Supplementary Material for "NeuPh: Scalable and Generalizable Neural Phase Retrieval with Local Conditional Neural Fields"

Hao Wang[a], **Jiabei Zhu**[a], **Yunzhe Li**[a,†], **Qianwan Yang**[a], **Lei Tian**[a,b,*]
[a]Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215, USA.
[b]Department of Biomedical Engineering, Boston University, Boston, MA 02215, USA.
[†]Current address: Department of Electrical Engineering & Computer Sciences, University of California, Berkeley, CA 94720, USA.

*Lei Tian, leitian@bu.edu

This document provides supplementary information to "NeuPh: Scalable, and Generalizable Neural Phase Retrieval with Local Conditional Neural Fields". We provide more details including the forward model of our imaging process, model-based reconstruction algorithms, detailed NeuPh network structure, reconstruction enhancement techniques of NeuPh network, data acquisition and preparation, details of network training, ablation study, and supplementary NeuPh reconstruction result.

1 Model-based reconstruction

We utilize Differential Phase Contrast (DPC)[20] for the initial estimation of cell phase images. It is important to acknowledge that DPC reconstruction relies on the weak object approximation, ensuring accurate results only when the phase change of the sample is below 0.64 radians[32]. However, our experimental setup involves Hela cell samples fixed in ethanol or formalin, leading to phase changes exceeding 2π, violating the weak object approximation and resulting in an underestimation of the object's phase. Despite these limitations, the DPC estimation serves as a valuable low-resolution initial guess for the

object's phase[24], and we incorporate this estimation into our network.

In our multiplexed Fourier Ptychographic Microscopy (FPM) experiment, utilizing only the five multiplexed intensity measurements to solve the FPM phase retrieval problem using the model-based algorithm results in severe reconstruction artifacts due to the highly ill-posed inverse problem[23]. To address this, for network training and to obtain ground truth phase images, we employ the standard sequential FPM measurement, iteratively reconstructing high-resolution phase images[21]. Details of the DPC and FPM forward models as well as the model-based DPC and FPM reconstruction are detailed in the following Secs. 1.1, 1.2, 1.3.

1.1 DPC-based phase imaging

Here, we briefly explain the principle of DPC phase imaging; additional details can be found in Ref. 20. DPC is a technique used to recover quantitative phase information from intensity images acquired with asymmetric illumination patterns. It offers an improved lateral resolution of $2\times$ compared to the native objective NA.

Under the weak object assumption: $o(\mathbf{c}) = e^{-\mu(\mathbf{c})+i\psi(\mathbf{c})} \approx 1 - \mu(\mathbf{c}) + i\psi(\mathbf{c})$, where $\mu(\mathbf{c})$ represents absorption and $\psi(\mathbf{c})$ represents phase, a BF intensity measurement $I_S(\mathbf{c})$ can be approximated to have a linear relationship with the sample[20]:

$$I_S(\mathbf{u}) = B\delta(\mathbf{u}) + H_{\text{abs}}(\mathbf{u})M(\mathbf{u}) + H_{\text{ph}}\Psi(\mathbf{u}), \tag{1}$$

where $I_S(\mathbf{u}), M(\mathbf{u}), \Psi(\mathbf{u})$ denote the spectrum of $I_s(\mathbf{c}), \mu(\mathbf{c})$ and $\psi(\mathbf{c})$, respectively, and

$\mathbf{u} = (u_x, u_y)$ represents the spatial frequency. B is a constant representing the background signal, and $\delta(\mathbf{u})$ is the Dirac delta function. $H_{\mathrm{abs}}, H_{\mathrm{ph}}$ are the transfer functions for amplitude and phase, respectively.

By subtracting the background term and normalizing the acquired BF intensity image, the DPC reconstruction can be formulated as:

$$\min_{M,\Psi} \sum_{j=1}^{N_{\mathrm{bf}}} \|I_{S-,j}(\mathbf{u}) - H_{\mathrm{abs},j}(\mathbf{u})M(\mathbf{u}) - H_{\mathrm{ph},j}\Psi(\mathbf{u})\|_2^2 + \tau_1 R_1(M(\mathbf{u})) + \tau_2 R_2(\Psi(\mathbf{u})), \quad (2)$$

where $I_{S-}(\mathbf{u})$ represents the spectrum of the background-subtracted intensity image, j is the index of DPC measurements, $N_{\mathrm{bf}} = 2$ denotes the number of captured BF images, and $\|\cdot\|_2$ represents the L_2 norm. τ_1, τ_2 are the regularization parameters, and R_1 and R_2 represent the regularization terms that incorporate prior information about the sample. Here, we utilized the L_2 regularization to solve the inverse problem[20].

1.2 FPM forward model

FPM is a recently developed computational imaging technique that enables increasing the imaging system's space-bandwidth product (SBP) by synthesizing multiple low-resolution images into a high-resolution image across a wide FOV[22]. The forward model of the standard sequential FPM describes the intensity image obtained from a single LED illumination. After appropriate normalization to account for the system magnification, it can be expressed as:

$$I_i(\mathbf{c}) = |\mathcal{F}^{-1}_{[O(\mathbf{u}-\mathbf{u}_i)P(\mathbf{u})]}(\mathbf{c})|^2, \quad (3)$$

where $I_i(\mathbf{c})$ represents the captured low-resolution intensity image for the i^{th} LED. $|\cdot|$ takes the amplitude of the complex field, and $\mathbf{c} = (x, y)$ denotes the lateral coordinates. $\mathcal{F}^{-1}$ represents the inverse Fourier transform, and $O(\mathbf{u})$ is the spectrum of the object $o(\mathbf{c})$. Each LED illumination is modeled as a plane wave with spatial frequency $\mathbf{u}_i = (u_{xi}, u_{yi}) = (\sin\theta_{xi}/\lambda, \sin\theta_{yi}/\lambda)$, where $(\theta_{xi}, \theta_{yi})$ defines the illumination angle of the i^{th} LED and λ denotes the central wavelength. The pupil function of the microscope, denoted by $P(\mathbf{u})$, is a circular low-pass filter with a diameter of $2\text{NA}/\lambda$, set by the objective lens NA.

In the case of multiplexed illumination, the sample is illuminated by different sets of LEDs based on different illumination patterns. The captured intensity image can be modeled as the sum of multiple intensity images obtained from individual LEDs[21]:

$$I_S(\mathbf{c}) = \sum_{i \in S} I_i(\mathbf{c}), \tag{4}$$

where the symbol $\in$ indicates that i is an element of the illumination set S. We use this multiplexed illumination forward model to generate our simulated data as mentioned in the main text.

1.3 Model-based FPM reconstruction

The standard sequential FPM reconstruction involves solving a non-convex optimization problem that jointly estimates the object $O(\mathbf{u})$ and the pupil function $P(\mathbf{u})$ by solving a

minimization problem:

$$\min_{O(\mathbf{k}),P(\mathbf{k}),\{b_i\}} \sum_{i=1}^{N_{\text{led}}} \left\| \sqrt{I_i(\mathbf{c})} - |\mathcal{F}^{-1}_{[O(\mathbf{u}-\mathbf{u}_i)P(\mathbf{u})]}(\mathbf{c})| - b_i \right\|_2^2, \tag{5}$$

where b_i is the background offset for the i^{th} image, and $N_{\text{led}} = 185$ is the total number of LEDs used in the standard sequential FPM measurement. The reconstruction from the standard sequential FPM measurement is performed by an iterative algorithm by following[21]. The results are used as the ground truth for training the NeuPh networks. It is worth noting that if only the five multiplexed intensity measurements are used to solve the FPM phase retrieval problem, the model-based algorithm results in severe reconstruction artifacts due to the highly ill-posed inverse problem[23].

2 Network structure of NeuPh

Our NeuPh network consists of a CNN-based encoder and an MLP-based decoder. A detailed visual illustration of the network can be found in Fig. S1.

Encoder: We use three CNN-based encoders, denoted as $\{e_1, e_2, e_3\}$ to independently encode three different types of input: BF, DF, and DPC images. Each encoder follows a deep residual network structure similar to Ref. 33. The encoders take specific image types as input and initially extract spatial features using a convolutional layer. The number of input channels for the first convolutional layer varies according to the number of input images: 2 for two BF images, 3 for three DF images, and 1 for the DPC image. The output channels for the first convolutional layer are fixed at 128 for all encoders.

After the initial convolutional layer, we employ 32 residual blocks to further extract spatial feature maps. Each residual block consists of two convolutional layers 128 input and output channels, a ReLU activation layer, and a multiplication layer with a factor of 1. Skip connections are incorporated in the residual blocks, where feature maps are added together. The spatial features extracted by the residual blocks are then passed through an output convolutional layer with 128 input and output channels. Finally, these features are added with the feature maps provided by the initial convolutional layer with a long skip connection. All convolutional layers use 3×3 convolutional kernels.

Once the feature maps are extracted from the three different types of input, they are concatenated to form the encoded latent space representation $\Phi \in \mathbb{R}^{H \times W \times D}$ of the image, where H and W represent the number of pixels along the x and y axes, respectively, while D represents the number of concatenated channels, which is calculated as 3 (number of encoders) $\times$ 128 (feature maps learned by each encoder) $= 384$. The H and W remain the same as the input low-resolution measurements, as we do not include pooling or upsampling layers in our encoder networks. The network structure of the encoders is visually depicted in Fig. S1: Encoder.

