optica

Reliable deep-learning-based phase imaging with uncertainty quantification: supplementary material

YUJIA XUE, SHIYI CHENG, YUNZHE LI, AND LEI TIAN*

Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215, USA
**leitian@bu.edu*

Published 7 May 2019

This document provides supplementary information to "Reliable deep learning-based phase imaging with uncertainty quantification," https://doi.org/10.1364/optica.6.000618. We provide more details including: the Bayesian Neural Network (BNN) implementation, a comparison between two implementations of uncertainty quantification including the Dropout network and Deep Ensembles, data acquisition, data preprocessing, examples of phase unwrapping, comparison of the phase reconstructed from the model-based multiplexed Fourier ptychographic microscopy algorithm and the proposed BNN, how training FOV affects predicted uncertainty, cross validation on unseen cell types and unseen experimental setups, and comparisons of phase predicted from the proposed BNN and convolutional neural network (CNN).

1. BNN IMPLEMENTATION

Our BNN follows the U-Net architecture and is modified to perform uncertainty learning. The input takes five 384×384 pixel images. The output predicts both the mean and standard deviation pixel-by-pixel in two channels, both with size 384×384 pixels. The downsampling is done by 2×2 maxpooling. The upsampling is done by 2×2 upsampling followed by a convolutional layer. The denseblock [1] is used to facilitate efficient training. All denseblocks have three inner layers and a growth rate 16. Our preprocessed data contain negative values. Correspondingly, we use LeakyRelu [2] with slope 0.2 for all the activation functions of the inner layers. We use the sigmoid activation function in the final layer to normalize the predictions between 0 and 1. The final results (both the mean phase and the standard deviation) are linearly scaled back to the original unit (rad). To achieve high resolution enhancement, we further adapt the generative adversarial network (GAN) by introducing an additional discriminator network. We use the PatchGAN [3] training procedure.

We train the BNN with an initial learning rate 10^{-4} and gradually decrease the rate when the loss plateaus until the learning rate reaches 10^{-7}. The batch size is two in all experiments set by the memory limit of the GPU. All the data processing and network training are implemented in Python using TensorFlow/Keras library. The training is done on Boston University Shared Computing Cluster using one Nvidia Tesla P100 GPU.

In Tab. S1 and Tab. S2, we provide the details of our customized U-Net architecture with denseblock modules. The table includes the input and output shapes, and the details of each layer (including the number of filters, filter size, stride and activation functions). We have also made our implementation open source along with pre-trained weights and test sample data on our GitHub project page [4].

To process the datasets from Hela cells fixed in ethanol and formalin, we adopted the Generative Adversarial Network (GAN) for better learning the high-resolution cellular features. The Generator architecture is described in Tab. S1 and Tab. S2. The Discriminator network is detailed in Tab. S3. We do not consider the uncertainty in the Discriminator network.

2. COMPARISON BETWEEN DROPOUT NETWORK AND DEEP ENSEMBLES

Here, we present the uncertainty predictions using the dropout network (DO) and deep ensembles (DE) on Hela (fixed in ethanol) dataset. We observe that both two approaches converge to very similar results. In DO, the predictions results are obtained by activating the dropout layers in the prediction stage and take 16 output ensembles. In DE, we independently trained 8 identical networks using the same procedure. The variations in the output are resulted from the random weight

Table S1. Detailed implementation of our network structure. Notations: B: batch size, N: number of kernels, K: kernel size, S: stride

Layer name	Input shape	Output shape	Comments
$Inputlayer$	None	$B \times 384 \times 384 \times 5$	None
$Conv2D_1$	$B \times 384 \times 384 \times 5$	$B \times 384 \times 384 \times 64$	$N64K3S1, LeakyReLU(0.2)$
DB_1	$B \times 384 \times 384 \times 64$	$B \times 384 \times 384 \times 112$	$denseblock$
$Pool_1$	$B \times 384 \times 384 \times 112$	$B \times 192 \times 192 \times 112$	$Maxpooling$
$Conv2D_2$	$B \times 192 \times 192 \times 112$	$B \times 192 \times 192 \times 128$	$N128K3S1, LeakyReLU(0.2)$
DB_2	$B \times 192 \times 192 \times 128$	$B \times 192 \times 192 \times 176$	$denseblock$
$Pool_2$	$B \times 192 \times 192 \times 176$	$B \times 96 \times 96 \times 176$	$Maxpooling$
$Conv2D_3$	$B \times 96 \times 96 \times 176$	$B \times 96 \times 96 \times 256$	$N256K3S1, LeakyReLU(0.2)$
DB_3	$B \times 96 \times 96 \times 256$	$B \times 96 \times 96 \times 304$	$denseblock$
$Pool_3$	$B \times 96 \times 96 \times 304$	$B \times 48 \times 48 \times 304$	$Maxpooling$
$Conv2D_4$	$B \times 48 \times 48 \times 304$	$B \times 48 \times 48 \times 512$	$N512K3S1, LeakyReLU(0.2)$
DB_4	$B \times 48 \times 48 \times 512$	$B \times 48 \times 48 \times 560$	$denseblock$
DO_4	$B \times 48 \times 48 \times 560$	$B \times 48 \times 48 \times 560$	$dropoutrate0.5$
$Pool_4$	$B \times 48 \times 48 \times 560$	$B \times 24 \times 24 \times 560$	$Maxpooling$
$Conv2D_5$	$B \times 24 \times 24 \times 560$	$B \times 24 \times 24 \times 1024$	$N1024K3S1, LeakyReLU(0.2)$
DB_5	$B \times 24 \times 24 \times 1024$	$B \times 24 \times 24 \times 1072$	$denseblock$
DO_5	$B \times 24 \times 24 \times 1072$	$B \times 24 \times 24 \times 1072$	$dropoutrate0.5$
$Up2d_5$	$B \times 24 \times 24 \times 1072$	$B \times 48 \times 48 \times 512$	$N512K2S1, LeakyReLU(0.2)$
$Concatenate_5$	$B \times 48 \times 48 \times 512$	$B \times 48 \times 48 \times 1072$	$Up2d_5 and DB_4$
$Conv2D_6$	$B \times 48 \times 48 \times 1072$	$B \times 48 \times 48 \times 512$	$N512K3S1, LeakyReLU(0.2)$
DB_6	$B \times 48 \times 48 \times 512$	$B \times 48 \times 48 \times 560$	$denseblock$
$Up2d_6$	$B \times 48 \times 48 \times 560$	$B \times 96 \times 96 \times 256$	$N256K2S1, LeakyReLU(0.2)$
$Concatenate_6$	$B \times 96 \times 96 \times 256$	$B \times 96 \times 96 \times 560$	$Up2d_6 and DB_3$
$Conv2D_7$	$B \times 96 \times 96 \times 560$	$B \times 96 \times 96 \times 256$	$N256K3S1, LeakyReLU(0.2)$
DB_7	$B \times 96 \times 96 \times 256$	$B \times 96 \times 96 \times 304$	$denseblock$
$Up2d_7$	$B \times 96 \times 96 \times 304$	$B \times 192 \times 192 \times 128$	$N128K2S1, LeakyReLU(0.2)$
$Concatenate_7$	$B \times 192 \times 192 \times 128$	$B \times 192 \times 192 \times 304$	$Up2d_7 and DB_2$
$Conv2D_8$	$B \times 192 \times 192 \times 304$	$B \times 192 \times 192 \times 128$	$N128K3S1, LeakyReLU(0.2)$
DB_8	$B \times 192 \times 192 \times 128$	$B \times 192 \times 192 \times 176$	$denseblock$
$Up2d_8$	$B \times 192 \times 192 \times 176$	$B \times 384 \times 384 \times 64$	$N64K2S1, LeakyReLU(0.2)$
$Concatenate_8$	$B \times 384 \times 384 \times 64$	$B \times 384 \times 384 \times 176$	$Up2d_8 and DB_1$
$Conv2D_9$	$B \times 384 \times 384 \times 176$	$B \times 384 \times 384 \times 64$	$N64K3S1, LeakyReLU(0.2)$
DB_9	$B \times 384 \times 384 \times 64$	$B \times 384 \times 384 \times 112$	$denseblock$
$Conv2D_{10}$	$B \times 384 \times 384 \times 112$	$B \times 384 \times 384 \times 16$	$N32K3S1, LeakyReLU(0.2)$
$Conv2D_{11}$	$B \times 384 \times 384 \times 16$	$B \times 384 \times 384 \times 2$	$N2K3S1, Softmax$

