, Compliant underwater manipulator with integrated tactile sensor for nonlinear force feedback control of an SMA actuation system, Sensors and Actuators A: Physical. 315 (2020) 112221. 10.1016/j.sna.2020.112221. [PubMed: 34629752]

[16]. You S, Guan J, Alido J, Hwang HH, Yu R, Kwe L, Su H, Chen S, Mitigating Scattering Effects in Light-Based Three-Dimensional Printing Using Machine Learning, Journal of Manufacturing Science and Engineering. 142 (2020) 081002. 10.1115/1.4046986.

[17]. You S, Wang P, Schimelman J, Hwang HH, Chen S, High-fidelity 3D printing using flashing photopolymerization, Additive Manufacturing. 30 (2019) 100834. 10.1016/j.addma.2019.100834. [PubMed: 32832382]

[18]. Choi W, Fang-Yen C, Badizadegan K, Oh S, Lue N, Dasari RR, Feld MS, Tomographic phase microscopy, Nat Methods. 4 (2007) 717–719. 10.1038/nmeth1078. [PubMed: 17694065]

[19]. Schürmann M, Scholze J, Müller P, Guck J, Chan CJ, Cell nuclei have lower refractive index and mass density than cytoplasm, J. Biophoton. 9 (2016) 1068–1076. 10.1002/jbio.201500273.

[20]. Tuchin VV, Tissue Optics: Light Scattering Methods and Instruments for Medical Diagnosis, Society of Photo-Optical Instrumentation Engineers (SPIE), 2015. 10.1117/3.1003040.

[21]. Han L-H, Mapili G, Chen S, Roy K, Projection Microfabrication of Three-Dimensional Scaffolds for Tissue Engineering, Journal of Manufacturing Science and Engineering. 130 (2008). 10.1115/1.2823079.

[22]. Grigoryan B, Paulsen SJ, Corbett DC, Sazer DW, Fortin CL, Zaita AJ, Greenfield PT, Calafat NJ, Gounley JP, Ta AH, Johansson F, Randles A, Rosenkrantz JE, Louis-Rosenberg JD, Galie PA, Stevens KR, Miller JS, Multivascular networks and functional intravascular topologies within biocompatible hydrogels, Science. 364 (2019) 458–464. 10.1126/science.aav9750. [PubMed: 31048486]

[23]. Angra S, Ahuja S, Machine learning and its applications: A review, in: 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), 2017: pp. 57–60. 10.1109/ICBDACI.2017.8070809.

[24]. Shinde PP, Shah S, A Review of Machine Learning and Deep Learning Applications, in: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018: pp. 1–6. 10.1109/ICCUBEA.2018.8697857.

[25]. Baumann FW, Sekulla A, Hassler M, Himpel B, Pfeil M, Trends of machine learning in additive manufacturing, International Journal of Rapid Manufacturing. 7 (2018) 310. 10.1504/IJRAPIDM.2018.095788.

[26]. Meng L, McWilliams B, Jarosinski W, Park H-Y, Jung Y-G, Lee J, Zhang J, Machine Learning in Additive Manufacturing: A Review, JOM. 72 (2020) 2363–2377. 10.1007/s11837-020-04155-y.

[27]. Razvi SS, Feng S, Narayanan A, Lee Y-TT, Witherell P, A Review of Machine Learning Applications in Additive Manufacturing, in: American Society of Mechanical Engineers Digital Collection, 2019. 10.1115/DETC2019-98415.

[28]. Qi X, Chen G, Li Y, Cheng X, Li C, Applying Neural-Network-Based Machine Learning to Additive Manufacturing: Current Applications, Challenges, and Future Perspectives, Engineering. 5 (2019) 721–729. 10.1016/j.eng.2019.04.012.

[29]. Jin Z, Zhang Z, Gu GX, Autonomous in-situ correction of fused deposition modeling printers using computer vision and deep learning, Manufacturing Letters. 22 (2019) 11–15. 10.1016/j.mfglet.2019.09.005.

[30]. Jin Z, Zhang Z, Shao X, Gu GX, Monitoring Anomalies in 3D Bioprinting with Deep Neural Networks, ACS Biomater. Sci. Eng. (2021). 10.1021/acsbiomaterials.0c01761.