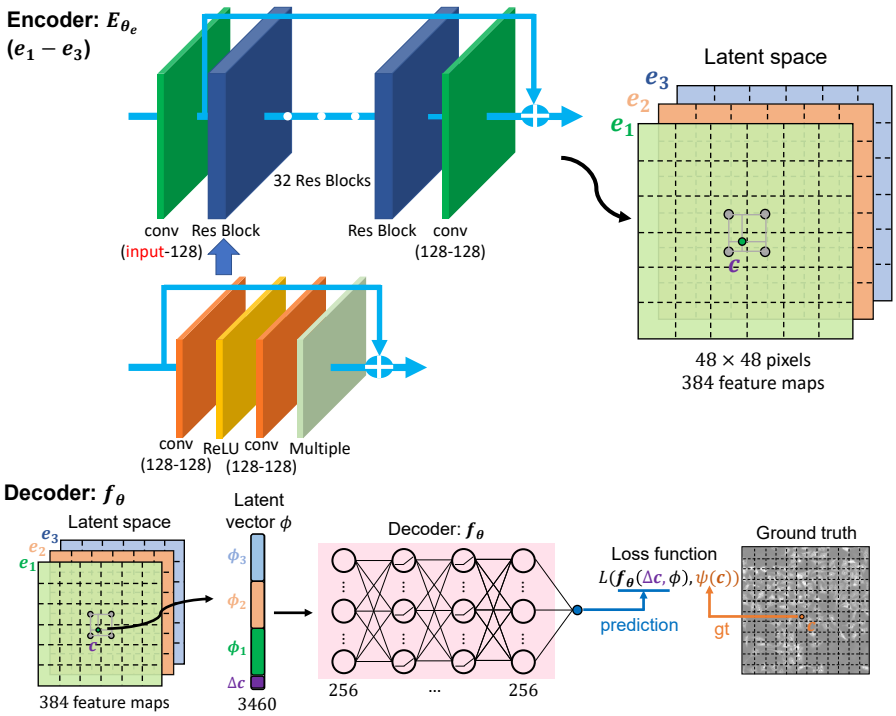

Fig S1: Network structure. The network structure of NeuPh consists of a CNN-based encoder and an MLP-based decoder. **Encoder:** The Encoder module includes three CNNs (e_1,e_2,e_3). Each encoder consists of an input convolutional layer, 32 residual blocks, and an output convolutional layer. The input channel of the input convolutional layer depends on the number of input images, denoted as "input" in the figure. For two brightfield intensity images, it has 2 channels. For three darkfield intensity images, it has 3 channels. And for the differential phase contrast image, it has 1 channel. The output channel for all three encoders is set to 128. Each residual block involves two convolutional layers (input channel: 128, output channel: 128), followed by a ReLU activation layer and a multiplication layer with a multiplication factor of 1. Each residual block utilizes a skip connection that adds the feature maps. The spatial features extracted by the residual blocks are then passed through the output convolutional layer, which has 128 input channels and 128 output channels. Finally, features extracted by the output convolutional layer are added with the feature maps provided by the input convolutional layer by a skip connection. All convolutional layers use 3×3 convolutional kernels. The feature maps generated by the three encoders are concatenated together to form the encoded latent space representation. **Decoder:** To generate a high-resolution image at the queried position based on the encoded latent vectors, we employ a 5-layer MLP with ReLU activation and hidden dimensions of 256 as the decoding function f_θ. The latent vector ϕ associated with the relative coordinates $\Delta\mathbf{c}$ of the queried position $\mathbf{c}$ (green dot) is constructed by concatenating the latent vectors generated by the three encoders: ϕ_1, ϕ_2, and ϕ_3. The input dimension of the MLP is 3460, calculated by 384 (feature maps learned by the encoder) $\times$ 9 (feature unfolding) + 2 (dimension of the coordinates) + 2 (cell decoding). The output dimension of the MLP is 1, representing the predicted phase value at the queried position. During training, we compared the MLP prediction $f_\theta(\Delta\mathbf{c}, \phi)$ with the ground truth $\psi(\mathbf{c})$ to train our network.

Decoder: To represent a high-resolution image in a continuous representation, we employ the LIIF approach [26], which represents an object in the encoded latent space and utilizes an MLP as a decoding function to decode the object from the latent space back to the object domain. In our case, we use a standard 5-layer MLP as the decoder, denoted as f_θ. Each layer of the MLP has 256 neurons, and ReLU activation is applied to the first four layers, while the last layer is unactivated. The input dimension of the MLP is 3460, which is obtained by 384 (number of features maps learned by the encoder) $\times$ 9 (feature unfolding) + 2 (dimension of the coordinates) + 2 (cell decoding), where feature unfolding and cell decoding are reconstruction enhancement techniques explained in Sec. 3. The output dimension of the MLP is 1, representing the predicted phase value at the queried location. The structure of the decoder is illustrated in Fig. S1: Decoder.

The decoding function can be expressed as:

$$\hat{\psi}(\mathbf{c}) = f_\theta(\mathbf{c}, \phi), \tag{6}$$

where $\hat{\psi}(\mathbf{c})$ is the decoded physical quantity, such as the phase value in our case, at the queried position $\mathbf{c}$. The variable $\mathbf{c}$ represents the 2D coordinates in the continuous image domain, assumed to range in $[-h, h]$ and $[-w, w]$ for the height and the width, respectively. $\phi \in \mathbb{R}^{1\times1\times D}$ is the selected latent vector from the latent space representation Φ, which is related to the queried position. The decoding function $f_\theta(\mathbf{c}, \phi)$ can be seen as a mapping function $f_\theta(\cdot, \phi) : \mathbf{c} \rightarrow \psi(\mathbf{c})$ that maps a coordinate to the phase value at the

position $\mathbf{c}$, with the latent vector ϕ as conditional parameters.

It should be noted that the latent space is a low-dimensional space with a dimension $H \times W \times D$, where we assign 2D coordinates to each latent vector with the pre-defined sparse grids, as depicted by the gray circles in Fig. S1 Encoder. However, for a continuous representation, we may need to query arbitrary coordinates that are not on the predefined grids, as shown by the green circle in Fig. S1 Encoder. Consequently, we cannot obtain the exact latent vector for the queried position $\mathbf{c}$ since the density of the grid in the latent space is much lower than that of the high-resolution grid ($H' \times W'$) for the same FOV. To bypass this issue, we adopt the LIIF approach [26], which assumes that the latent space is continuous; in addition, each latent vector can represent a local part of the continuous image and is responsible for predicting the signals at the set of coordinates that are closest to itself. Accordingly, we reformulate Eq. (6) as:

$$\hat{\psi}(\mathbf{c}) = f_\theta(\Delta\mathbf{c}, \phi), \tag{7}$$

where ϕ is the selected latent vector for coordinate $\mathbf{c}$, determined by the nearest latent vector based on the Euclidean distance. Here, $\Delta\mathbf{c} = \mathbf{c} - v$ and v represent the actual coordinate of the selected latent vector ϕ, respectively. Taking Fig. S1 Encoder as an example, the bottom-left gray circle represents the selected latent vector, and v denotes the coordinate of this chosen latent vector. To further improve our reconstruction result, we employ the reconstruction enhancement techniques, which are detailed in Sec. 3.

In summary, our network utilizes CNN-based encoders to encode the measurements into a low-dimensional latent space representation, where coordinates are assigned to latent vectors using predefined sparse grids. We can then query the phase value at arbitrary coordinates and use the MLP decoder to decode the physical quantity based on the selected latent vector. The latent space representation, generated by the encoders, adapts to different objects, allowing our decoding function $f_\theta(\cdot, \phi)$ to demonstrate robust generalization capabilities compared to traditional NF methods.

3 Reconstruction enhancement techniques

To enhance the information extraction from the latent space and improve the continuity of the reconstruction, we utilize feature unfolding, local ensemble, and cell decoding techniques as described in the LIIF method[26].

Feature unfolding: To capture additional information beyond a single latent vector ϕ, we employ feature unfolding, which extends ϕ to $\hat{\phi}$. Specifically, $\hat{\phi}$ is obtained by concatenating the 3×3 neighboring latent vectors of ϕ, as illustrated in Fig. S2(a), and is defined as:

$$\hat{\phi}_{p,q} = \mathrm{Concat}(\{\phi_{p+l,q+n}\})_{l,n\in\{-1,0,1\}}, \tag{8}$$

where Concat represents the concatenation of a set of latent vectors. The indices p and q correspond to the selected latent code ϕ that matches the queried coordinate $\mathbf{c}$ in the latent space. If the queried position is at the image's edge, the latent space Φ is padded with zero-vectors.

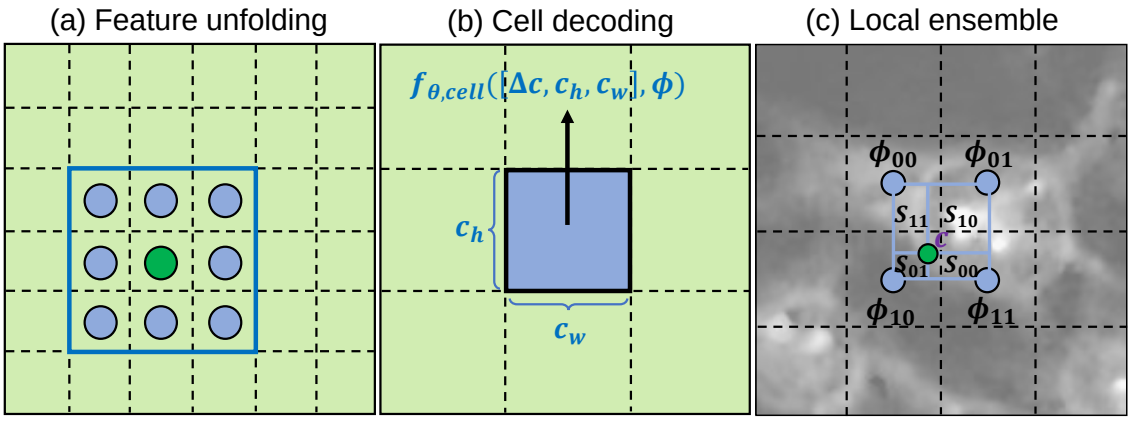

Fig S2: Reconstruction enhancement techniques. (a) Feature unfolding. The 3×3 neighborhood latent vectors (blue dots) surrounding the selected latent vector (green dot) are concatenated together to provide enriched information. (b) Cell decoding. The pixel size (c_h, c_w) are concatenated with the coordinates to improve the reconstruction. (c) Local ensemble. The local ensemble is used to enhance the continuity of the reconstruction. We utilize the nearest four latent vectors (blue dots, $\{\phi_{00}, \phi_{01}, \phi_{10}, \phi_{11}\}$) to make predictions. These four predictions are then merged by voting with normalized confidences, which are proportional to the area of the rectangle formed between the query point and its selected latent vector's diagonal counterpart.

Cell decoding: We incorporate cell decoding, which takes into account the pixel size information in the decoding function f_θ, as illustrated in Fig. S2(b). The updated decoding function is expressed as:

$$\hat{\psi}(\mathbf{c}) = f_{\theta,\text{cell}}([\Delta\mathbf{c}, c_h, c_w], \hat{\phi}), \tag{9}$$

where the $[c_h, c_w]$ specifies the height and width of the query pixel with the desired pixel size in the reconstruction. The notation $[\Delta\mathbf{c}, c_h, c_w]$ denotes the concatenation of the coordinate and the pixel size. Thus, $f_{\theta,\text{cell}}([\Delta\mathbf{c}, c_h, c_w], \hat{\phi})$ signifies that the decoding function reconstructs the value with the relative coordinate $\Delta\mathbf{c}$ and the pixel size (c_h, c_w), conditioned on the "unfolded" latent vector $\hat{\phi}$ at the coordinate $\mathbf{c}$.