Table S2. Details on denseblock module. Notations: B: batch size, N: number of kernels, K: kernel size, S: stride, R: number of rows of input tensor, C: number of columns of input tensor, L: number of layers of input tensor.

Layer name	Input shape	Output shape	Comments
$Inputlayer$	None	$B \times R \times C \times L$	None
BN_1	$B \times R \times C \times L$	$B \times R \times C \times L$	$BatchNormalization$
$ReLU_1$	$B \times R \times C \times L$	$B \times R \times C \times L$	$ReLU$
$Conv2D_1$	$B \times R \times C \times L$	$B \times R \times C \times 16$	$N16K3S1$
DO_1	$B \times R \times C \times 16$	$B \times R \times C \times 16$	$dropoutrate0.5$
$Concatenate_1$	$B \times R \times C \times 16$	$B \times R \times C \times (L+16)$	$InputlayerandDO_1$
BN_2	$B \times R \times C \times (L+16)$	$B \times R \times C \times (L+16)$	$BatchNormalization$
$ReLU_2$	$B \times R \times C \times (L+16)$	$B \times R \times C \times (L+16)$	$ReLU$
$Conv2D_2$	$B \times R \times C \times (L+16)$	$B \times R \times C \times 16$	$N16K3S1$
DO_2	$B \times R \times C \times 16$	$B \times R \times C \times 16$	$dropoutrate0.5$
$Concatenate_2$	$B \times R \times C \times 16$	$B \times R \times C \times (L+32)$	$Concatenate_1andDO_2$
BN_3	$B \times R \times C \times (L+32)$	$B \times R \times C \times (L+32)$	$BatchNormalization$
$ReLU_3$	$B \times R \times C \times (L+32)$	$B \times R \times C \times (L+32)$	$ReLU$
$Conv2D_3$	$B \times R \times C \times (L+32)$	$B \times R \times C \times 16$	$N16K3S1$
DO_3	$B \times R \times C \times 16$	$B \times R \times C \times 16$	$dropoutrate0.5$
$Concatenate_3$	$B \times R \times C \times 16$	$B \times R \times C \times (L+48)$	$Concatenate_2andDO_3$

Table S3. Details on discriminator network. Notations: B: batch size, N: number of kernels, K: kernel size, S: stride. No zero padding in convolution layers.

Layer name	Input shape	Output shape	Comments
$Inputlayer$	None	$B \times 96 \times 96 \times 1$	Diced patches
$Conv2D_1$	$B \times 96 \times 96 \times 1$	$B \times 46 \times 46 \times 64$	$N64K5S2, LeakyReLU(0.2)$
$Conv2D_2$	$B \times 46 \times 46 \times 64$	$B \times 21 \times 21 \times 64$	$N64K5S2, LeakyReLU(0.2)$
$Flatten$	$B \times 21 \times 21 \times 64$	$B \times 28224$	$Flatten$
FC_1	$B \times 28224$	$B \times 512$	$fullyconnected(512), LeakyReLU(0.2)$
FC_2	$B \times 512$	$B \times 512$	$fullyconnected(512), LeakyReLU(0.2)$
DO_1	$B \times 512$	$B \times 512$	$dropoutrate0.4$
FC_3	$B \times 512$	$B \times 512$	$fullyconnected(512), LeakyReLU(0.2)$
DO_2	$B \times 512$	$B \times 512$	$dropoutrate0.4$
FC_4	$B \times 512$	$B \times 1$	$fullyconnected(1), Softmax$

initialization, dropout, and stochastic gradient descent training algorithm. The uncertainty maps are calculated following the procedure discussed in the main article.

3. DATA ACQUISITION

Our technique is tested on two LED array based computational microscope setups detailed in [5, 6] and five different types of biological samples. We capture intensity measurements using both sequential and our five multiplexed illumination patterns.

On setup [5], we collect data on unstained Hela cells prepared with two fixation conditions, including ethanol and formalin. All images are captured using a $4\times$, 0.1 NA objective (Nikon CFI Plan Achromat). The ethanol fixed Hela data contains 22 full-FOV measurements. The formalin fixed Hela data contains 19 full-FOV measurements. Each group consists of the multiplexed data (2 brightfield and 3 darkfield images) and the corresponding sFPM data (185 images). Both the multiplexed and sFPM data are captured with the same 0.41 illumination NA, providing 0.51 NA final resolution.

We validate our technique on the data from [6]. The multiplexed measurements are synthesized by summing the sFPM images. We experimentally validate this procedure on setup [5] and find the numerically synthesized multiplexed intensity closely match with the physically captured measurement since the LEDs are spatially and temporally *incoherent*. We test our method on both *fixed* (U2OS, and MCF10A cells) and *dynamic* (live Hela cells) biological samples. The data from fixed U2OS cells and live Hela cells were captured with a $4\times$, 0.2 NA objective (CFI Plan Apo Lambda), and 0.6 illumination NA, that provide 0.8 final NA . The data from fixed MCF10A cells were captured with a $4\times$, 0.2 NA objective (CFI Plan Apo Lambda), and 0.5 illumination NA, that provide 0.7 final NA . The fixed U2OS data contains a single full FOV measurement. The fixed MCF10A data contains a single full FOV measurement. The dynamic Hela cell data contains a time series experiment over the course of 4 hours consisting of 120 full-FOV frames taken at 2 min intervals. Each dataset contains synthesized multiplexed (2 brightfield and 3 darkfield images) and corresponding sFPM data.

4. DATA PREPARATION

We discuss the data preparation in the following three parts: 1) ground truth phase calculation, 2) training and testing data separation, and 3) training and testing data preprocessing.

Distinct from computer vision applications, biomedical microscopy is often lack of ground truth that can only be approximated by alternative methods. Here, we adopt the strategy in [7] and use the phase reconstructed from sFPM as the ground truth. We use the algorithm in [8] to perform sFPM reconstruction. In practice, sFPM reconstructed phase contains several types of artifacts originated from phase wrapping, model mis-calibration, and noise propagation in the phase-retrieval algorithm, which have to be carefully taken into account in our data analysis.

