optica

Deep learning microscopy: supplementary material

YAIR RIVENSON,[1] ZOLTÁN GÖRÖCS,[1,2,3,†] HARUN GÜNAYDIN,[1,†] YIBO ZHANG,[1,2,3] HONGDA WANG,[1,2,3] AYDOGAN OZCAN,[1,2,3,4,*]

[1]*Electrical Engineering Department, University of California, Los Angeles, CA 90095, USA*
[2]*Bioengineering Department, University of California, Los Angeles, CA 90095, USA*
[3]*California NanoSystems Institute (CNSI), University of California, Los Angeles, CA 90095, USA*
[4]*Department of Surgery, David Geffen School of Medicine, University of California, Los Angeles, CA 90095, USA*
[†]*These authors contributed equally to the paper*
**Corresponding author: ozcan@ucla.edu*

Published 20 November 2017

This document provides supplementary information to "Deep learning microscopy," https://doi.org/10.1364/optica.4.001437.

https://doi.org/10.6084/m9.figshare.5552338

1. DEEP LEARNING NETWORK ARCHITECTURE

The schematics of the architecture for training our deep neural network is depicted in Fig. S1. The input images are mapped into 3 color channels: red, green and blue (RGB). The input convolutional layer maps the 3 input color channels, into 32 channels, as depicted in Fig. S2. The number of output channels of the first convolutional layer was empirically determined to provide the optimal balance between the deep neural network's size (which affects the computational complexity and image output time) and its image transform performance. The input convolutional layer is followed by K=5 residual blocks [1]. Each residual block is composed of 2 convolutional layers and 2 rectified linear units (ReLU) [2,3], as shown in Fig. S1. The ReLU is an activation function which performs $\mathrm{ReLU}(x) = \max(0, x)$. The formula of each residual block can be summarized as:

$$X_{k+1} = X_k + \mathrm{ReLU}(\mathrm{ReLU}(X_k * W_k^{(1)}) * W_k^{(2)}), \quad \textbf{(S1)}$$

where $*$ refers to convolution operation, X_k is the input to the k-th block, X_{k+1} denotes its output, $W_k^{(1)}$ and $W_k^{(2)}$ denote an ensemble of learnable convolution kernels of the k-th block, where the bias terms are omitted for simplicity. The output feature maps of the convolutional layers in the network are calculated as follows:

$$g_{k,j} = \sum_i f_{k,i} * w_{k,i,j} + \beta_{k,j}\mathbf{\Omega}, \quad \textbf{(S2)}$$

where $w_{k,i,j}$ is a learnable 2D kernel (i.e., the (i,j)-th kernel of W_k) applied to the i-th input feature map, $f_{k,i}$ (which is an M×M-pixel image in the residual blocks), $\beta_{k,j}$ is a learnable bias term, $\mathbf{\Omega}$ is an M×M matrix with all its entries set as 1, and $g_{k,j}$ is the convolutional layer j-th output feature map (which is also an M×M-pixel image in the residual blocks). The size of all the kernels (filters) used throughout the network's convolutional layers is 3×3. To resolve the dimensionality mismatch of Eq. (2), prior to convolution, the feature map $f_{k,i}$ is zero-padded to a size of (M+2)×(M+2) pixels, where only the central M×M-pixel part is taken following the convolution with kernel $w_{k,i,j}$.

To allow high level feature inference we increase the number of features learnt in each layer, by gradually increasing the number of channels, using the pyramidal network concept [4]. Using such pyramidal networks helps to keep the network's width compact in comparison to designs that sustain a constant number of channels throughout the network. The channel increase formula was empirically set according to [4]:

$$A_k = A_{k-1} + floor((\alpha \times k) / K + 0.5) \quad \textbf{(S3)}$$

where $A_0 = 32$, k=[1:5], which is the residual block number, K=5 is the total number of residual blocks used in our architecture and α is a constant that determines the number of channels that will be added at each residual block. In our implementation, we used α=10, which yields $A_5 = 62$ channels at the output of the final residual block. In addition, we utilized the concept of residual connections (shortcutting the block's input to its output, see Fig. S1), which was demonstrated to improve the training of deep neural networks by providing a clear path for information flow [3] and speed up the convergence of the training phase. Nevertheless,