Local ensemble A concern with Eq. (9) is the discontinuous prediction when the queried coordinate crosses the middle area between two neighboring latent vectors, resulting in a switch between latent codes (i.e. the selection of the nearest latent vector changes). For example, it occurs when the queried coordinate $\mathbf{c}$ (green dot) crosses the dashed line depicted in Fig. S2(c). Around such coordinates, predictions for two infinitesimally close coordinates are generated based on different latent vectors. Due to imperfections in the learned encoder E_{θ_e} and decoding function f_θ, these borders may exhibit discontinuous patterns. To address this issue, we employ the local ensemble technique, extending Eq. (9) to:

$$\hat{\psi}(\mathbf{c}) = \sum_{t \in \{00,01,10,11\}} \frac{S_t}{S} \cdot f_{\theta,\text{cell}}([\Delta\mathbf{c}_t, c_h, c_w], \hat{\phi}_t), \tag{10}$$

where $\hat{\phi}_t$ $(t \in \{00, 01, 10, 11\})$ represents the four nearest latent vectors (top-left, top-right, bottom-left, bottom-right) based on the queried coordinate, $\Delta\mathbf{c}_t$ denotes the relative coordinate between the queried coordinate and the selected latent vector, and S_t indicates the area of the rectangle between the queried coordinate and the coordinate of latent vector diagonal to the selected latent vector, as shown in Fig. S2(c). The weights S_t are normalized by $S = \sum_t S_t$. Moreover, the latent space representation Φ is mirror-padded outside the edge, allowing the above formula to work for coordinates near the image borders.

4 Data acquisition and preparation

In our study, we train our NeuPh using both experimental and simulated datasets. We first capture the experimental dataset as mentioned in the main manuscript Sec. 2.1. To prepare the input for training the NeuPh network, we performed the following steps. Firstly, we extracted the central 250×250 pixels from the raw low-resolution intensity images. Then, we applied dynamic range correction by clipping the minimum 0.1% and maximum 0.1% pixel values for each measurement, following the approach described in Ref. 23. This correction helped suppress shot noise and hot pixels. Next, we used the DPC reconstruction algorithm, as explained in Sec. 1.1, to generate a linear estimation of the phase based on the two BF intensity measurements. Additionally, we normalized the LED intensities by dividing the intensity images by their mean value. We also applied a morphological open operator to estimate and subtract the slow-changing background, following the method described in Ref. 23. This process effectively eliminated the unwanted background components and improved the accuracy of the subsequent learning process. Finally, we concatenated the preprocessed low-resolution intensity images with the DPC image.

To obtain ground-truth high-resolution phase images for the experimental data, we applied the following procedure. Firstly, for each standard FPM measurement, we sequentially illuminated 185 LEDs and captured the corresponding low-resolution intensity images. Then, we employed the model-based FPM reconstruction algorithm, detailed in Sec. 1.3, to reconstruct the phase of the central 250×250-pixel region and produce a

high-resolution phase image of 1500×1500 pixels. Next, we applied a phase unwrapping algorithm[34] to unwrap the reconstructed high-resolution phase image. Furthermore, we addressed the slow-varying background component present in the reconstructed high-resolution phase image by utilizing a morphological open operator with a kernel size of 50. This step removed the slowly changing background, enhancing the clarity and quality of the phase image. To normalize the range of values in the high-resolution phase image, we clipped the phase range within [0, 12] radian for the Hela cells fixed in both ethanol and formalin. Subsequently, we divided the phase images by this clipping threshold, resulting in a normalized range of values within [0, 1]. Finally, we paired the preprocessed low-resolution input images with the normalized high-resolution reconstructions, which served as the training data for our neural network.

We also create a simulated dataset to train our network. The whole dataset consisted of 900 cropped high-resolution natural images from the DIV2K dataset[25], each with a size of 600×600 pixels. Since the natural images have different histogram and spectral distributions compared to the biological cell images, we performed a preprocessing procedure on these images. The preprocessing involved removing the slowly varying background using an open operator with a kernel size of 20. Then, we applied a maximum value threshold of 0.6 to crop the image values and normalized the cropped images to the range $[0, 1]$ by dividing them by this threshold. To ensure consistency between the simulated dataset and the experimental Hela cells fixed in ethanol (here, we only utilize the data for the Hela cells fixed in ethanol since it contains more frequency content), we took steps to

match the power spectrum density (PSD) of the simulated dataset with the experimental data. This involved multiplying the spectrum of each simulated data with a correction map, whose value at a specific frequency is determined by the ratio between the square root of the PSDs of the experimental and the simulated dataset. We then normalized the spectrum-corrected image by dividing it by its maximum value. The resulting normalized and spectrum-corrected high-resolution images were used as ground truth for our network. The effect of the preprocessing procedure for the natural images can be observed in Fig. S3.

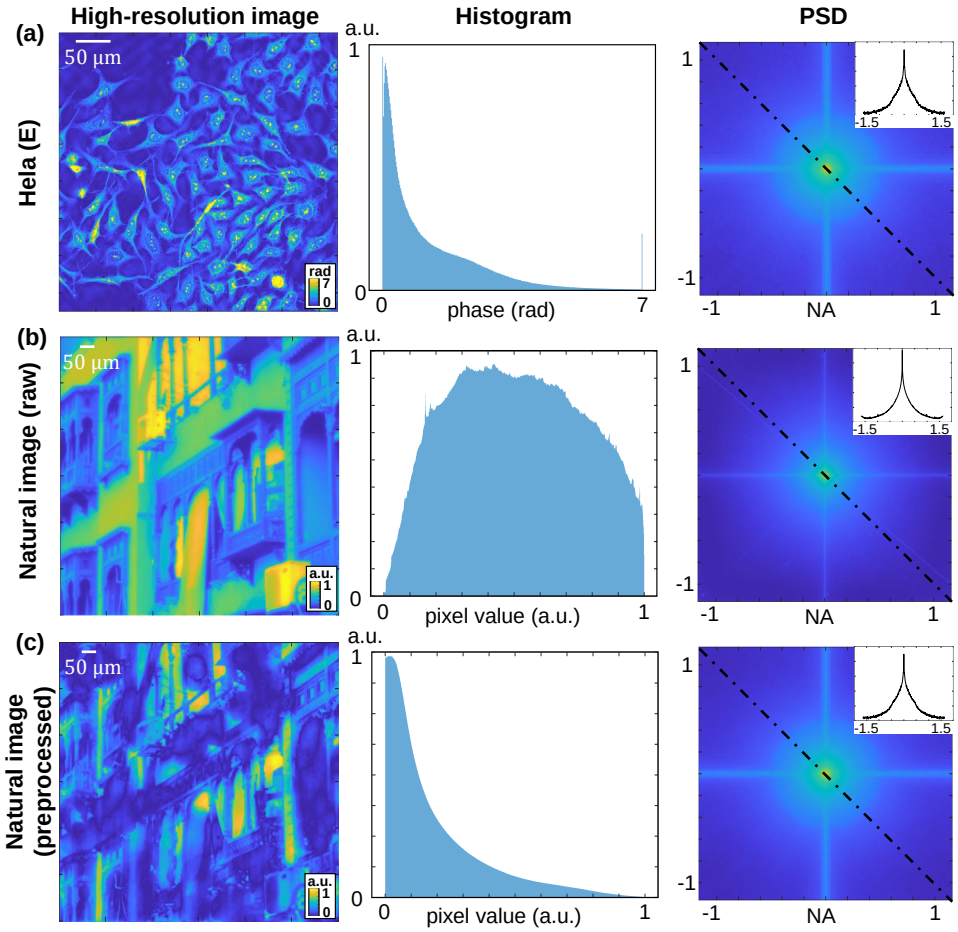

Fig S3: Preprocessing of natural images. To align the statistical characteristics of natural images with the experimental cell images, a preprocessing step is performed as described in Sec. 4. The figure presents the effects of this preprocessing step. (a) Model-based reconstruction of ethanol-fixed Hela cells. First column: an example of preprocessed high-resolution model-based FPM reconstruction. Second column: the histogram of the phase values derived from 22 preprocessed high-resolution Hela cell images. Third column: the PSD of the 22 preprocessed high-resolution reconstructions. The figure demonstrates that the phase values are predominantly distributed within the range of [0, 7] radian. However, a small fraction, approximately 0.5% of pixels lie outside this range. To capture valuable information and eliminate noisy pixels, the phase of the preprocessed experimental dataset is set within the range of [0, 12] radian. (b) Raw natural images in the DIV2K dataset. First column: an example high-resolution natural image in DIV2K dataset. Second column: the histogram of the pixel values obtained from 900 high-resolution natural images. Third column: the PSD of the 900 high-resolution natural images. (c) Preprocessed natural images. First column: an example background-removed and spectrum-corrected natural image. Second column: the histogram of the pixel values derived from 900 preprocessed natural images. Third column: the PSD of the 900 preprocessed images. The profiles of the PSD images along the diagonal (indicated by the dashed black lines) are shown in the top right corner of each image.

To generate the un-normalized object phase, we multiplied the normalized and spectrum-corrected high-resolution images by a factor of 9 and then subtracted 2.5, resulting in a phase range of $[-2.5, 6.5]$ radian. This range corresponds closely to the predominant distribution observed in the histogram of the experimental ethanol-fixed Hela cell dataset and also balances the effect from the large phase values observed in the experimental dataset. To simulate the low-resolution intensity images, we used Eq. (4) as the forward model and downsampled the simulated intensity images to 100×100 pixels. Throughout the simulation process, we make the assumption that our simulated system does not exhibit aberration. As a result, the pupil function $P(\mathbf{u})$ in Eq. (4) is considered an ideal circular low-pass filter, with a value of 1 within the circular region and 0 outside of it. We applied the same preprocessing steps used for the experimental dataset to obtain the preprocessed simulated low-resolution intensity images, which served as the input for the network. Finally, we paired the preprocessed high-resolution images with their corresponding low-resolution intensity images to create the training pairs for the network training.

5 Network training

5.1 Implementation of network training

To train our network with either purely experimental or simulated datasets, we follow the procedure outlined in Sec. 4 to prepare the training data, which consists of paired input images and the corresponding ground-truth phase images. To train networks using

a mixed dataset of experimental and simulated data, we adjust the value range of the ground truth images in the simulated dataset (originally ranging from 0 to 1) by scaling them with a factor of 0.75. This adjustment aligns the phase range of the simulated dataset ([0, 9] radians) to a normalized value range of [0, 0.75], matching the experimental dataset, which aligns the phase range ([0, 12] radians) to a normalized value range of [0, 1]. During each training step, we randomly crop a smaller patch of size 48×48 pixels from the input images. Recall that the size of the raw input measurements differs for the experimental and simulated datasets, with dimensions of 250×250 pixels and 100×100 pixels respectively.