[31]. Goldberg DE, Genetic algorithms in search, optimization, and machine learning, Addison-Wesley Pub. Co, Reading, Mass, 1989.

[32]. MATLAB R2021a and Global Optimization Toolbox 4.5, The Mathworks Inc., Natick, Massachusetts, United States, 2021.

[33]. Ronneberger O, Fischer P, Brox T, U-Net: Convolutional Networks for Biomedical Image Segmentation, in: Navab N, Hornegger J, Wells WM, Frangi AF (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer International Publishing, Cham, 2015: pp. 234–241. 10.1007/978-3-319-24574-4_28.

[34]. Isola P, Zhu J-Y, Zhou T, Efros AA, Image-to-Image Translation with Conditional Adversarial Networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Honolulu, HI, 2017: pp. 5967–5976. 10.1109/CVPR.2017.632.

[35]. Ioffe S, Szegedy C, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, JMLR.org, Lille, France, 2015: pp. 448–456.

[36]. Nair V, Hinton GE, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 2010: pp. 807–814.

[37]. Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A, Automatic differentiation in PyTorch, (2017).

[38]. Kingma DP, Ba J, Adam: A Method for Stochastic Optimization, in: Bengio Y, LeCun Y (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings, 2015. http://arxiv.org/abs/1412.6980 (accessed April 26, 2020).

[39]. Bishop CM, Neural networks for pattern recognition, Oxford university press, 1995.

[40]. Shorten C, Khoshgoftaar TM, A survey on Image Data Augmentation for Deep Learning, Journal of Big Data. 6 (2019) 60. 10.1186/s40537-019-0197-0.

[41]. An G, The Effects of Adding Noise During Backpropagation Training on a Generalization Performance, Neural Computation. 8 (1996) 643–674. 10.1162/neco.1996.8.3.643.

[42]. Deng J, Dong W, Socher R, Li L-J, Li Kai, Fei-Fei Li, ImageNet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009: pp. 248–255. 10.1109/CVPR.2009.5206848.

Significance Statement

This work applies a deep learning method with learning-based data augmentation to improve printing fidelity in Digital Light Processing (DLP)-based 3D bioprinting using cell-loaded biomaterials, where cell-induced light scattering tends to decelerate the printing quality. The deep learning algorithm is able to learn the scattering behavior of the bioink and automatically generate a digital mask to compensate this light scattering effect. The learning-based data augmentation method is shown to greatly reduce the required sample prints. Experimental results show that the printing fidelity has been significantly improved through machine learning.