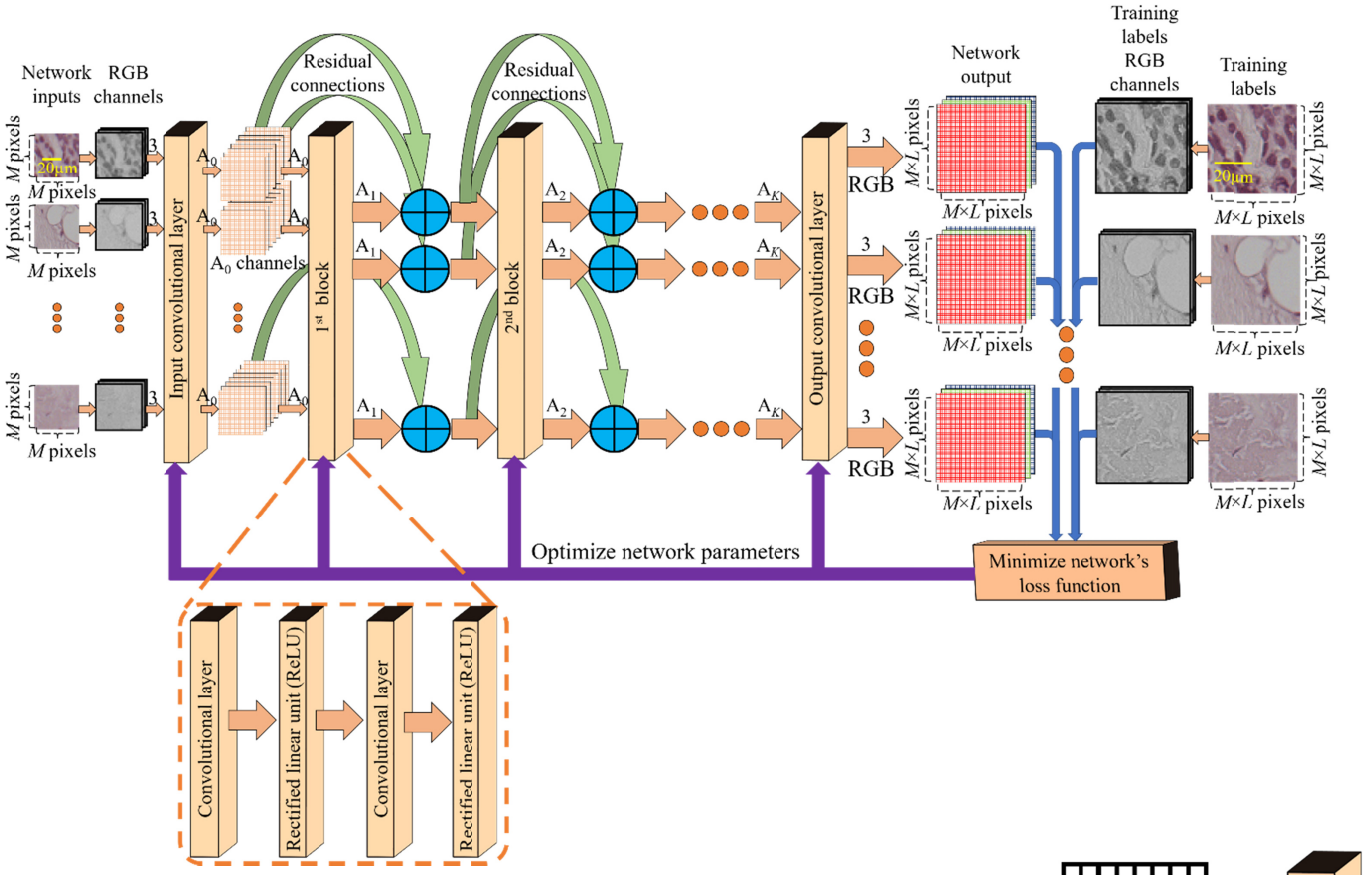

Fig. S1. Detailed schematics of the deep neural network training phase.

increasing the number of channels at the output of each layer leads to a dimensional mismatch between the inputs and outputs of a block, which are element-wise summed up in Eq. (1). This dimensional mismatch is resolved by augmenting each block's input channels with zero valued channels, which virtually equalizes the number of channels between a residual block input and output.

In our experiments, we have trained the deep neural network to extend the output image space-bandwidth-product by a non-integer factor of L^2=2.5^2=6.25 compared to the input images. To do so, first the network learns to enhance the input image by a factor of 5×5 pixels followed by a learnable down-sampling operator of 2×2, to obtain the desired L=2.5 factor (Fig. S3). More specifically, at the output of the K-th residual block $A_K = A_5 = 62$ channels are mapped to $3\times5^2 = 75$ channels (Fig. S3), followed by resampling of these 75 $(M\times M)$ pixels channels to three channels with $(M\times5)\times(M\times5)$ pixels grid [5,6]. These three $(M\times5)\times(M\times5)$ pixels channels are then used as input to an additional convolutional layer (with learnable kernels and biases, as the rest of the network), that two-times down-samples these images to three $(M\times2.5)\times(M\times2.5)$ color pixels. This is performed by using a two-pixel stride convolution, instead of a single pixel stride convolution, as performed throughout the other convolutional layers of the network. This way, the network learns the optimal down-sampling procedure for our microscopic imaging task. It is important to note that during the testing phase, if the number of input pixels to the network is odd, the resulting number of output image pixels will be determined by the ceiling operator. For instance, a 555×333-pixel input image will result in a 1388×833-pixel image for L=2.5.