First, due to the fixation, the cells contain phase values larger than 2π, so the raw sFPM phase map exhibits many phase wrapping artifacts. We use the algorithm in [9] to unwrap the phase. Examples from this procedure are detailed in Sec. 5.

The unwrapped phase typically suffers from a slowly varying background artifact. Our *"ground truth phase" preprocessing* first corrects for this using the morphological opening algorithm, and generate the background removed phase y^{raw}. Next, we perform *dynamic range correction* by choosing a threshold τ satisfying

$$P\{y_i^{\text{raw}} < \tau | \text{ for all } y_i^{\text{raw}} \in y^{\text{raw}}\} = 0.999, \tag{S1}$$

which sets the corrected phase to be within $[0, \tau]$ and clips the 0.1% pixels having extreme values to be τ. Next, the phase is linearly normalized to $[0, 1]$ by dividing the image by τ. The phase map is then cropped into small patches for training. Still, the unwrapped phase contains residual isolated errors typically around large-phase or complex cellular features. This results in *incorrect "phase labels"* in the training data that later affects the prediction. The impact of *incorrect labels* and *phase clipping* to the UL are analyzed in the main text.

Second, spatially varying aberrations [10] and illumination mis-calibration [11] are the main source of model mis-calibration induced errors in both sFPM and our multiplexed measurements. To perform high-quality sFPM reconstruction, we first calibrate the illumination angles using the algorithm in [11] prior to the reconstruction, and then digitally correct for the aberrations using the algorithm in [8].

In contrast, our BNN does *not* directly take any calibration information when constructing the network. The BNN learns the spatially varying imaging model indirectly from information contained in the multiplexed intensity measurements and the matching ground truth phase. We design our experiment to test the robustness of our technique against these spatially varying model errors, as well as the predictive power of UL to out-of-distribution data. During the training, data from only a limited FOV region are given to the BNN. By doing so, the aberration and misalignment information from the rest of the FOV has never been seen by the BNN. These correspond to out-of-distribution data that *should* lead to larger uncertainty in a well-calibrated neural network [12]. During the testing, data that have never been used in the training and from the entire FOV are used. The results are detailed in the main text.

Third, even after careful model calibrations, the sFPM phase still inevitably contain reconstruction noise [13], which we termed the intrinsic phase noise. Following [6], we measure the standard deviation in the background region and treat it as the intrinsic phase noise. We assume that the same noise level is uniformly distributed also across the sample (e.g. cell) regions. This noise level sets the tightest credible interval our BNN can provide; the detailed analysis is presented in the main text.

To pre-process the intensity measurements, first we remove a constant background estimated from the histograms. Negative values in the darkfield images are set to zero since they are primarily from shot noise [8, 13]. Second, we perform dynamic range correction (same as ground truth phase preprocessing). Third, we divide the full FOV into small patches. Next, we perform cubic interpolation on the intensity patches so that the input to the BNN has the same pixel numbers as the ground truth phase. For *training*, the matching phase and intensity patches are fed into the BNN. For *testing*, we apply additional corrections to intensity patches from the *untrained* FOV region to alleviate the out-of-distribution effect. For an intensity patch P_i from the *untrained* region, we perform *mean equalization* to match the mean with that from the *trained* region:

$$\tilde{P}_i = P_i \frac{\mu_{P_{\text{trained}}}}{\mu_{P_i}} \text{ for } P_i \in \text{untrained region}, \tag{S2}$$

where μ_{P_i} and $\mu_{P_{\text{trained}}}$ denote the intensity mean from the untrained and trained region, respectively. We find this procedure is essential to improve the BNN's generalization to spatially

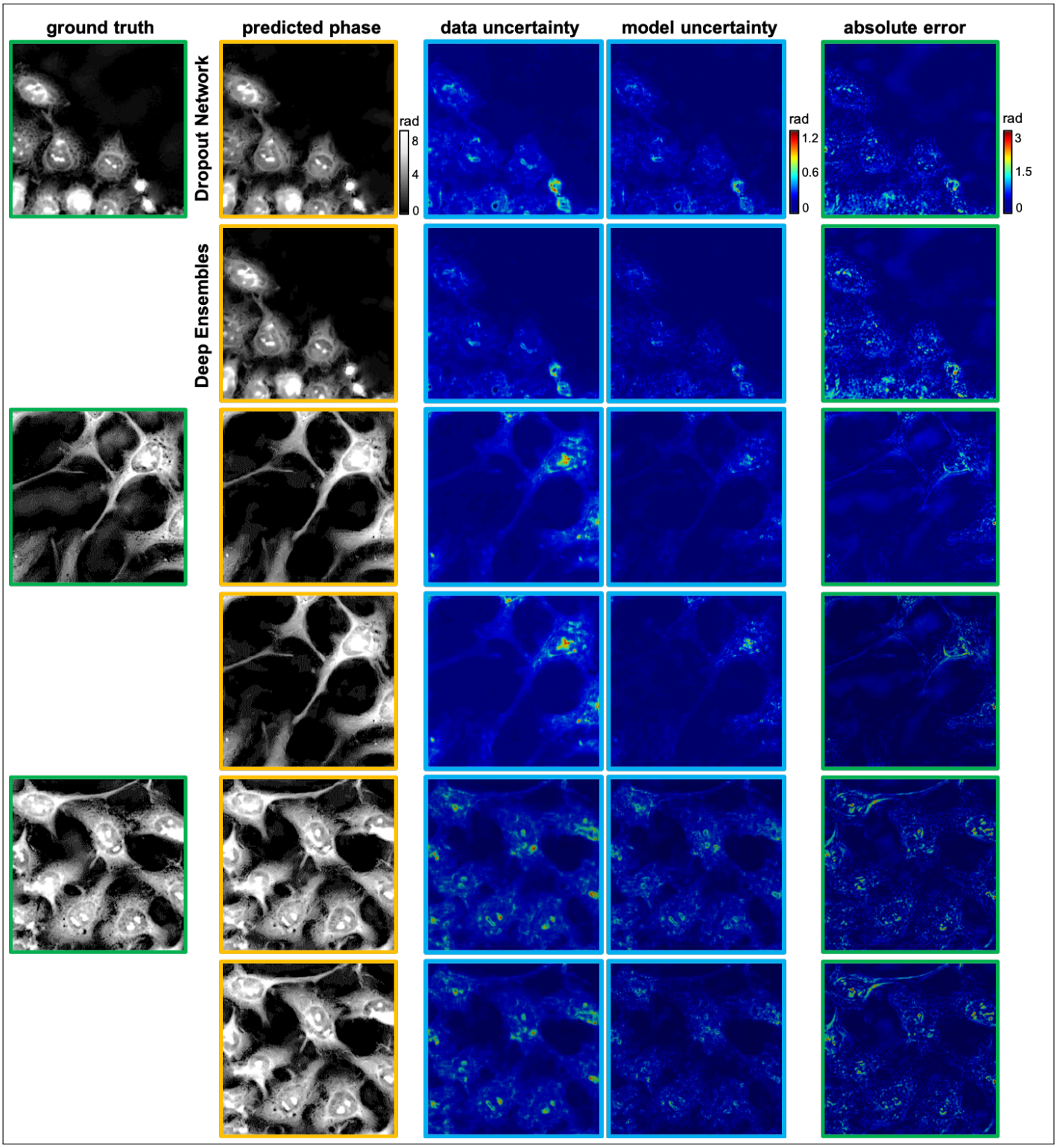

Fig. S1. Comparison between two BNN implementations: Dropout network and Deep ensembles. Both implementations give similar uncertainty quantification results and match with prediction error well.

varying model errors. However, mean equalization only compensate for the mismatch in the *first order* statistics. As a result, we expect the BNN should still predict larger uncertainty in the *untrained* regions, as demonstrated in the main text.