We encode the input using three encoders, as described in detail in Sec. 2, resulting in a latent space representation Φ with dimensions of $H = 48, W = 48, D = 384$.

Subsequently, we assign 2D coordinates to each latent vector $\phi \in \mathbb{R}^{1\times1\times384}$, which is defined on a sparse grid with the same grid density of 48×48 as the input. The height and width range of the latent space is set as $[-H, H]$ and $[-W, W]$ respectively, resulting in a distance of 2 between neighboring latent vectors.

The high-resolution ground-truth phase images, with an original pixel resolution of 1500×1500 pixels for the experimental dataset and 600×600 pixels for the simulated dataset, are correspondingly cropped into 288×288-pixel patches to match the same FOV as the input images. This scaling indicates that our ground-truth high-resolution phase image has a pixel resolution $6\times$ higher than that of the input in both the x and y directions.

Similar to the assignment of 2D coordinates in the latent space, we assign 2D coordinates to the cropped high-resolution image within the height and width range of $[-H, H]$ and $[-W, W]$ respectively. The grid density is increased to 288×288, and the distance between adjacent pixels is reduced to 1/3. This coordinate assignment ensures positional consistency across the measurement domain, latent space, and reconstruction domain, assuming that the information within a 2D image is inherently positionally dependent and the information at a given position is preserved across different domains.

Next, we randomly select 2304 pixels from the high-resolution image patch as the ground-truth phase values by randomly choosing the coordinates defined in the high-resolution grid. These coordinates are also used to select the corresponding latent vectors ϕ from Φ, as described in Sec. 2. The selected latent vectors and relative coordinates are concatenated and input into the MLP. We further employ the reconstruction enhancement techniques detailed in Sec. 3.

The output of the MLP is the predicted phase value at the queried position $\mathbf{c}$, and we train our network by comparing this prediction with the ground truth using the L_1 norm.

It is important to note that during the training stage, we define grids for the high-resolution ground-truth image and query the high-resolution image at these predefined coordinates. However, after training, we no longer need to query points at predefined grids and can freely query phase values at any coordinates since our MLP can effectively represent the object in a continuous manner.

We utilize the PyTorch framework for training our network. The Adam optimizer is

employed, with an initial learning rate of 1×10^{-4}. To adaptively adjust the learning rate during training, we use the `ReduceLROnPlateau` method in PyTorch. This method reduces the learning rate by a factor of 0.2 when the loss function fails to improve. During training, except for the ablation study (as explained in Sec. 6), a batch size of 5 is used, and the total number of coordinates we used at each step is $2304 \times 5 = 11520$.

5.2 Training with different datasets

To thoroughly assess NeuPh's performance and investigate the impact of domain shift between simulated and experimental data on the reconstruction results, we trained 11 NeuPh models using various datasets. As mentioned in Sec. 2.1 of the main manuscript, we prepared three types of data: 22 pairs of Hela cells fixed in ethanol, 20 pairs fixed in formalin, and 900 pairs of simulated natural images. We use different numbers of these prepared images to train different networks, and the quantities for training and validation across different networks are summarized in Table S1, with the remaining data reserved for testing. These trained networks are categorized into the following four different scenarios:

Training with the full experimental dataset. We train two networks (**NeuPh$_{\mathbf{E(18)}}$**, **NeuPh$_{\mathbf{F(16)}}$**) using experimentally captured Hela cell datasets.

Training with a single pair of experimental dataset. To further evaluate NeuPh's generalization capability, we conduct training using only *a single training image pair* from the two cell types, denoted as **NeuPh$_{\mathbf{E(1)}}$** and **NeuPh$_{\mathbf{F(1)}}$**. This allows us to assess

the network's performance when trained on extremely limited experimental data.

Training with purely simulated natural images dataset. We also train three networks using only simulated datasets with the different number of natural images, denoted as $\mathbf{NeuPh_{Sim(18)}}$, $\mathbf{NeuPh_{Sim(16)}}$ and $\mathbf{NeuPh_{Sim}}$. The $\mathbf{NeuPh_{Sim(18)}}$ and $\mathbf{NeuPh_{Sim(16)}}$ networks are used to compare with the $\mathbf{NeuPh_{E(18)}}$ and $\mathbf{NeuPh_{F(16)}}$ respectively to study the impact of domain shift between the experiment and simulation on NeuPh's reconstruction. To fully utilize the available simulated images, we further train $\mathbf{NeuPh_{Sim}}$. These simulated datasets allow us to assess the network's performance in the absence of an experimental training dataset, and to gain insights into its ability to generalize from simulation to experiment.

Training with mixed experimental and simulated images. To further investigate the impact of domain shift between the experiment and simulation on NeuPh's reconstruction results, we conduct further training using datasets with different proportions of experimental and simulated images. To facilitate a comprehensive analysis, we maintained the total number of training data consistent with the experimental dataset while adjusting the ratio between experimental and simulated data within the training dataset. Specifically, for comparison with $\mathbf{NeuPh_{E(18))}}$, we train $\mathbf{NeuPh_{E:Sim(9:9)}}$ and $\mathbf{NeuPh_{E:Sim(1:17)}}$, and for comparison with $\mathbf{NeuPh_{F(16)}}$, we train $\mathbf{NeuPh_{F:Sim(8:8)}}$ and $\mathbf{NeuPh_{F:Sim}}(1{:}15)$. Additionally, as mentioned before, we trained NeuPh using pure-simulated datasets ($\mathbf{NeuPh_{Sim(18)}}$, $\mathbf{NeuPh_{Sim(16)}}$). This analysis enables us to further investigate how changes in the dataset composition affect NeuPh's reconstruction performance.

Additionally, to assess the networks' performance outside the training FOV, we collected two additional datasets, each containing 100 paired samples of Hela cells–one set fixed in ethanol and the other in formalin–as testing datasets.

Table S1: Training NeuPh with various datasets using different numbers and types of images. "Hela(E)" and "Hela(F)" denote experimental cells fixed in ethanol or formalin, respectively, while "Nature image" refers to simulated natural images. The numbers listed in table represent the quantity of images used for training the networks. Note that for networks trained with only one paired image ($\text{NeuPh}_{\text{E(1)}}$, $\text{NeuPh}_{\text{F(1)}}$), we assumed only a single image is available and thus use that single image for both training and validation.

Network	**Training dataset**	**Validation dataset**
$\text{NeuPh}_{\text{E(18)}}$	18 Hela(E)	2 Hela(E)
$\text{NeuPh}_{\text{E(1)}}$	1 Hela(E)	1 Hela(E)
$\text{NeuPh}_{\text{E:Sim(9:9)}}$	9 Hela(E) + 9 Nature image	1 Hela(E) + 1 Nature image
$\text{NeuPh}_{\text{E:Sim(1:17)}}$	1 Hela(E) + 17 Nature image	1 Hela(E) + 1 Nature image
$\text{NeuPh}_{\text{Sim(18)}}$	18 Nature image	2 Nature image
$\text{NeuPh}_{\text{F(16)}}$	16 Hela (F)	2 Hela(F)
$\text{NeuPh}_{\text{F(1)}}$	1 Hela (F)	1 Hela(F)
$\text{NeuPh}_{\text{F:Sim(8:8)}}$	8 Hela(F) + 8 Nature image	1 Hela(F) + 1 Nature image
$\text{NeuPh}_{\text{F:Sim(1:15)}}$	1 Hela(F) + 15 Nature image	1 Hela(F) + 1 Nature image
$\text{NeuPh}_{\text{Sim(16)}}$	16 Nature image	2 Nature image
$\text{NeuPh}_{\text{Sim}}$	800 Nature image	50 Nature image

6 Ablation study

6.1 DPC initialization

In our NeuPh structure, we include the DPC-based reconstruction as an input for the encoder e_3. To fully evaluate the impact of this initial reconstruction on the final high-resolution phase retrieval, we conducted an ablation study by removing the encoder e_3 and inputting only the bright and dark field intensity images into e_1 and e_2, respectively. We use full experimental data to train networks. The reconstruction results are shown in Fig. S6, and the test results for another 100 samples are presented in Table S2. From these results, we observe that the DPC initial reconstruction input helps improve our reconstruction. However, our network can still successfully reconstruct high-resolution phase images purely from the low-resolution intensity images, addressing the highly ill-posed problem.

6.2 Cell decoding

In our NeuPh, we introduce three different types of reconstruction enhancement techniques: feature unfolding, cell decoding, and local ensemble. The effects of these three techniques are fully studied in the Ref. 26. However, there are some differences between our dataset and the dataset used in Chen et al's[26] study. In our experiment, we train our NeuPh with a fixed six-times upsampling, which satisfies the minimum sampling requirement for the resolution-improved imaging system. In contrast, Chen et al. train their network with continuous random downsampling scales uniformly sampled in $\times 1$-

$\times 4$, meaning their cell decoding pixel size changes during training, while ours remains fixed. To fully evaluate the effect of the cell decoding technique in our experiment, we conduct an ablation study, training the network with and without cell decoding. We train our network with full experimental dataset and the results, shown in Fig. S6 and Table S2, indicate that cell decoding improves our reconstruction results.

6.3 MLP-based decoder

To evaluate the impact of using MLP as a decoder and coordinate-based training strategy, we conducted experiments to assess if NeuPh achieves better reconstruction results compared to CNN architectures. To rigorously evaluate NeuPh against CNNs, we replaced our MLP decoder with a CNN structure and trained it using the same dataset as NeuPh. Specifically, as shown in Fig. S4, we utilized NeuPh's encoder to obtain the latent space. Subsequently, instead of employing an MLP decoder, we utilized a widely used deep residual network structure as a decoder whose structure is similar to the encoder (e_1), as depicted in Fig. S1. To ensure the same number of trainable parameters with NeuPh, we initially employed a convolutional layer with 64 channels, followed by 12 residual blocks with 64 channels, and followed with another convolutional layer yielding a single output channel for reconstruction prediction in our CNN-based decoder. Notably, we omitted the long skip connection in this CNN-based decoder, as the number of output channels of the first convolutional layer (64) differs from the final output channel number (1).