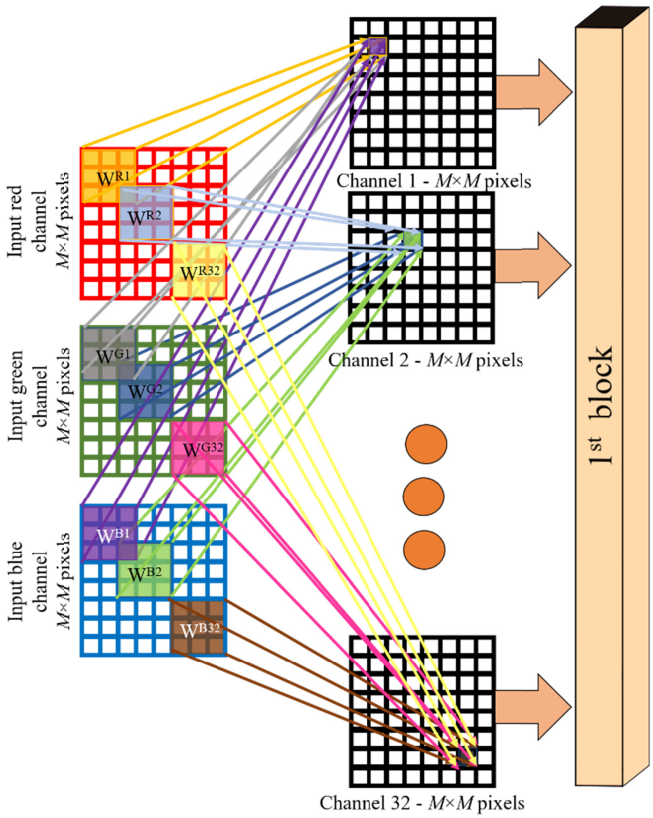

Fig. S2. Detailed schematics of the input layer of the deep neural network.

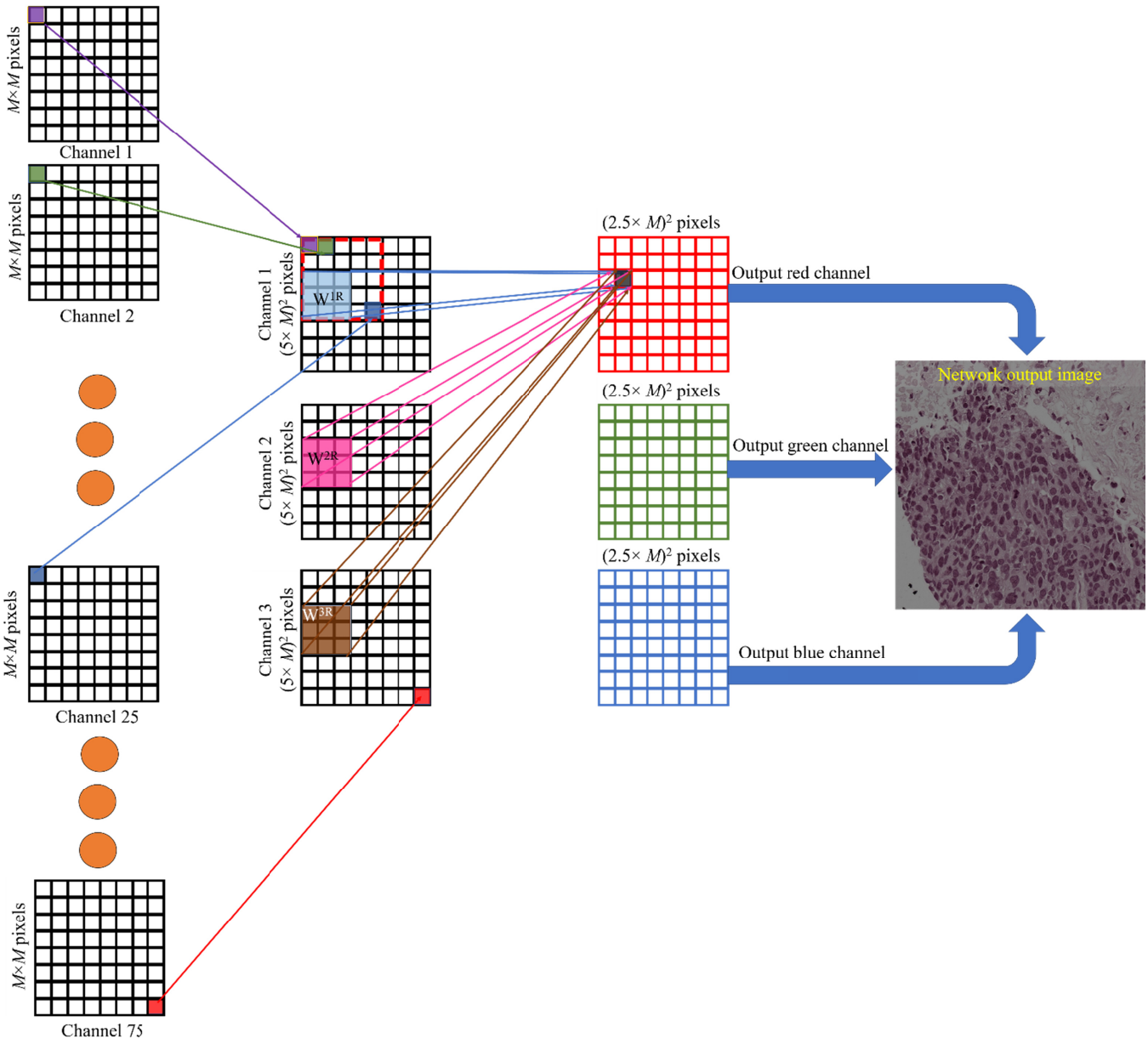

Fig. S3. Detailed schematics of the output layer of the deep neural network for L=2.5.