5. PHASE UNWRAPPING EXAMPLES

Our samples from the Hela cells fixed in ethanol and in formalin have a large phase range larger than 2π. The raw phase reconstructed from sFPM contain significant amount of phase wrapping artifacts. Before feeding into network for training, the phase images are unwrapped using a least-square based algorithm [9]. Fig. S2 illustrates a few representative examples before and after the phase unwrapping.

6. COMPARISON BETWEEN ALGORITHMS

Because of the highly multiplexed illumination scheme, the underlying inverse problem is highly nonlinear and ill-posed. In Fig. S3, we compare the phase prediction from the proposed BNN with two state-of-the-art model-based algorithms, including the linear differential phase contrast (DPC) model [14] and the multiplexed Fourier Ptychographic microscopy (mFPM) algorithm [6]. Our results show that DPC can only reconstruct phase with limited resolution. The mFPM can recover slightly more high frequency details by taking into account the dark-field measurements; however, the results also suffer from many high frequency artifacts. Our BNN can consistently provide high-quality high resolution phase reconstructions. In terms of the processing speeds, the time for processing a 384×384-pixel patch is 6 seconds for the mFPM algorithm and 0.12 second for our BNN to make 16 prediction ensembles.

7. EFFECT OF TRAINING SAMPLE TYPE AND SETUP

In Fig. S4, we provide additional examples of the BNN under different training and testing configurations. The multiplexed intensity measurements were synthesized using the data from [6]. In total, we test three different cases. First, the training and testing under the same cell type and the same illumination NA are marked in blue, as shown in the images along the main diagonal in Fig. S4. Second, the training and testing under different cell types and the same illumination NA are marked in green. Finally, the training and testing under both different cell types and different illumination NAs are marked in orange. We see that the networks trained and tested on the same cell type can provide high-quality phase prediction and uncertainty quantification. When tested on unseen cell type data under the same illumination NA, the networks are still able to make high-quality phase maps, with slight degradation in the reconstructed high-frequency features. When tested on unseen cell types and using a different illumination NA, the phase predictions degrade further. In certain cases, severe artifacts, such as contrast reversal can be observed. The reason why the large degradation of the phase reconstruction using data captured from a different illumination NA is because our BNN is trained to solve a specific inverse problem with pre-defined physical parameters. As the illumination NA changes, it changes the underlying physical model. As a result, the BNN produces less accurate phase predictions. Importantly, in all cases, the predicted uncertainty maps can still detect and identify the potential errors in the phase predictions, consistent with the true absolute error.

In Fig. S5, we further test if the network trained on the data from one setup is able to make reliable predictions using the data taken from a different setup. It is observed that the predicted phase further degrades and contains severe artifacts. This is expected since our BNN is trained to solve a particular inverse problem. In our case, the two setups contain intensities measured using different objective NAs (0.1 vs 0.2) and different illumination NAs (0.41 vs 0.6). As a result, the intensity measurements taken from one setup will result in severe out-of-distribution artifacts when they are input to the BNN trained on a different setup. Nevertheless, the uncertainty map remains highly indicative to the prediction errors and are useful to detect abnormalities in the data.

8. EFFECT OF TRAINING FOV

In the main article, we show that by training only in the central FOV, the trained network is able to identify degradations in the phase predictions made on the untrained outer FOV regions. This shows that our BNN is able to quantify the effect of the out-of-distribution data due to limited FOV. Here, we provide additional examples to further support this proposition. We train multiple BNNs using data taking from different sub-FOV regions and then evaluate the performance on both seen and unseen FOV regions. The two cases shown in Fig. S6 and Fig. S7 show that both the phase prediction error and the BNN predicted uncertainty maps have higher values in the unseen FOV regions. Quantitatively, we calculate the uncertainty level and the absolute error from the seen and unseen FOVs under four different training FOV configurations, including left, top, center and outer FOV. Tab. S4 clearly demonstrates that the BNN consistently predicts higher uncertainty level in the unseen FOV regions.

9. COMPARISON BETWEEN BNN AND CNN

We compare prediction results using our BNN framework and using the standard convolutional neural network (CNN). In Fig. S8, we first demonstrate that when trained and tested on the same cell type, the BNN and CNN provide almost identical results. The small difference in the phase predictions are expected due to the intrinsic variations discussed in the main text. In Fig. S9, we further compare prediction result when the networks are tested on unseen cell types. The BNN and CNN show comparable generalizability to variations in the sample types. The main benefit of using BNN comes from its ability to provide uncertainty quantification without the need for the ground truth.

REFERENCES

1. G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Conference on Computer Vision and Pattern Recognition*, (2017), pp. 2261–2269.
2. B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," arXiv:1505.00853 (2015).
3. P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," CoRR **abs/1611.07004** (2016).
4. Y. Xue, S. Cheng, Y. Li, and L. Tian, "https://github.com/bu-cisl/illumination-coding-meets-uncertainty-learning," .