During CNN training, like NeuPh, we randomly cropped 48×48 small patches from

input images at each training step. It is noteworthy that for NeuPh, theoretically, this step is unnecessary since we employ randomly selected points to train our network, effectively expanding our training dataset from a single paired image to millions of paired pixels. We include this step in NeuPh to further reduce memory cost. Conversely, in CNNs, this step is essential as it is similar to data augmentation. Without it, overfitting is likely, particularly when training with only a single paired image. To optimize CNN performance, we did not use upsampling layers to increase lateral pixel number by $6\times$, instead, we employ cubic interpolation for our input low-resolution images as Ref. 23. Consequently, the input lateral pixel number was set to 288×288. Subsequently, we fed the BF, DF, and DPC images into three encoders as NeuPh and employed our CNN-based decoder to reconstruct the image from the latent space. High-resolution images served as the ground truth. Both the initial learning rate and learning rate adjustment strategy mirrored NeuPh's settings. However, due to increased lateral pixel numbers and the utilization of the CNN-based decoder, the memory requirements for storing intermediate variables were several times larger than NeuPh. To assess the extreme case and consider the memory of our GPU, we trained our CNN with a single paired image with a batch size setting of 1. The NeuPh is thus also retrained with batch size 1 for a fair comparison. Both networks were trained until convergence. Because of the computational cost limitation, we cannot infer the whole image (250×250 pixels) directly using the CNN, thus we performed inference patch-by-patch (48×48 pixels) and then utilized alpha blending as Ref. 23 to obtain the final result. In summary, we ensure a fair comparison and employ several tech-

niques to enhance the performance of CNN. While there may be additional techniques to fine-tune the CNN structure, in comparison, our NeuPh requires only 5 fully connected layers and avoids the need for designing complex structures. We utilize quantitative metrics such as mean square error (MSE), structural similarity index measure (SSIM), peak signal-to-noise ratio (PSNR), and frequency measurement (FM)[28] to evaluate the reconstruction and the comparison result is detailed in Fig. 3(b) in the main text with additional results shown in Fig. S7(b) and S10. The metrics for Fig. 3(b), S7(b) and S10 are shown in Table S3. The metrics for an additional 100 test samples are presented in Table S4. As we can see from the figure and tables, our NeuPh reconstruction is better than the CNN-based structure.

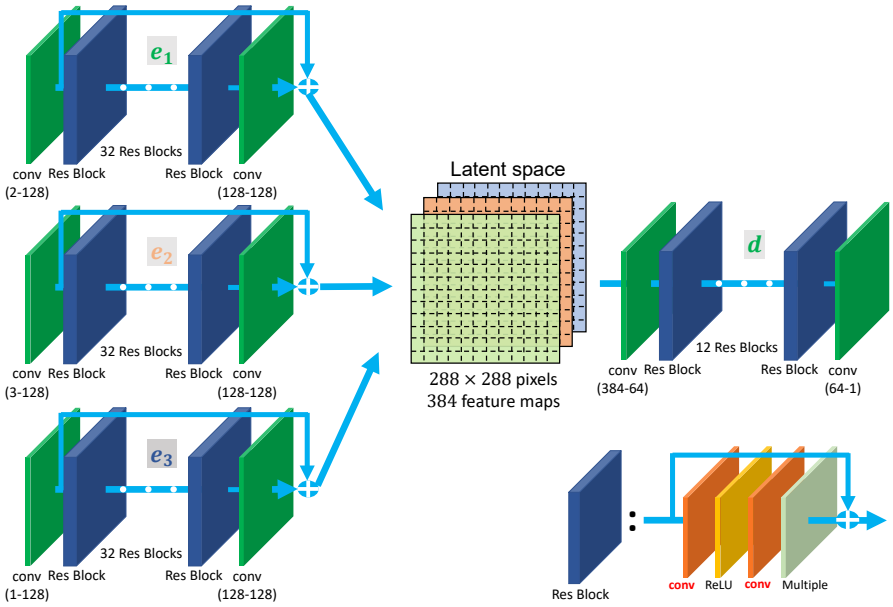

Fig S4: CNN structure used for ablation study. We replace the NeuPh's MLP-based decoder with a CNN-based structure. The Encoder module is the same as NeuPh, including three CNNs (e_1,e_2,e_3). The Decoder module, instead, uses a CNN-based deep residual network (d). The input-output channels in the residual blocks are set at 128-128 for the encoders (e_1,e_2,e_3) and 64-64 for the decoder (d), respectively. Notably, the lateral pixel dimensions at the latent space of the CNN are 288×288 since the raw FPM input is first preprocessed by cubic interpolation, differing from NeuPh's dimension (64×64).

7 Supplementary NeuPh reconstruction result

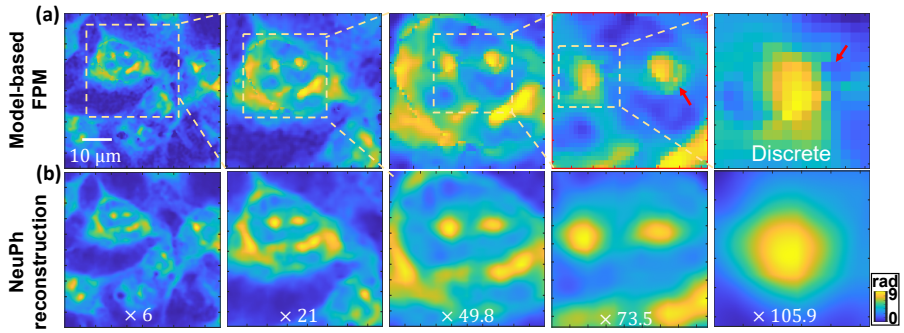

Fig S5: NeuPh learns continuous-domain representation and can reconstruct phase maps on an arbitrary pixel grid (illustration with $6\times, 21\times, 49.8\times, 73.5\times$, and $105.9\times$ upsampling compared with the raw measurement image). (a) Model-based FPM reconstruction with zoom-in areas. Reconstruction artifacts are noted by red arrows. (b) NeuPh continuous reconstruction of high-resolution phase image.

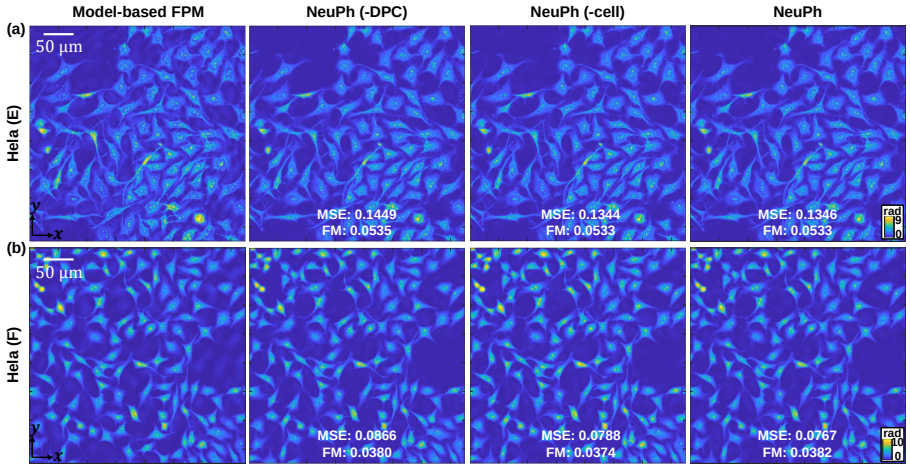

Fig S6: Effect of DPC initial reconstruction and cell decoding on NeuPh. Reconstructed phase images without DPC input (**NeuPh(-DPC)**), without cell decoding (**NeuPh(-cell)**), and with complete NeuPh reconstruction (**NeuPh**) are shown. The MSE and FM values are noted at the bottom of the figures. For Hela cells fixed in Ethanol (Hela(E)), the PSNR and SSIM values for **NeuPh(-DPC)** are 29.9734dB and 0.7890, respectively, for **NeuPh(-cell)** they are 30.2998dB and 0.8043, and for **NeuPh** they are 30.2943dB and 0.8012. For Hela cells fixed in Formalin (Hela(F)), the PSNR and SSIM values for **NeuPh(-DPC)** are 32.2070dB and 0.8319, respectively, for **NeuPh(-cell)** they are 32.6161dB and 0.8482, and for **NeuPh** they are 32.7348dB and 0.8474.