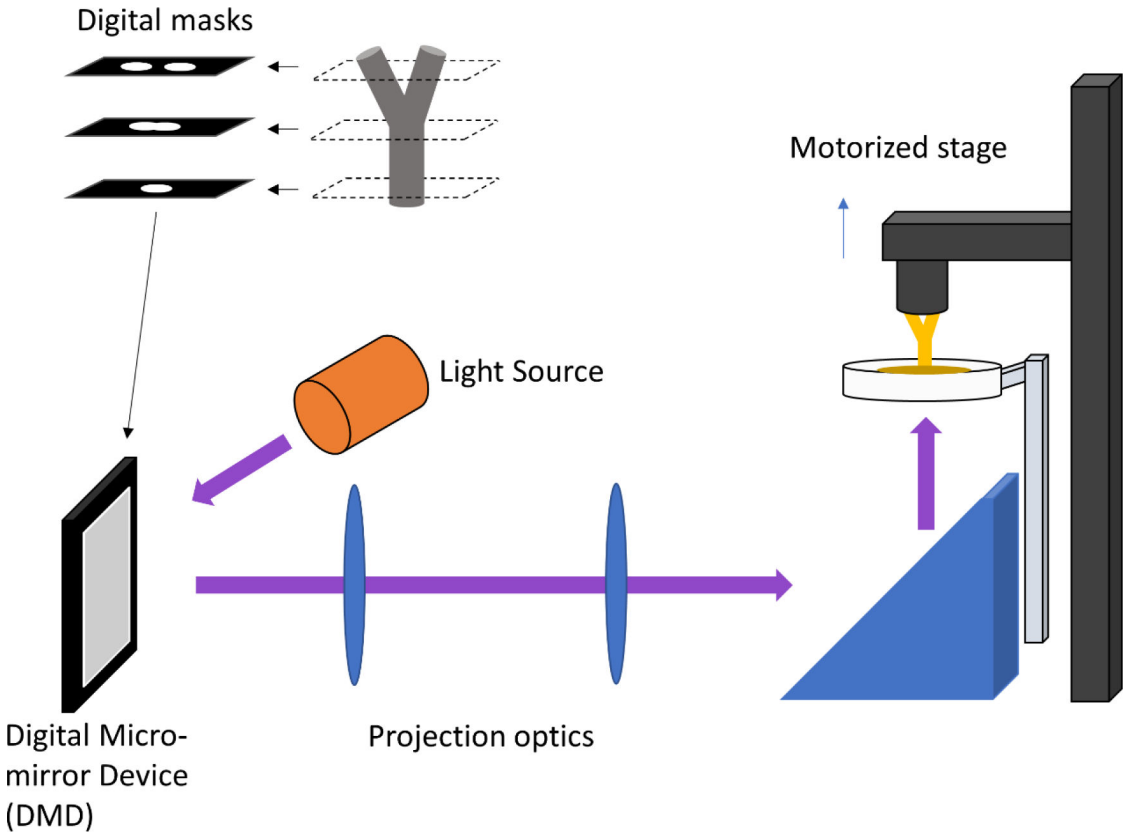

Fig. 1.
Schematic of the DLP-based 3D bioprinting setup.