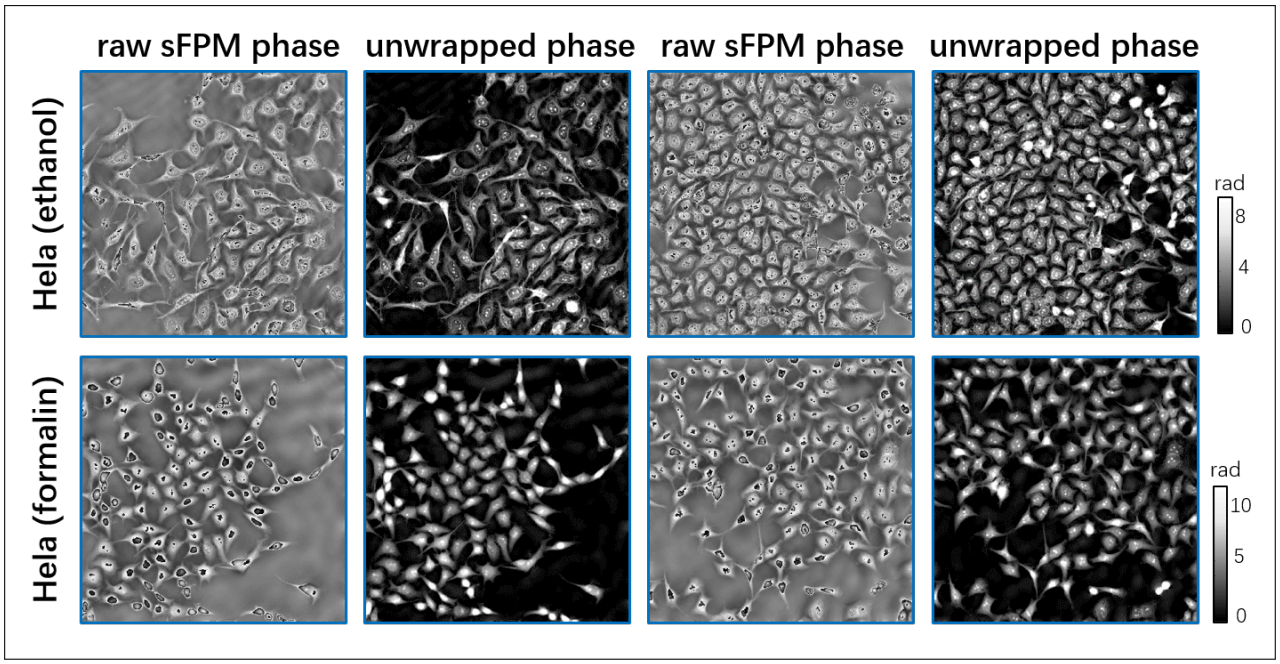

Fig. S2. Examples of phase unwrapping. Groundtruth phase images are unwrapped before training.

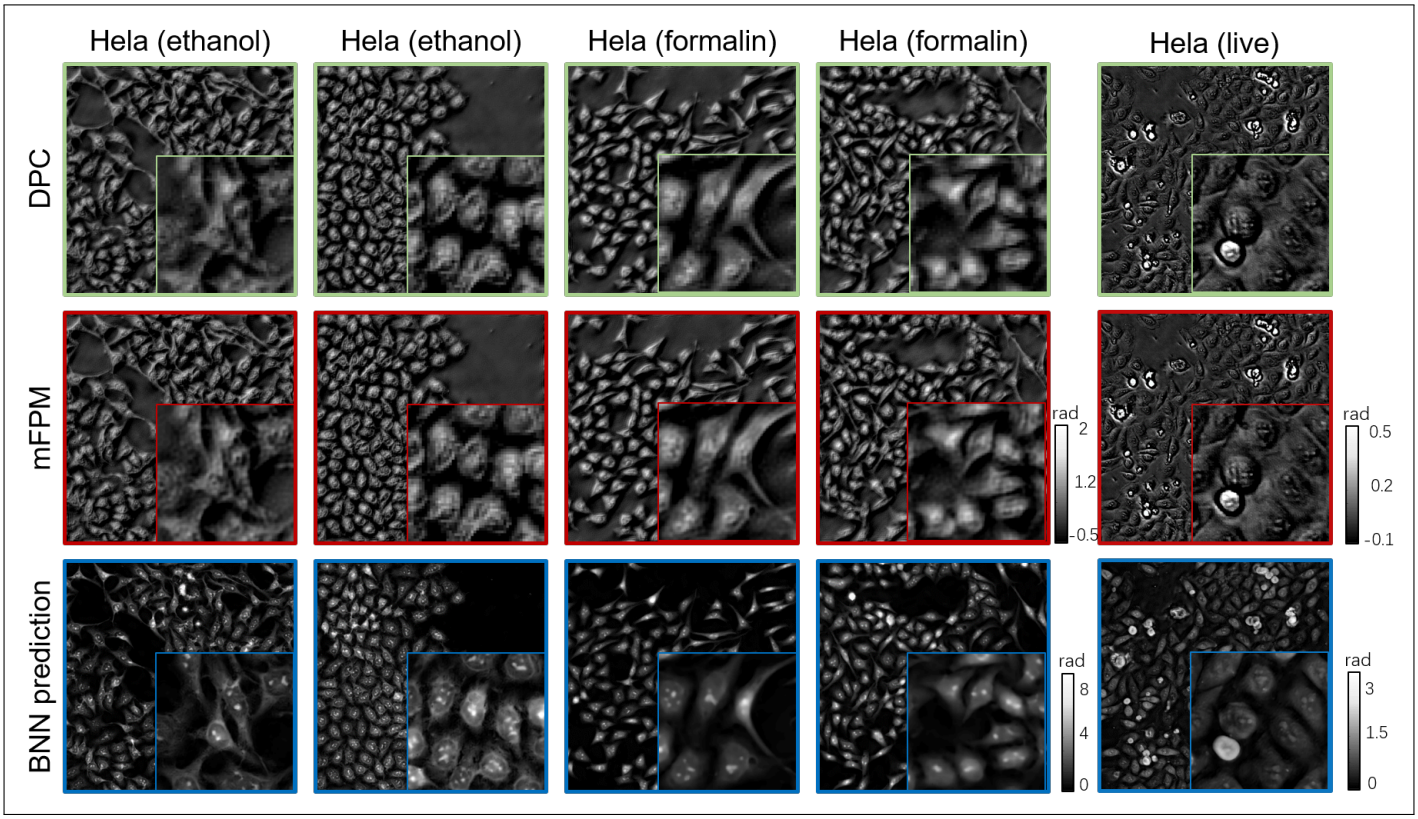

Fig. S3. Phase reconstruction results from the differential phase contrast, multiplexed Fourier ptychographic microscopy algorithm, and our BNN using intensity measurements from the proposed illumination scheme.