The above-discussed deep network architecture provides two major benefits: first, the up-sampling procedure becomes a learnable operation with supervised learning, and second, using low resolution images throughout the network's layers makes the time and memory complexities of the algorithm L^2 times smaller [6] when compared to approaches that up-sample the input image as a precursor to the deep neural network. This has a positive impact on the convergence speed of both the training and image transformation phases of our network.

2. DATA PRE-PROCESSING

To achieve optimal results, the network should be trained with accurately aligned low-resolution input images and high-resolution label image data. We match the corresponding input and label image pairs using the following steps: (A) Color images are converted to grayscale images. (B) A large field-of-view image is formed by stitching a set of low resolution images. (C) Each high-resolution label image is down-sampled (bicubic) by a factor L. This down-sampled image is used as a template image to find the highest correlation matching patch in the low-resolution stitched image. The highest correlating patch from the low-resolution stitched image is then digitally cropped. This cropped low-resolution image and the original high-resolution image, form an input-label pair, which is used for the network's training and testing. (D) Additional alignment is then performed on each of the

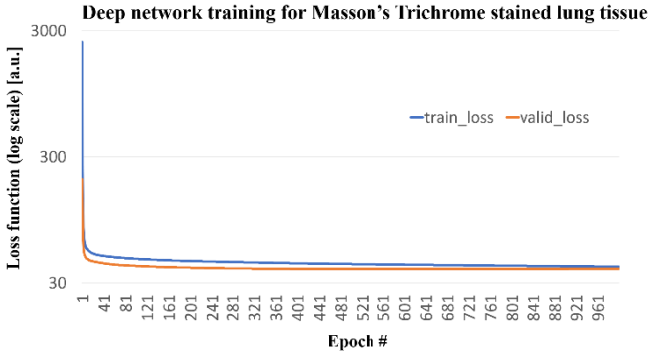

Fig S4. Training and validation dataset errors as a function of the number of epochs for the Masson's trichrome stained lung tissue dataset.

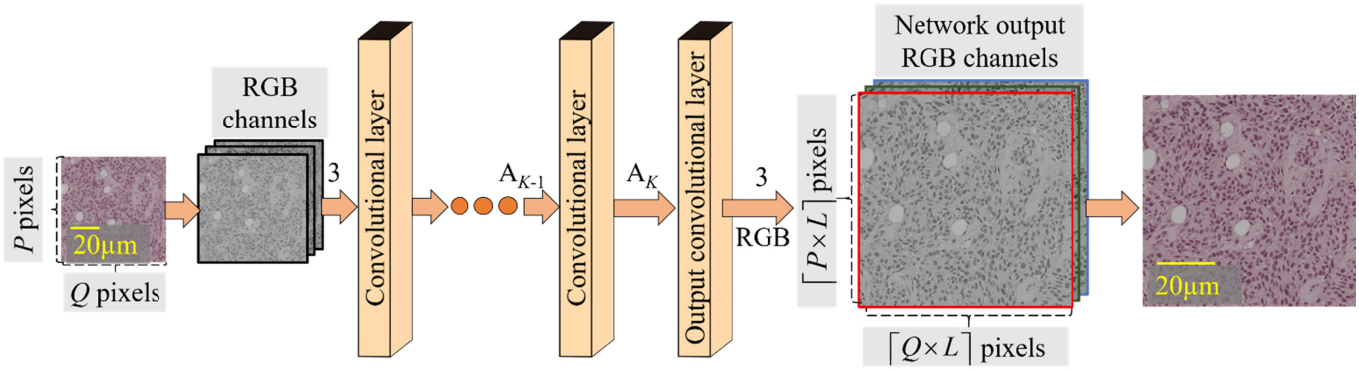

Fig. S5. Detailed schematics of the deep neural network high-resolution image inference (i.e., the testing phase).