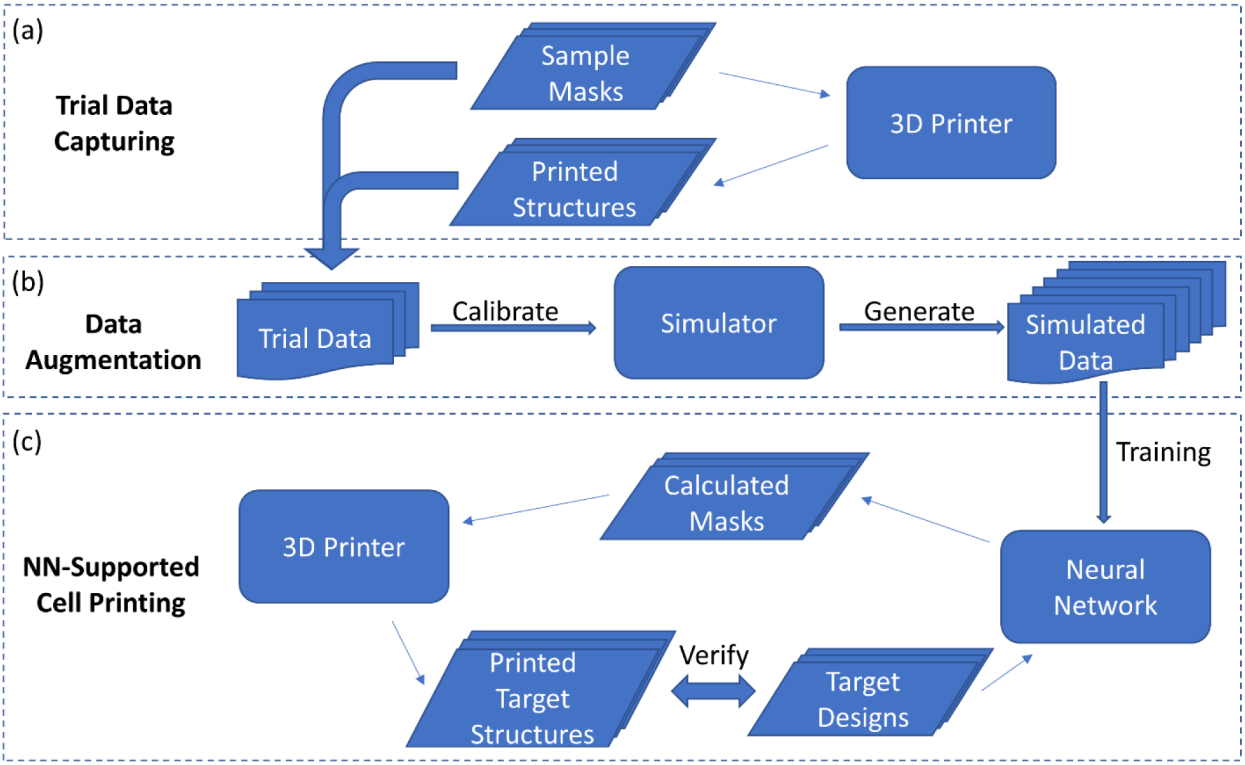

Fig. 2.

Data Flow and schematic of the learning process. (a) 32 sample masks are used to print 32 single-layer trial prints for training the algorithm. (b) The trial prints calibrate the printer simulator, and the calibrated simulator generates thousands of new training data. (c) The printed samples and the simulator generated samples are used to train the neural network, the trained network calculates the appropriate masks that compensate the cell scattering effect, and the final print quality is tested with some predefined target designs.

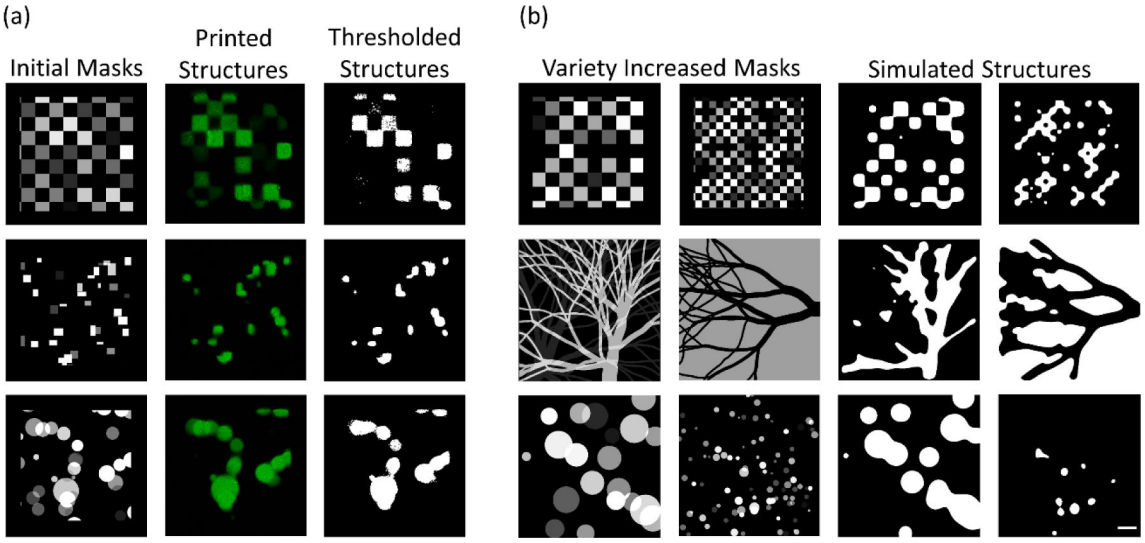

Fig. 3.

Examples of the calibration and training data. (a) The calibration data for the simulator. (b) The training data generated from the calibrated simulator for NN training. Scale bar is 200 μm.