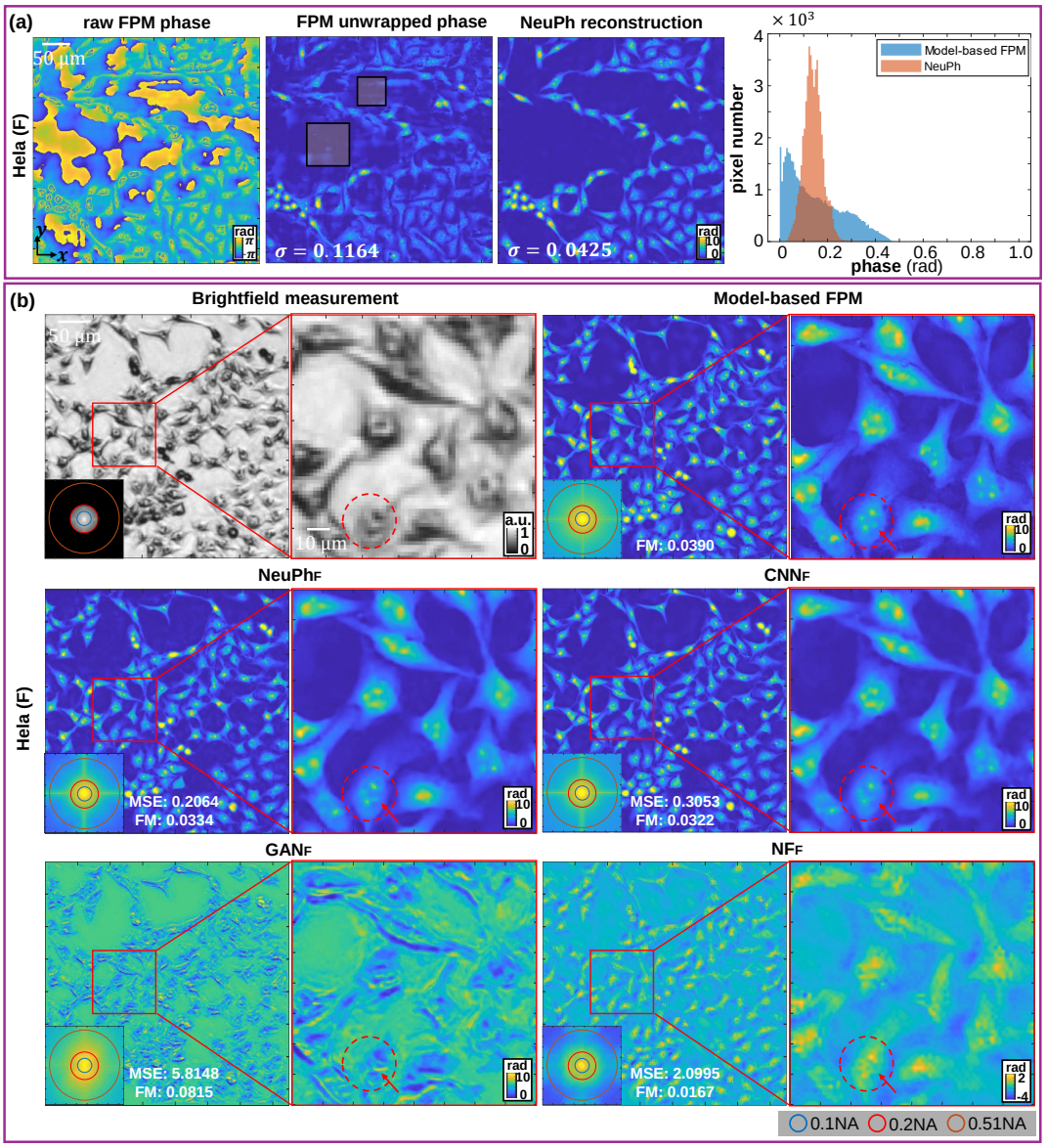

Fig S7: (a) NeuPh's robustness to phase artifacts. NeuPh eliminates the background artifacts (noted by the block box). The phase histogram of the background areas, measuring the residual background fluctuations, is shown in the rightmost column. The standard deviations (σ) are shown at the bottom of the reconstructions. (b) NeuPh outperforms CNN-based reconstruction method and existing neural networks. Comparison between the reconstructions by NeuPh (**NeuPh$_F$**), CNN-based (**CNN$_F$**) networks, GAN (**GAN$_F$**) and traditional NF network (**NF$_F$**), benchmarked by the ground-truth model-based reconstruction. Zoomed-in regions showcase intricate subcellular features that can be reconstructed with better resolution by NeuPh than other neural networks, as highlighted by the red circles and arrows. The reconstructed spectra are shown at the bottom left of each image, with blue, red, and brown circles indicating the bandwidth of the objective (0.1 NA), BF measurements (0.2 NA), and theoretically achievable reconstruction bandwidth (0.51 NA), respectively. The MSE and FM are noted at the bottom of figures.

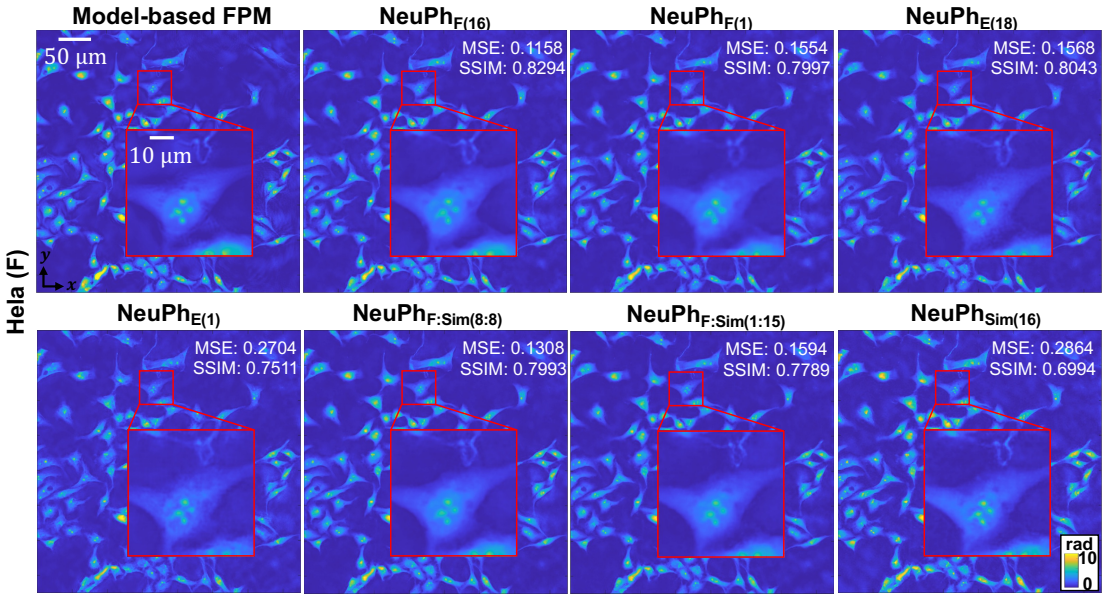

Fig S8: Strong generalization capability of NeuPh. Reconstructions of formalin-fixed Hela cells with different dataset-trained networks.

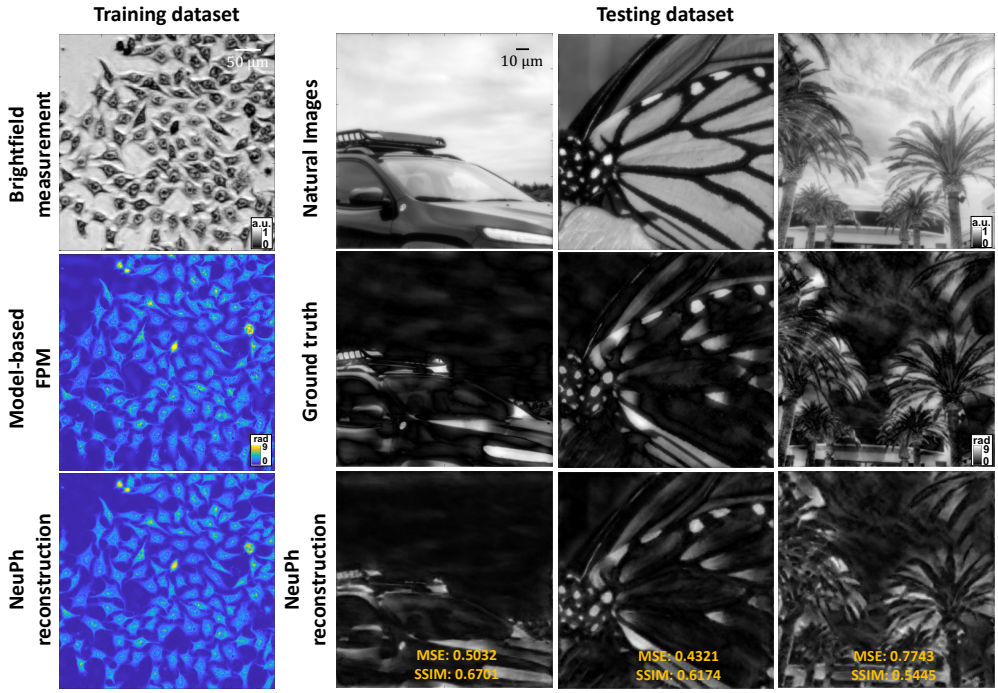

Fig S9: NeuPh's ability to generalize from an experimental dataset to simulated datasets. The training dataset comprises 18 pairs of Hela cells captured experimentally and fixed in ethanol. Our testing dataset ground truth consists of background-removed and spectrum-adjusted natural images. NeuPh reconstruction demonstrates that our network can effectively reconstruct images such as cars, butterflies, and trees, which were not "seen" during the training process. The MSE and SSIM are noted at the bottom of the images. The PSNR values for "car", "butterfly", and "tree" are 22.0678 dB, 22.7288 dB, and 20.1957 dB, respectively. Additionally, the FM values for "car", "butterfly", and "tree" are 0.0164, 0.0167, and 0.0149, with corresponding ground truth FM values of 0.0256, 0.0198, and 0.0230, respectively.

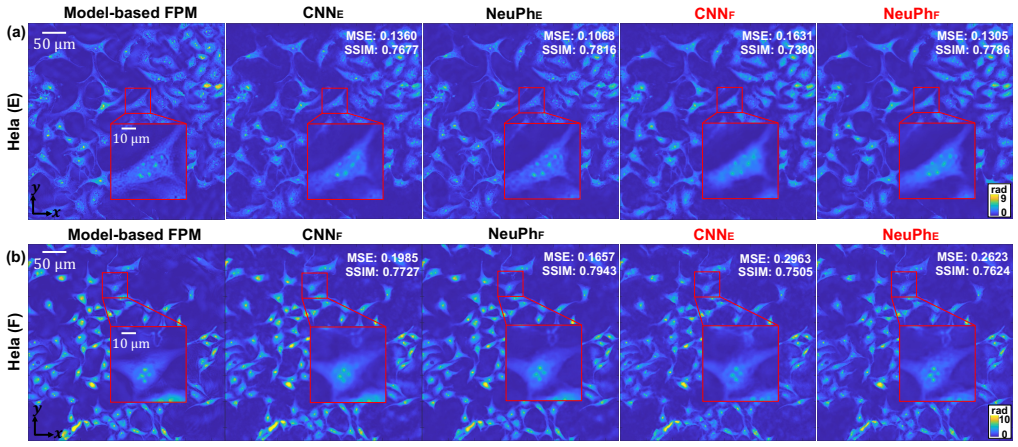

Fig S10: The reconstruction of Hela cells using NeuPh and CNN-based structure trained with different datasets. The networks trained and tested with different types of datasets are indicated in red. The MSE and FM are noted at the top right of each figure, while other metrics are presented in Table S3.