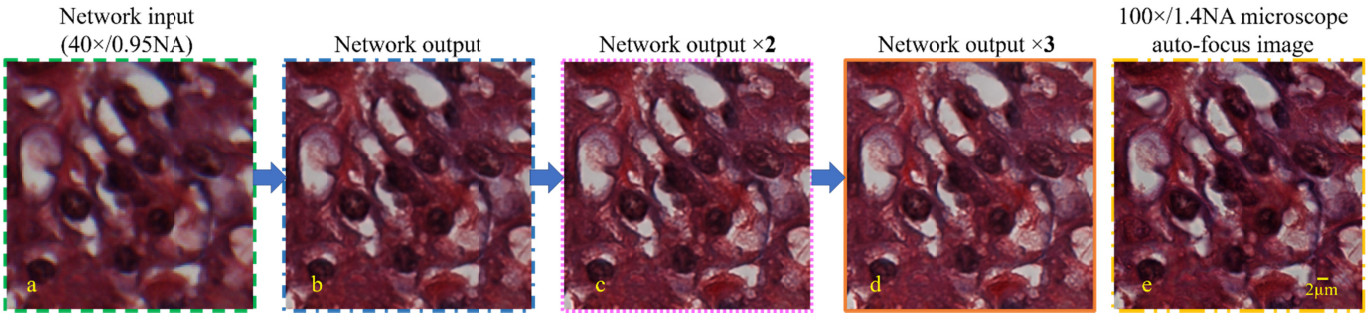

Fig. S6. Result of applying the deep neural network in a cyclic manner on Masson's trichrome stained kidney section images. (**a**), Input image acquired with a 40×/0.95NA objective lens. The deep neural network is applied on this input image once, twice and three times, where the results are shown in (**b**, **c)** and (**d**), respectively. (**e**), 100×/1.4NA image of the same field-of-view is shown for comparison.

input-label pairs to further refine the input-label matching, mitigating rotation, translation and scaling discrepancies between the lower resolution and higher resolution images. This step was performed by matching Speeded-Up Robust Features (SURF) [7], implemented using Matlab [8].

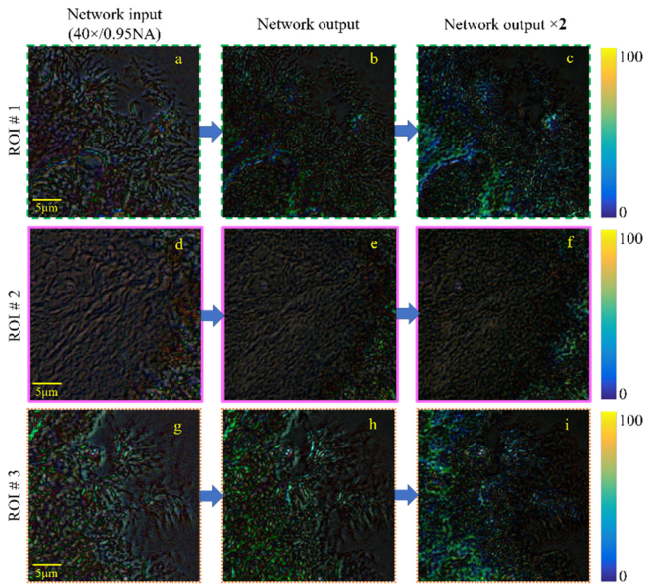

Fig S7. The percentage of the pixel-level differences for the network input or output images calculated with respect to the gold standard images captured using a 100×/1.4NA objective lens. ROIs correspond to the Masson's trichrome stained lung tissue shown in Fig. 2 of the main text. The colorbar spans 0-100%.

3. NETWORK TRAINING

The network was trained by optimizing the following loss function (ℓ) given the high-resolution training labels Y^{HR}:

$$\ell(\Theta)=\frac{1}{3\times M^2\times L^2}\sum_{c=1}^{3}\sum_{u=1}^{M\times L}\sum_{v=1}^{M\times L}\left\|Y^{\Theta}_{c,u,v}-Y^{HR}_{c,u,v}\right\|_2^2+ \\ +\lambda\frac{1}{3\times M^2\times L^2}\sum_{c=1}^{3}\sum_{u=1}^{M\times L}\sum_{v=1}^{M\times L}\left|\nabla Y^{\Theta}\right|^2_{c,u,v}, \quad \textbf{(S4)}$$

where $Y^{\Theta}_{c,u,v}$ and $Y^{HR}_{c,u,v}$ denote the u,v-th pixel of the c-th color channel (where in our implementation we use three color channels, RGB) of the network's output image and the high resolution training label image, respectively. The network's output is given by $Y^{\Theta}=F(X^{LR}_{input};\Theta)$, where F is the deep neural network's operator on the low-resolution input image X^{LR}_{input} and Θ is the network's parameter space (e.g., kernels, biases, weights). Also, $(M\times L)\times(M\times L)$ is the total number of pixels in each color channel, λ is a regularization parameter, empirically set to ~0.001. $\left|\nabla Y^{\Theta}\right|^2_{c,u,v}$ is u,v-th pixel of the c-th color channel of the network's output image gradient [9], applied separately for each color channel, which is defined as: $\left|\nabla Y^{\Theta}\right|^2=\left(h*Y^{\Theta}\right)^2+\left(h^T*Y^{\Theta}\right)^2$, with:

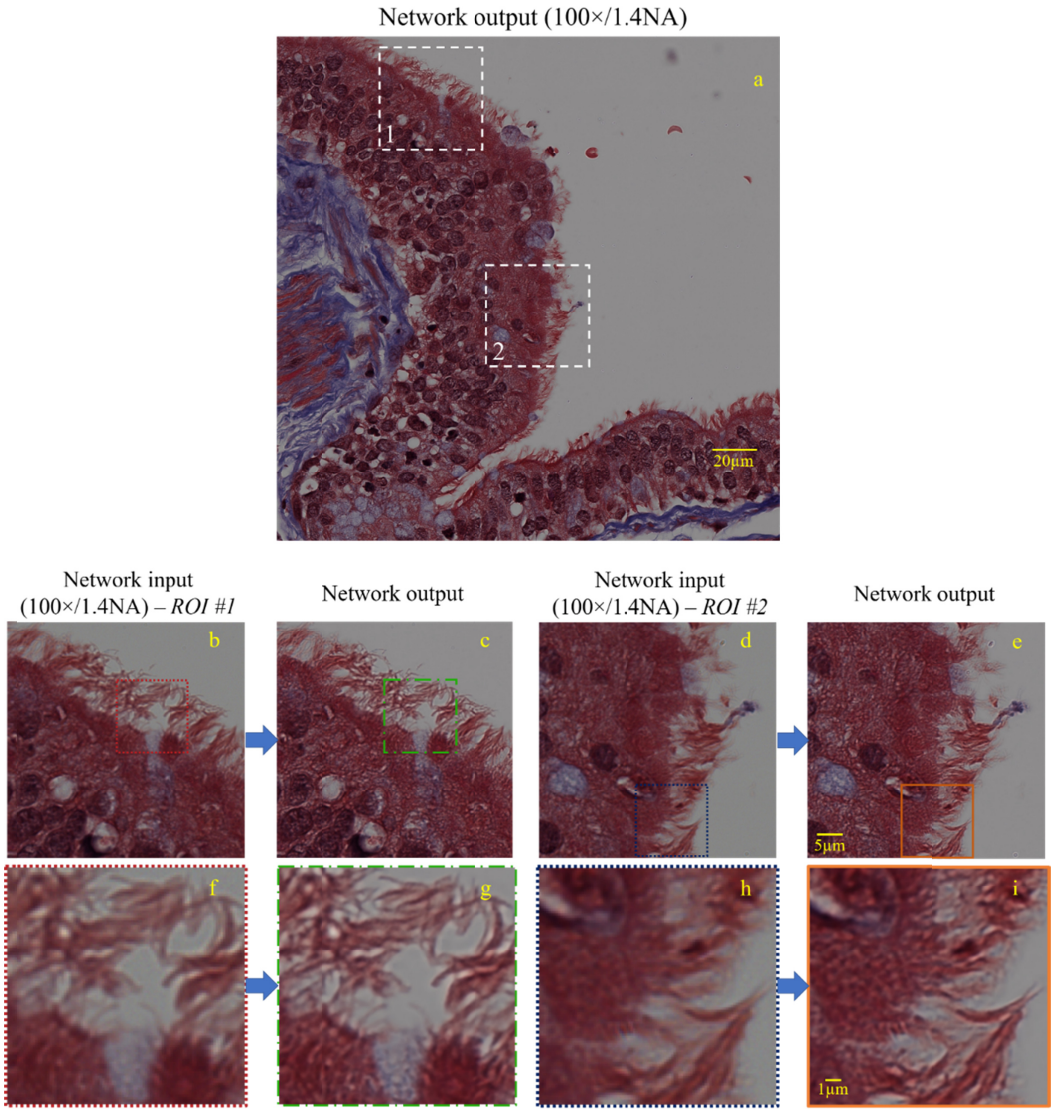

Fig. S8. Deep neural network output image corresponding to a Masson's trichrome stained lung tissue section taken from a pneumonia patient. The network was trained on images of a Masson's trichrome stained lung tissue taken from a different tissue block that was not used as part of the CNN training phase. (**a**), Image of the deep neural network output corresponding to a 100×/1.4NA input image. (**b**, **f**, **d**, **h**) Zoomed-in ROIs of the input image (100×/1.4NA). (**c**, **g**, **e**, **i**) Zoomed-in ROIs of the neural network output image.

$$h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad \textbf{(S5)}$$

and $(.)^T$ refers to the matrix transpose operator.

The above defined loss function balances between the mean-squared-error (MSE) and the image sharpness with a regularization parameter, λ. The MSE is used as a data fidelity term and the l_2-norm image gradient approximation helps mitigating the spurious edges that result from the pixel up-sampling process. Following the estimation of the loss function, the error is backpropagated through the network, and the network's parameters are learnt by using the Adaptive Moment Estimation (ADAM) optimization [10], which is a stochastic optimization method, that we empirically set a learning rate parameter of 10^{-4} and a mini-batch size of 64 image patches. All the kernels (for instance $w_{k,i,j}$) used in convolutional layers have 3×3 elements and their entries are initialized using truncated normal distribution with 0.05 standard deviation and 0 mean [1] All the bias terms (for instance, $\beta_{k,j}$) are initialized with 0.

Table S1. Average structural similarity index (SSIM) for the Masson's trichrome stained lung tissue and H&E stained breast tissue datasets, comparing bicubic up-sampling and the deep neural network output.

	Test set	Bicubic up-sampling SSIM	Deep neural network SSIM
Masson's trichrome stained lung tissue	20 images (224×224 pixels)	0.672	0.796
H&E stained breast tissue	7 images (660×660 pixels)	0.685	0.806

Table S2. Deep neural network training details for the Masson's trichrome stained lung tissue and H&E stained breast tissue datasets.