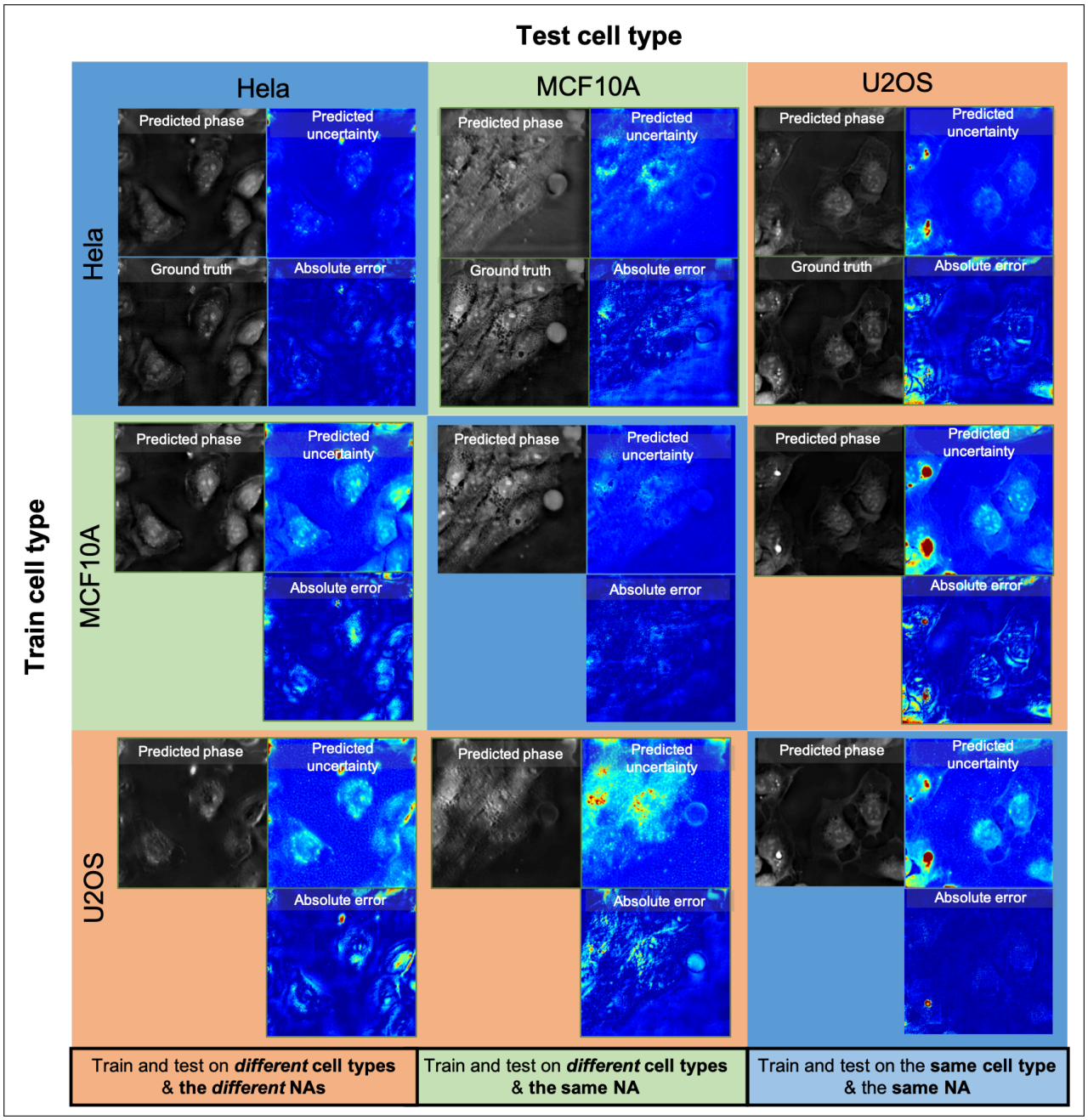

Fig. S4. Prediction results under different training and testing data configurations. Orange: the training and testing under different cell types and different illumination NAs. Green: the training and testing under different cell types and the same illumination NA. Blue: the training and testing under the same cell type and the same illumination NA.

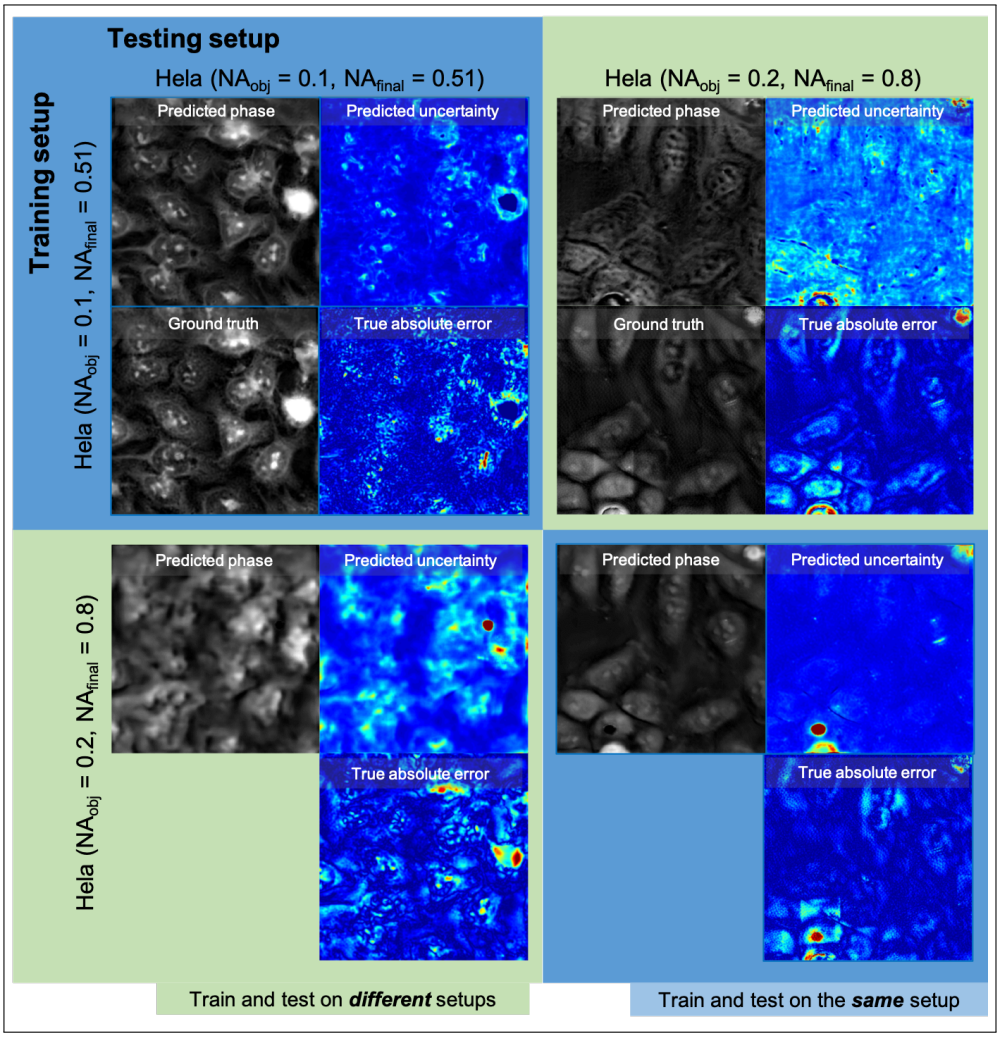

Fig. S5. Prediction results using data from different setups.

Table S4. Predicted uncertainty level on seen and unseen FOVs under different training FOVs.

Training FOV:	Left	Top	Center	Outer
Seen FOV	0.0483	0.0510	0.0387	0.0576
Unseen FOV	0.0585	0.0576	0.0465	0.0570