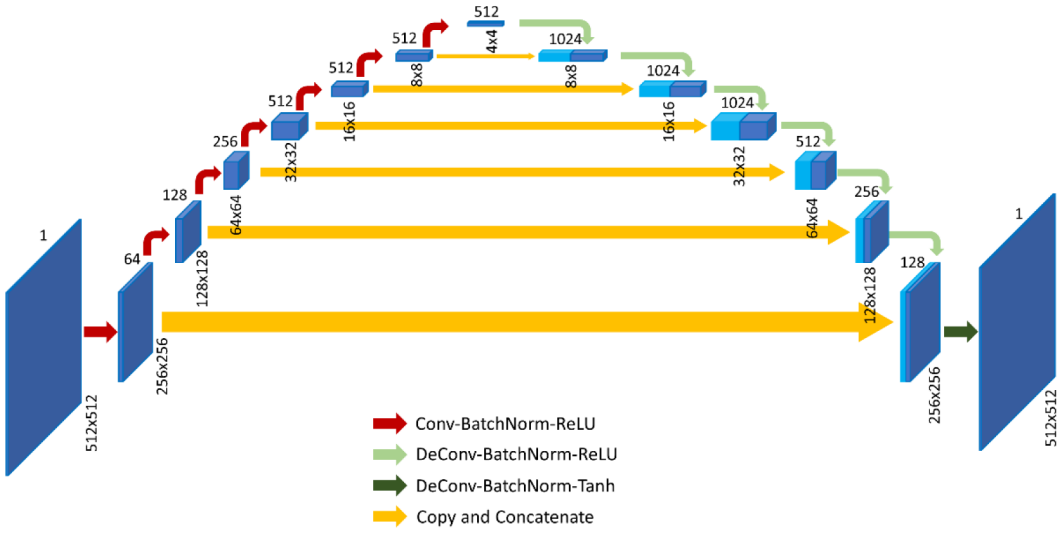

Fig. 4.

Architecture of the deep neural network. The neural network is composed of fourteen convolution or deconvolution layers with batch normalization, ReLU and Tanh activation function, as well as U-net style skip connections. The cuboids represent the feature maps of the input, intermediate, and output layers of the network. The feature resolution is denoted at bottom of each cuboid, and the corresponding channel size is on the top.

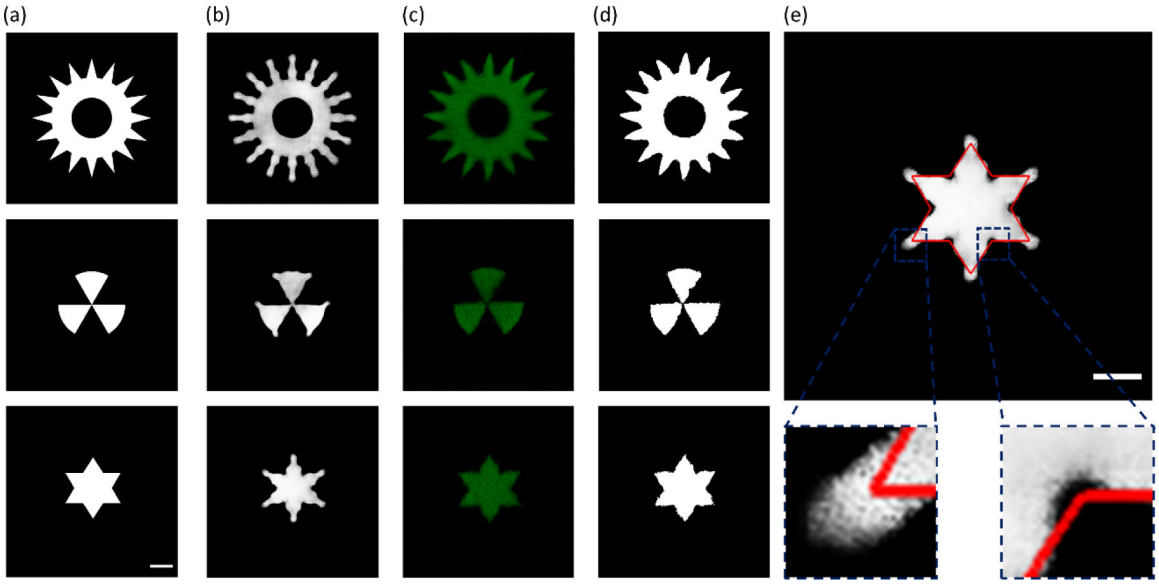

Fig. 5.

NN-calculated masks and the corresponding images of printing results. (a) The target structures. (b) The grayscale masks calculated by the trained NN. (c) The fluorescent images of the 3D printed structures using the masks from b. (d) The binarized images of c for easier comparison with the target. (e) The NN-calculated mask overlaid with a red contour showing its corresponding target structure. Scale bars are 200 μm.