	Number of input-output patches (number of pixels for each low-resolution image)	Validation set (number of pixels for each low-resolution image)	Number of epochs till convergence	Training time
Masson's trichrome stained lung tissue	9,536 patches (60×60 pixels)	10 images (224×224 pixels)	630	4hr, 35min
H&E stained breast tissue	51,008 patches (60×60 pixels)	10 images (660×660 pixels)	460	14hr, 30min

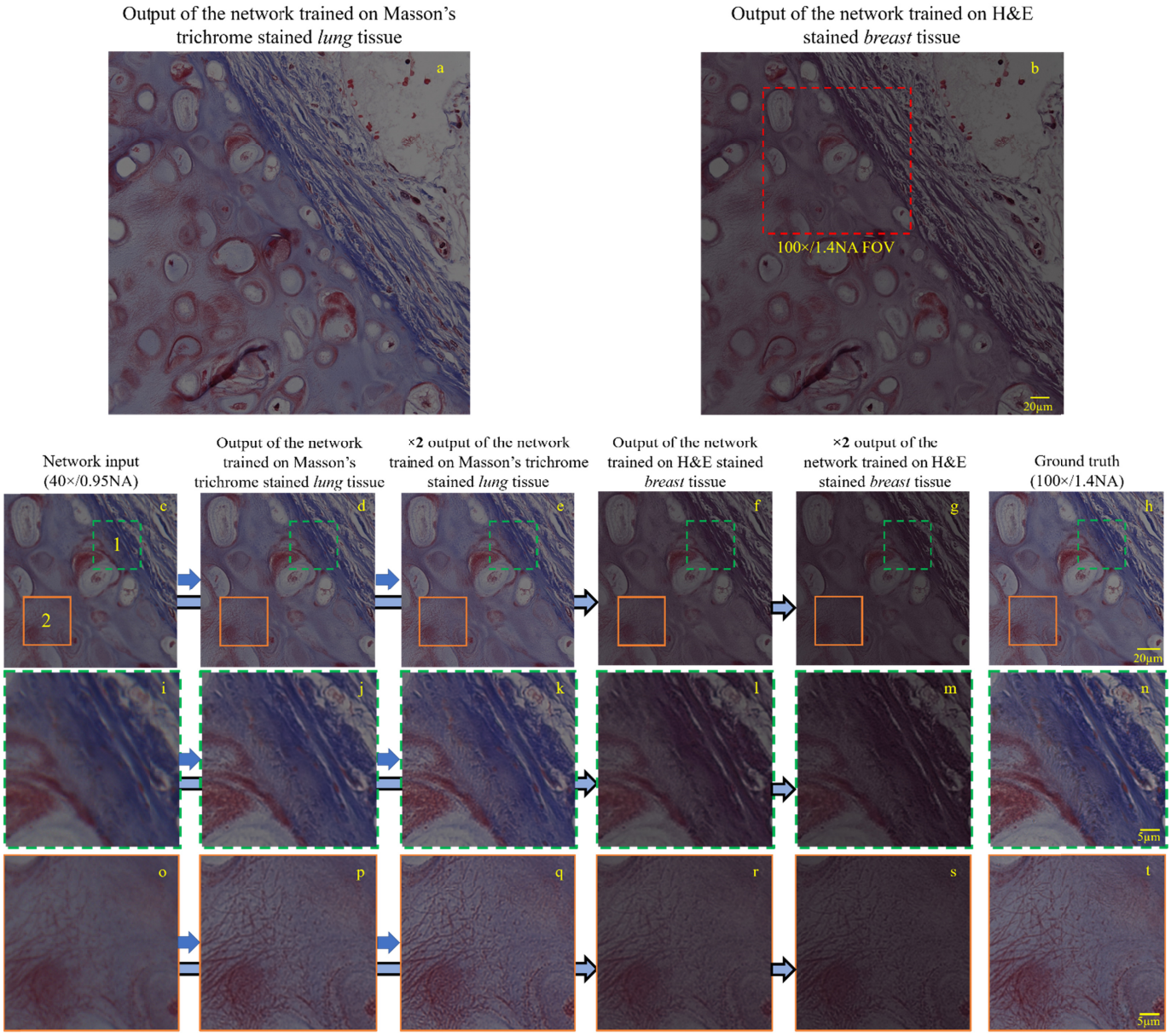

Fig. S9. (**a**), Result of applying the *lung* tissue trained deep neural network model on a 40×/0.95NA *lung* tissue input image. (**b**), Result of applying the *breast* tissue trained deep neural network model on a 40×/0.95NA *lung* tissue input image. (**c**, **i**, **o**) Zoomed in ROIs corresponding to the 40×/0.95NA input image. (**d**, **j**, **p**) Neural network output images, corresponding to input images (**c**, **i**) and (**o**), respectively; the network is trained with lung tissue images. (**e**, **k**, **q**) Neural network output images, corresponding to input images (**d**, **j**), and (**p**), respectively; the network is trained with lung tissue images. (**f**, **l**, **r**) Neural network output images, corresponding to input images (**c**, **i**) and (**o**), respectively; the network is trained with breast tissue images stained with a different dye, H&E. (**g**, **m**, **s**) Neural network output images, corresponding to input images (**f**, **l**), and (**r**), respectively; the network is trained with breast tissue images stained with H&E. (**h**, **n**, **t**) Comparison images of the same ROIs acquired using a 100×/1.4NA objective lens.