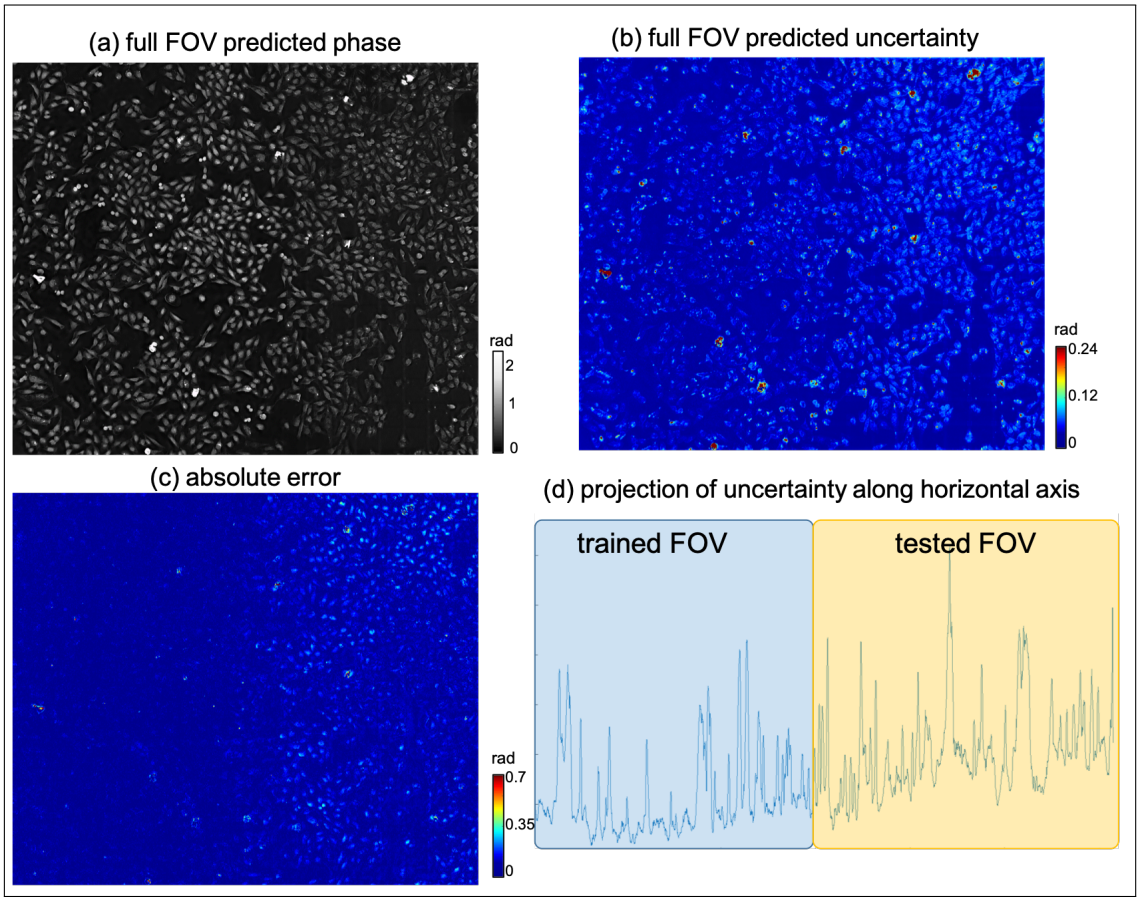

Fig. S6. Network trained on the left-half of the FOV and tested on the entire FOV. Both the phase prediction error and the predicted uncertainty are higher on the unseen right-half of the FOV region.

5. R. Ling, W. Tahir, H.-Y. Lin, H. Lee, and L. Tian, "High-throughput intensity diffraction tomography with a computational microscope," Biomed. Opt. Express **9**, 2130 (2018).
6. L. Tian, Z. Liu, L.-H. Yeh, M. Chen, J. Zhong, and L. Waller, "Computational illumination for high-speed in vitro Fourier ptychographic microscopy," Optica **2**, 904–911 (2015).
7. T. Nguyen, Y. Xue, Y. Li, L. Tian, and G. Nehmetallah, "Deep learning approach for Fourier ptychography microscopy," Opt. Express **26**, 26470 (2018).
8. L. Tian, X. Li, K. Ramchandran, and L. Waller, "Multiplexed coded illumination for Fourier ptychography with an LED array microscope," Biomed. Opt. Express **5**, 2376–2389 (2014).
9. D. C. Ghiglia and L. A. Romero, "Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods," J. Opt. Soc. Am. A **11**, 107 (1994).
10. X. Ou, G. Zheng, and C. Yang, "Embedded pupil function recovery for Fourier ptychographic microscopy," Opt. Express **22**, 4960–4972 (2014).
11. R. Eckert, Z. F. Phillips, and L. Waller, "Efficient illumination angle self-calibration in Fourier ptychography," Appl. Opt. **57**, 5434 (2018).
12. V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," arXiv:1807.00263 (2018).
13. L.-H. Yeh, J. Dong, J. Zhong, L. Tian, M. Chen, G. Tang, M. Soltanolkotabi, and L. Waller, "Experimental robustness of Fourier ptychography phase retrieval algorithms," Opt. Express **23**, 33214–33240 (2015).
14. L. Tian and L. Waller, "Quantitative differential phase contrast imaging in an LED array microscope," Opt. Express **23**, 11394–11403 (2015).

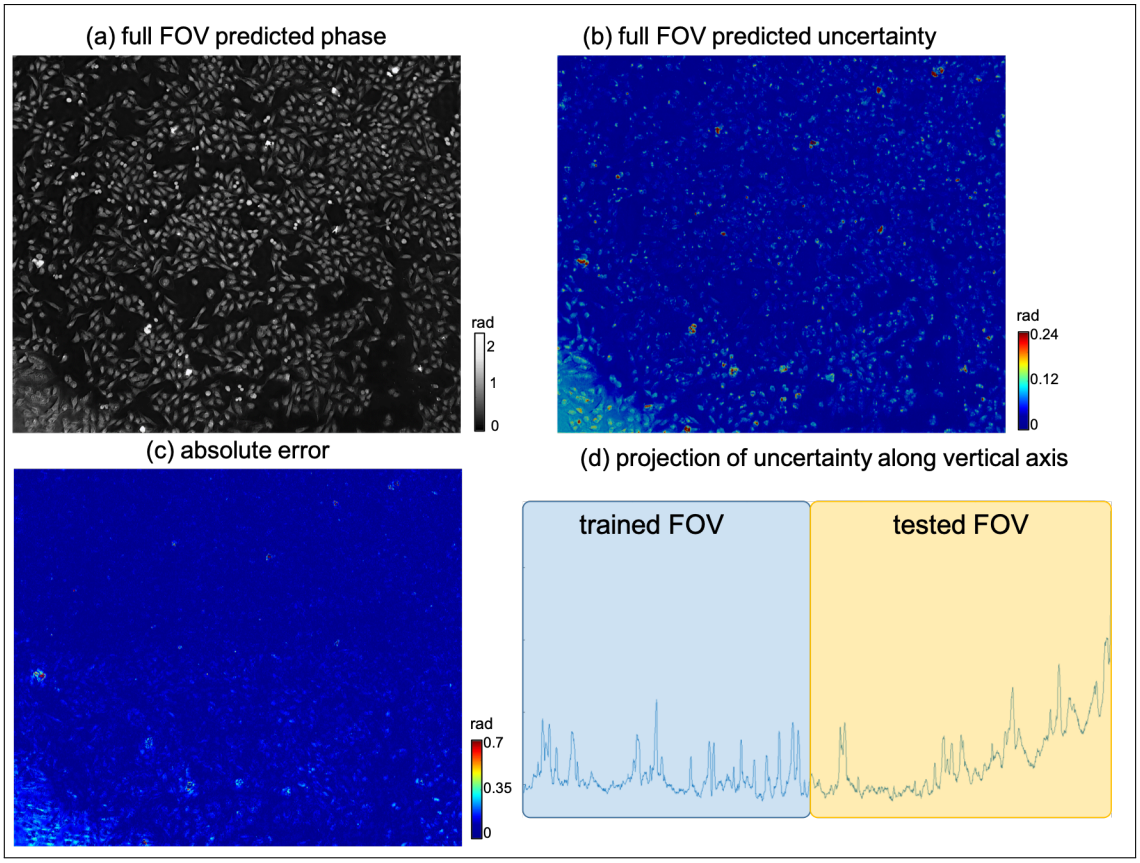

Fig. S7. Network trained on the top-half of the FOV and tested on the entire FOV. Both the phase prediction error and the predicted uncertainty are higher on the unseen bottom-half of the FOV region.