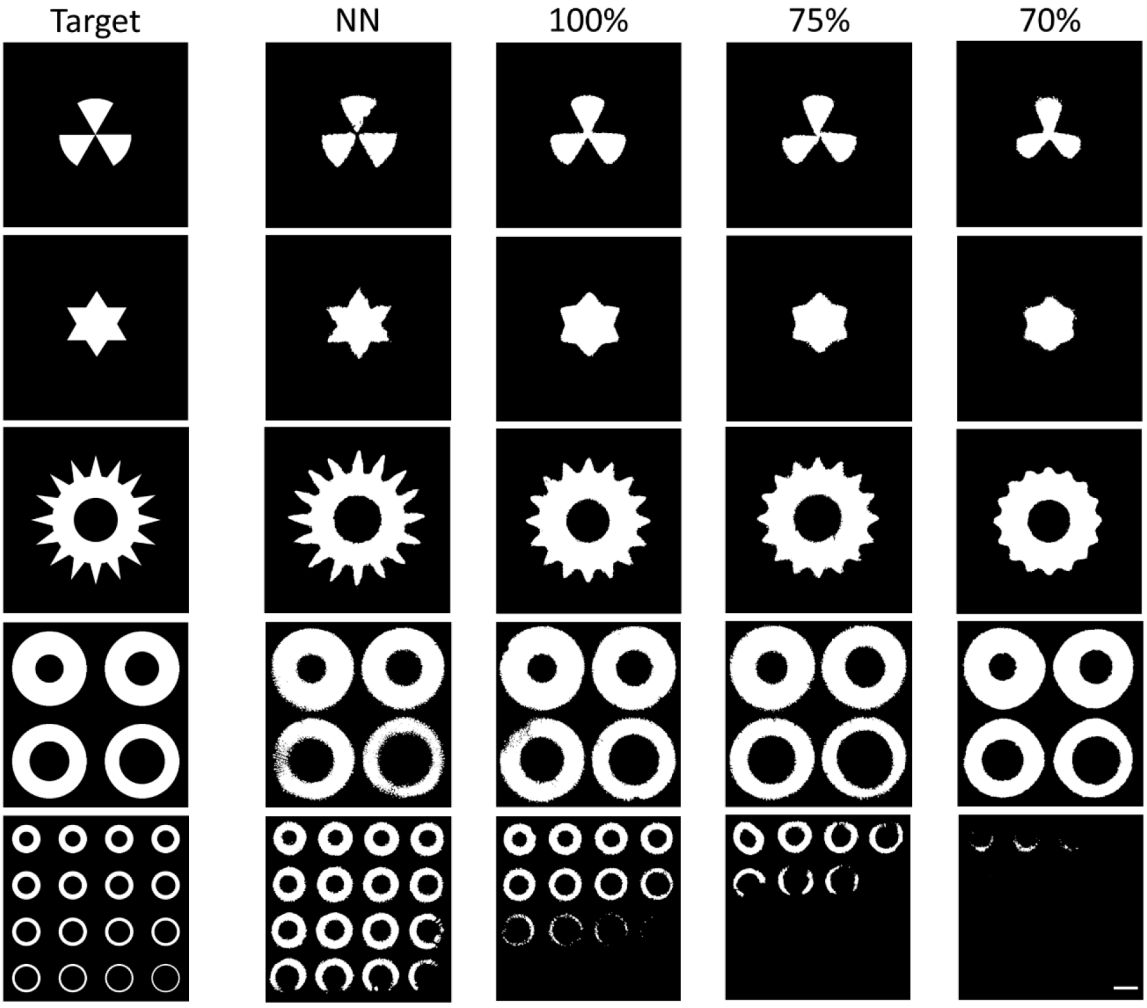

Fig. 6.

Postprocessed microscopic images of the test printing results. The first column shows the designed target structures. The second column shows the printed structure using NN-calculated masks. The third to fifth column are the printing results using the identical masks with 100%, 75%, and 70% exposure dose respectively. Scale bar is 200 μm.