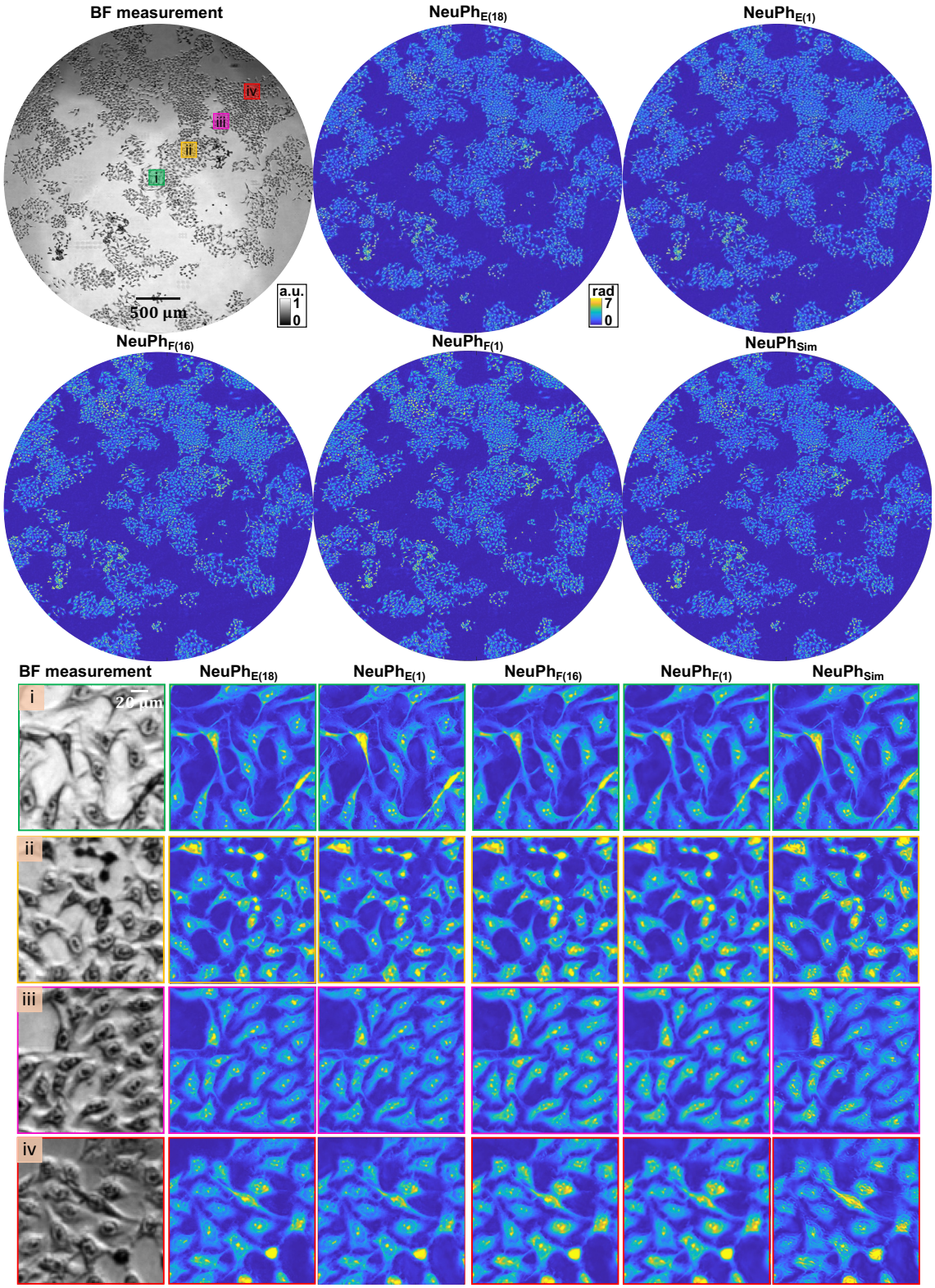

Fig S11: Wide-FOV high-resolution reconstructions of Hela cells fixed in ethanol using NeuPh networks trained with different datasets.

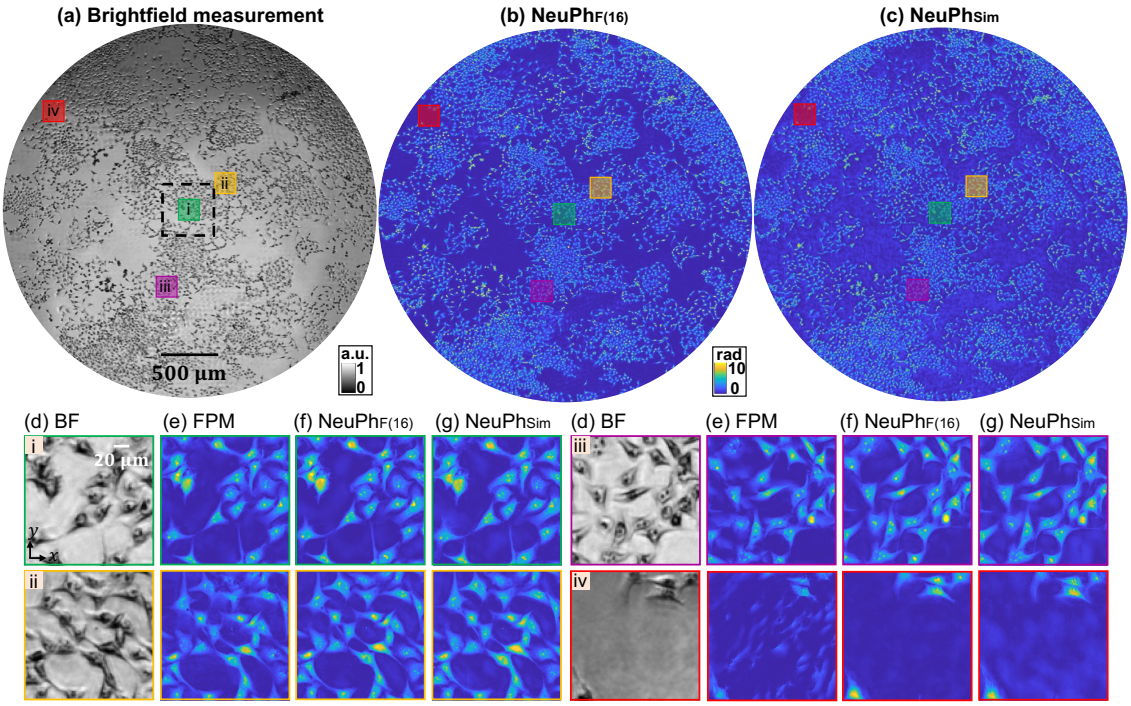

Fig S12: Wide-FOV high-resolution NeuPh reconstruction of Hela cells fixed in formalin. (a) BF intensity image captured over a 2160-pixel diameter (3.51 mm) FOV. Wide-FOV NeuPh reconstruction by (b) $\mathbf{NeuPh_{F(16)}}$ and (c) $\mathbf{NeuPh_{Sim}}$. (d-g) Selected subareas extracted from the central to the edge of the FOV, identified as i-iv and enclosed within different colored boxes. (d) BF intensity image. (e) model-based FPM reconstruction. (f) $\mathbf{NeuPh_{F(16)}}$ reconstruction trained with the experimental dataset. The experimental dataset used for training the NeuPh network is obtained from the central region, indicated by the dashed black square. (g) $\mathbf{NeuPh_{Sim}}$ reconstruction trained with the simulated dataset. NeuPh can successfully reconstruct full-FOV high-resolution images and eliminate the artifacts introduced by the model-based reconstruction, as shown in iv.

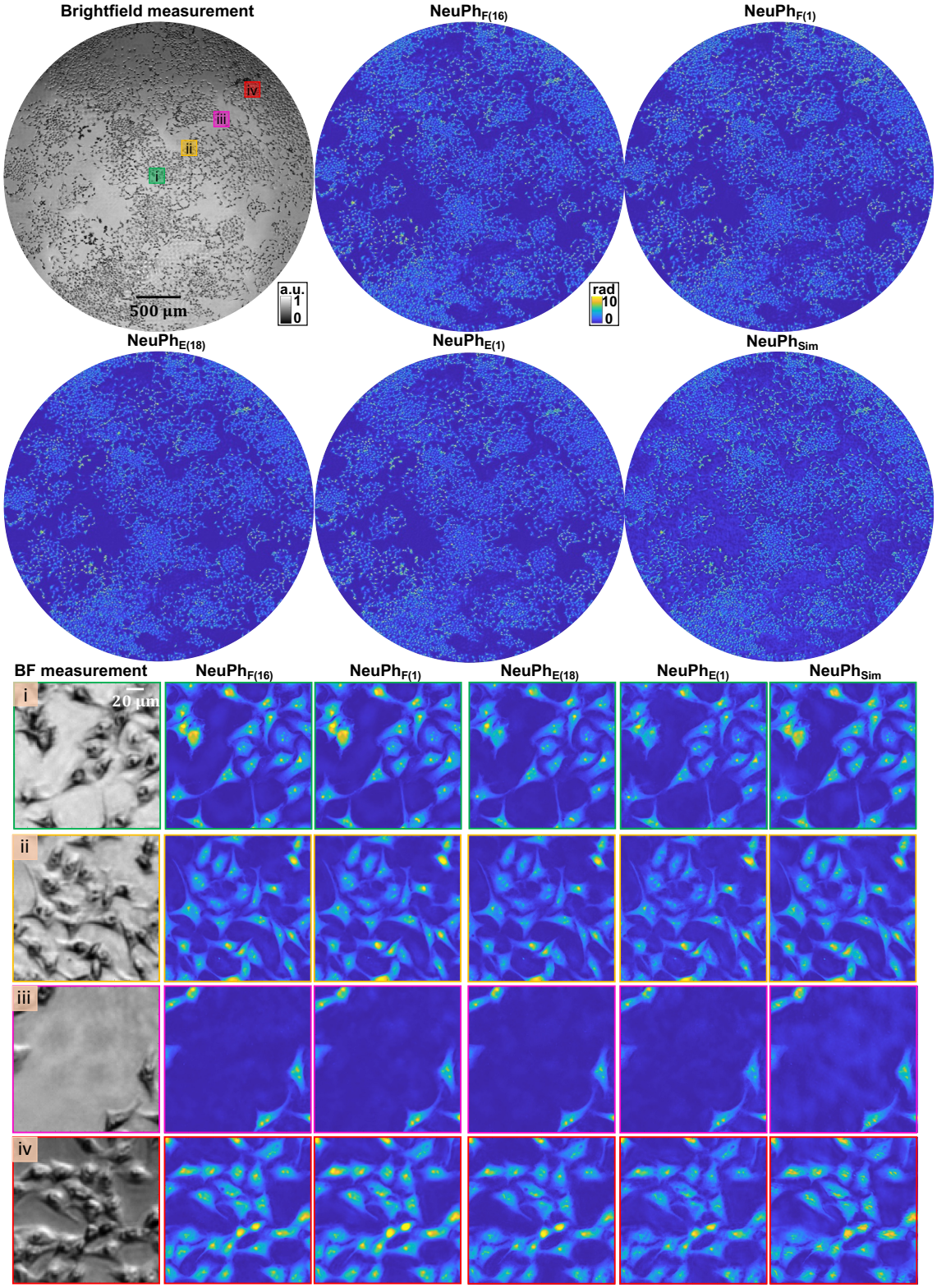

Fig S13: Wide-FOV high-resolution reconstructions of Hela cells fixed in formalin using NeuPh networks trained with different datasets.

Table S2: Quantitative metrics for comparison between NeuPh without DPC initialization input (NeuPh(-DPC)), without cell decoding (NeuPh(-cell)), and NeuPh (mean $\pm$ standard deviation). In this context, "Hela (E)" and "Hela (F)" refer to fixed Hela cells preserved in ethanol (E) and formalin (F), respectively. The following metrics were computed by comparing the 100 high-resolution phase images (1500×1500 pixels) predicted by the different networks and the model-based FPM reconstructions. The reconstruction patches were extracted from FOVs beyond those in the training dataset.