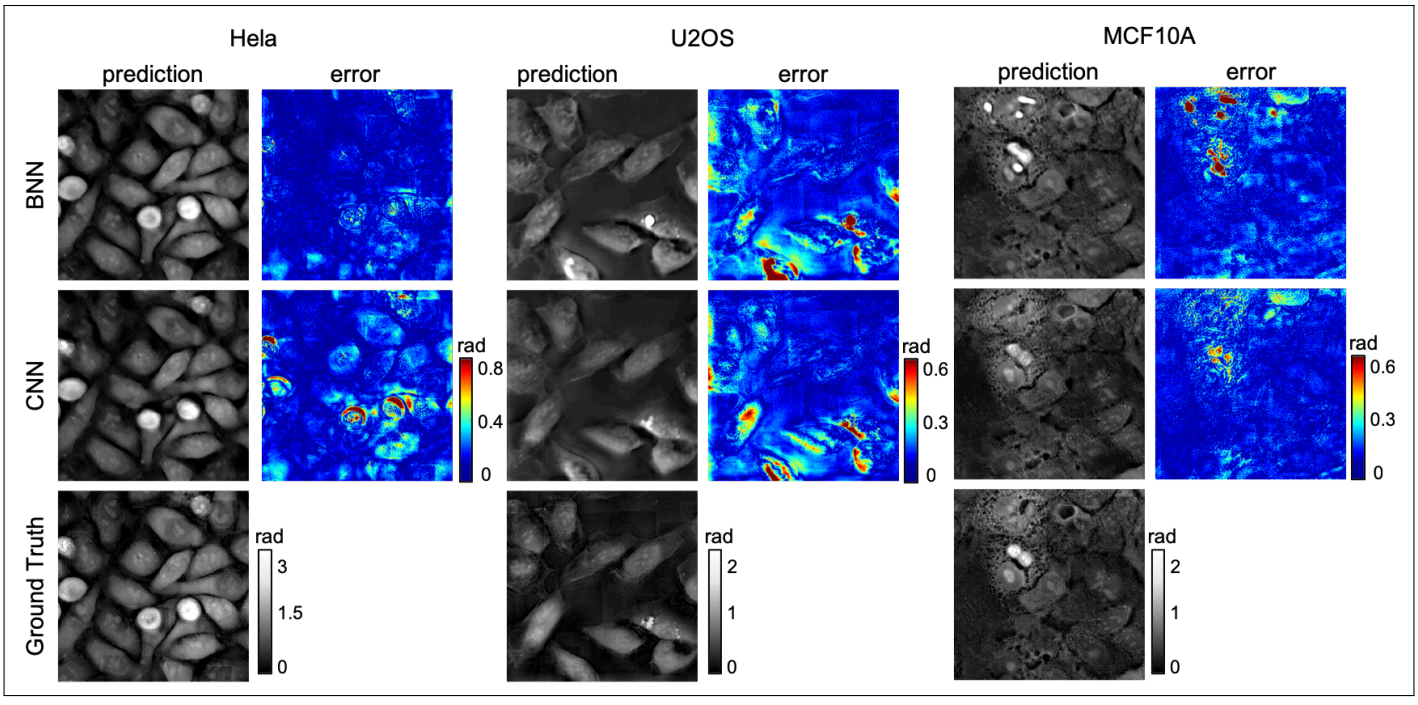

Fig. S8. Comparison between our BNN and standard CNN when trained and tested on the same cell type.

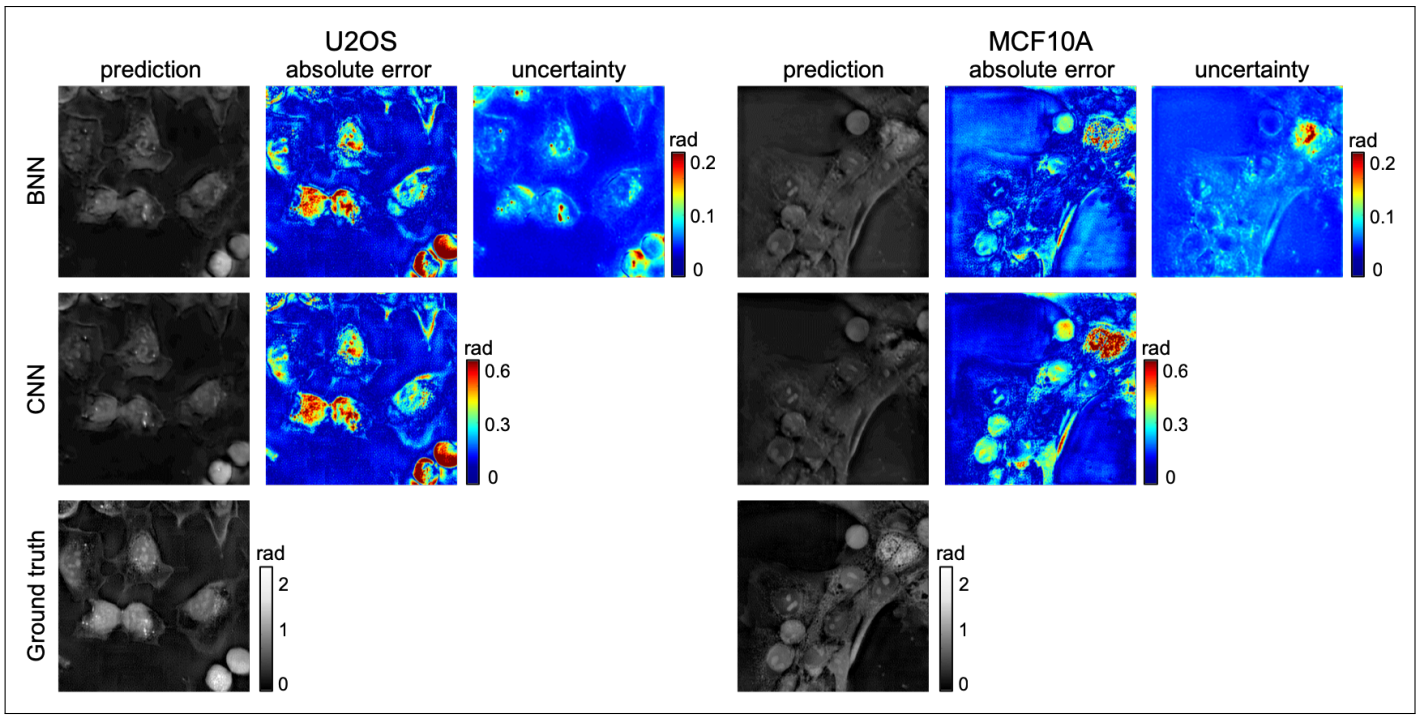

Fig. S9. Comparison between our BNN and standard CNN when tested on unseen cell types. Both network were trained on Hela cells and tested on U2OS and MCF10A cells.