Dataset	**Method**	**MSE**	**PSNR(dB)**	**SSIM**	**FM**
Hela (E)	$\text{NeuPh}_{\text{E(18)}}$(-DPC)	0.1798±0.0621	29.9059±1.7272	0.7961±0.0233	0.0618±0.0110
	$\text{NeuPh}_{\text{E(18)}}$(-cell)	0.1753±0.0604	30.0121±1.7274	0.7994±0.0245	0.0611±0.0108
	$\text{NeuPh}_{\text{E(18)}}$	0.1753±0.0606	30.0157±1.7363	0.7999±0.0243	0.0609±0.0107
	Model-based FPM	-	-	-	0.0689±0.0141
Hela (F)	$\text{NeuPh}_{\text{F(16)}}$(-DPC)	0.2068±0.0728	30.7228±2.0886	0.7820±0.0479	0.0445±0.0064
	$\text{NeuPh}_{\text{F(16)}}$(-cell)	0.2001±0.0688	30.8585±2.0770	0.7815±0.048	0.0435±0.0061
	$\text{NeuPh}_{\text{F(16)}}$	0.1978±0.0685	30.9115±2.0947	0.7817±0.0484	0.0445±0.0061
	Model-based FPM	-	-	-	0.0479±0.0080

Table S3: Quantitative metrics for comparison between NeuPh and CNN. In this context, "Hela (E)" and "Hela (F)" refer to fixed Hela cells preserved in ethanol (E) and formalin (F), respectively." Figure" indicates the evaluated figure. Here, we consider the model-based FPM reconstruction as the ground truth for assessing the performance of the networks.

Dataset	**Figure**	**Method**	**MSE**	**PSNR(dB)**	**SSIM**	**FM**
Hela (E)	Fig. 3	CNN_E	0.3816	25.6853	0.6912	0.0342
		$NeuPh_E$	0.2229	28.0203	0.7219	0.0501
		Model-based FPM	-	-	-	0.0592
Hela (E)	Fig. S10	CNN_E	0.1360	29.3833	0.7677	0.0520
		$NeuPh_E$	0.1068	30.4336	0.7816	0.0610
		CNN_F	0.1631	28.5938	0.7380	0.0419
		$NeuPh_F$	0.1305	29.5632	0.7786	0.0486
		Model-based FPM	-	-	-	0.0789
Hela (F)	Fig. S7	CNN_F	0.3053	26.7357	0.7994	0.0322
		$NeuPh_F$	0.2064	28.4360	0.8303	0.0334
		Model-based FPM	-	-	-	0.0390
Hela (F)	Fig. S10	CNN_F	0.1985	28.6055	0.7727	0.0378
		$NeuPh_F$	0.1657	29.3893	0.7943	0.0399
		CNN_E	0.2963	26.8670	0.7505	0.0460
		$NeuPh_E$	0.2623	27.3954	0.7624	0.0499
		Model-based FPM	-	-	-	0.0511

Table S4: Quantitative metrics for comparison between NeuPh and CNN (mean $\pm$ standard deviation). In this context, "Hela (E)" and "Hela (F)" refer to fixed Hela cells preserved in ethanol (E) and formalin (F), respectively. The following metrics were computed by comparing the 100 high-resolution phase images (1500×1500 pixels) predicted by the different networks and the model-based FPM reconstructions. The reconstruction patches were extracted from FOVs beyond those in the training dataset.

Dataset	**Method**	**MSE**	**PSNR(dB)**	**SSIM**	**FM**
Hela (E)	CNN_E	0.2899±0.1078	27.8779±1.3687	0.7402±0.0315	0.0537±0.0089
	NeuPh_E	0.2261±0.0816	28.9374±1.4266	0.7442±0.0288	0.0610±0.0100
	CNN_F	0.3208±0.0993	27.3610±1.5195	0.7208±0.0348	0.0430±0.0073
	NeuPh_F	0.2642±0.0882	28.2321±1.6716	0.7534±0.0271	0.0473±0.0080
	Model-based FPM	-	-	-	0.0689±0.0141
Hela (F)	CNN_F	0.2947±0.0634	28.9795±1.2844	0.7560±0.0513	0.0396±0.0051
	NeuPh_F	0.2304±0.0584	30.1360±1.6085	0.7756±0.0489	0.0422±0.0058
	CNN_E	0.3911±0.1167	27.8709±1.1355	0.7309±0.0500	0.0479±0.0065
	NeuPh_E	0.3015±0.0869	28.9929±1.1656	0.7292±0.0556	0.0522±0.0071
	Model-based FPM	-	-	-	0.0479±0.0080

Table S5: Quantitative metrics for different dataset trained NeuPh. "Hela (E)", and "Hela (F)" refer to Hela cells fixed in ethanol (E) and formalin (F), respectively. For the experiment dataset, we use model-based FPM reconstruction as the ground truth.

Dataset	**Method**	**MSE**	**PSNR(dB)**	**SSIM**	**FM**
Hela (E)	$\text{NeuPh}_{\text{E(18)}}$	0.0949	30.9448	0.8026	0.0591
	$\text{NeuPh}_{\text{E(1)}}$	0.1044	30.5291	0.7705	0.0582
	$\text{NeuPh}_{\text{F(16)}}$	0.1147	30.1204	0.7913	0.0550
	$\text{NeuPh}_{\text{F(1)}}$	0.1262	29.7082	0.7779	0.0485
	$\text{NeuPh}_{\text{E:Sim(9:9)}}$	0.1016	30.6507	0.7954	0.0558
	$\text{NeuPh}_{\text{E:Sim(1:17)}}$	0.1177	30.0093	0.7707	0.0518
	$\text{NeuPh}_{\text{Sim(18)}}$	0.1490	28.9874	0.7100	0.0576
	Model-based FPM	-	-	-	0.0789
Hela (F)	$\text{NeuPh}_{\text{F(16)}}$	0.1158	30.9461	0.8294	0.0429
	$\text{NeuPh}_{\text{F(1)}}$	0.1554	29.6702	0.7997	0.0395
	$\text{NeuPh}_{\text{E(18)}}$	0.1568	29.6301	0.8043	0.0433
	$\text{NeuPh}_{\text{E(1)}}$	0.2704	27.2629	0.7511	0.0466
	$\text{NeuPh}_{\text{F:Sim(8:8)}}$	0.1308	30.4170	0.7993	0.0421
	$\text{NeuPh}_{\text{F:Sim(1:15)}}$	0.1594	29.5575	0.7789	0.0406
	$\text{NeuPh}_{\text{Sim(16)}}$	0.2864	27.0143	0.6994	0.0421
	Model-based FPM	-	-	-	0.0511

Table S6: Quantitative metrics for networks trained with varied datasets (mean $\pm$ standard deviation). In this context, "Hela (E)" and "Hela (F)" refer to fixed Hela cells preserved in ethanol (E) and formalin (F), respectively. The following metrics were computed by comparing the 100 high-resolution phase images (1500×1500 pixels) predicted by the networks trained with different datasets and the model-based FPM reconstructions. The reconstruction patches were extracted from FOVs beyond those in the training dataset.

Dataset	**Method**	**MSE**	**PSNR(dB)**	**SSIM**	**FM**
Hela (E)	$\text{NeuPh}_{\text{E(18)}}$	0.1753 ± 0.0606	30.0157 ± 1.7363	0.7999 ± 0.0243	0.0609 ± 0.0107
	$\text{NeuPh}_{\text{E(1)}}$	0.2233 ± 0.0818	29.0015 ± 1.3947	0.7275 ± 0.0344	0.0573 ± 0.0099
	$\text{NeuPh}_{\text{F(16)}}$	0.2356 ± 0.0848	28.7574 ± 1.8154	0.7715 ± 0.0231	0.0531 ± 0.0095
	$\text{NeuPh}_{\text{F(1)}}$	0.2591 ± 0.0872	28.3186 ± 1.6702	0.7516 ± 0.0290	0.0463 ± 0.0079
	$\text{NeuPh}_{\text{E:Sim(9:9)}}$	0.1792 ± 0.0589	29.9085 ± 1.7085	0.7986 ± 0.0241	0.0597 ± 0.0106
	$\text{NeuPh}_{\text{E:Sim(1:17)}}$	0.2241 ± 0.0769	28.9600 ± 1.4084	0.7431 ± 0.0267	0.0534 ± 0.0102
	$\text{NeuPh}_{\text{Sim(18)}}$	0.3347 ± 0.1217	27.2564 ± 1.4643	0.6282 ± 0.0307	0.0582 ± 0.0116
	$\text{NeuPh}_{\text{Sim}}$	0.3704 ± 0.1290	26.7218 ± 1.6527	0.6435 ± 0.0310	0.0772 ± 0.0157
	Model-based	-	-	-	0.0689 ± 0.0141
Hela (F)	$\text{NeuPh}_{\text{F(16)}}$	0.1978 ± 0.0685	30.9115 ± 2.0947	0.7817 ± 0.0484	0.0445 ± 0.0061
	$\text{NeuPh}_{\text{F(1)}}$	0.2322 ± 0.0685	30.1063 ± 1.6490	0.7738 ± 0.0499	0.0419 ± 0.0059
	$\text{NeuPh}_{\text{E(18)}}$	0.2074 ± 0.0467	30.5551 ± 1.2916	0.7934 ± 0.0469	0.0448 ± 0.0065
	$\text{NeuPh}_{\text{E(1)}}$	0.3075 ± 0.0917	28.9166 ± 1.1433	0.7252 ± 0.0557	0.0483 ± 0.0066
	$\text{NeuPh}_{\text{F:Sim(8:8)}}$	0.2092 ± 0.0690	30.6509 ± 2.0096	0.7825 ± 0.0480	0.0438 ± 0.0060
	$\text{NeuPh}_{\text{F:Sim(1:15)}}$	0.2386 ± 0.0599	29.9841 ± 1.5689	0.7805 ± 0.0437	0.0425 ± 0.0065
	$\text{NeuPh}_{\text{Sim(16)}}$	0.3491 ± 0.0880	28.3090 ± 1.0581	0.6748 ± 0.0575	0.0438 ± 0.0071
	$\text{NeuPh}_{\text{Sim}}$	0.3358 ± 0.0713	28.4408 ± 1.4819	0.6527 ± 0.0598	0.0503 ± 0.0092
	Model-based	-	-	-	0.0479 ± 0.0080