4. NETWORK TESTING

A fixed network architecture, following the training phase is shown in Fig. S5, which receives an input of $P\times Q$-pixel image and outputs a $\lceil (P\times L)\rceil\times\lceil (Q\times L)\rceil$-pixel image, where $\lceil . \rceil$ is the ceiling operator. To numerically quantify the performance of our trained network models, we independently tested it using

Table S3. Average runtime for different regions-of-interest shown in Fig. 2.

		Single GPU runtime (*sec*)		Dual GPU runtime (*sec*)	
Image FOV	Number of Pixels (input)	Network Output	Network Output x2 (Self-feeding)	Network Output	Network Output x2 (Self-feeding)
378.8 × 378.8 μm (e.g., Fig. 2A)	2048×2048	1.193	8.343	0.695	4.615
151.3 × 151.3 μm (e.g., red box in Fig. 2A)	818×818	0.209	1.281	0.135	0.730
29.6 × 29.6 μm (e.g., Figs. 2B-L)	160×160	0.038	0.081	0.037	0.062

validation images, as detailed in Table S1. The output images of the network were quantified using the structural similarity index (SSIM) [11]. SSIM, which has a scale between 0 and 1, quantifies a human observer's perceptual loss from a gold standard image by considering the relationship among the contrast, luminance, and structure components of the image. SSIM is defined as 1 for an image that is identical to the gold standard image.

Table S4. Calculated contrast values for the USAF resolution test target elements.

Period (Cycles/mm)	100×/1.4NA input contrast (a.u.)	Network output contrast (a.u.)
256	0.801144658	0.75627907
287.3502844	0.790853855	0.729511618
322.5397888	0.790801661	0.72449378
362.038672	0.795555918	0.709122843
406.3746693	0.787270294	0.726269147
456.1401437	0.771249585	0.774461828
512	0.713336904	0.681675286
574.7005687	0.636153491	0.640392221
645.0795775	0.577148622	0.588478766
724.0773439	0.517576094	0.585453198
812.7493386	0.516547441	0.63392948
912.2802874	0.439410917	0.597450901
1024	0.368925748	0.585228416
1149.401137	0.400244051	0.53887823
1290.159155	0.303987367	0.496261545
1448.154688	0.228926978	0.4729755
1625.498677	0.20194026	0.542857143
1824.560575	0.12865681	0.454976303
2048	0.110901071	0.259743799
2298.802275	0	0.254320988
2580.31831	0	0.182785747
2896.309376	0	0.072
3250.997354	0	0
3649.12115	0	0

5. IMPLEMENTATION DETAILS

The program was implemented using Python version 3.5.2, and the deep neural network was implemented using TensorFlow framework version 0.12.1 (Google). We used a laptop computer with Core i7-6700K CPU @ 4GHz (Intel) and 64GB of RAM, running a Windows 10 professional operating system (Microsoft).

The network training and testing were performed using GeForce GTX 1080 GPUs (NVidia). For the training phase, using a dual-GPU configuration resulted in ~33% speedup compared to training the network with a single GPU. The training time of the deep neural networks for the lung and breast tissue image datasets is summarized in Table S2 (for the dual-GPU configuration).

Following the conclusion of the training stage, the fixed deep neural network intakes an input stream of 100 low-resolution images each with 2,048×2,048-pixels, and outputs for each input image a 5,120×5,120-pixel high-resolution image at a total time of ~119.3 seconds (for all the 100 images) on a single laptop GPU. This runtime was calculated as the average of 5 different runs. Therefore, for L=2.5 the network takes 1.193 sec per output image on a single GPU. When employing a dual-GPU for the same task, the average runtime reduces to 0.695 sec per 2,048×2,048-pixel input image (see Table S3 for additional details on the network output runtime corresponding to other input image sizes, including self-feeding of the network output).

6. MODULATION TRANSFER FUNCTION (MTF) ANALYSIS

To quantify the effect of our deep neural network on the spatial frequencies of the output image, we have applied the CNN that was trained using the Masson's trichrome stained lung tissue samples on a resolution test target (Extreme USAF Resolution Target on 4×1 mm Quartz Circle Model 2012B, Ready Optics), which was imaged using a 100×/1.4NA objective lens, with a 0.55NA condenser. The objective lens was oil immersed as depicted in Fig. 5(a), while the interface between the resolution test target and the sample cover glass was not oil immersed, leading to an effective objective NA of ≤1 and a lateral diffraction limited resolution ≥ 0.354μm (assuming an average illumination wavelength of 550 nm). MTF was evaluated by calculating the contrast of different elements of the resolution test target [12] For each element, we horizontally averaged the resulting image along the element lines (~80-90% of the line length). We then located the center pixels of the element's minima and maxima and used their values for contrast calculation. To do that, we calculated the length of the element's cross-section from the resolution test target group and element number in micrometers, cut out a corresponding cross section length from the center of the horizontally averaged element lines. This also yielded the center pixel locations of the

element's local maximum values (2 values) and minimum values (3 values). The maximum value, $I_{\max}$, was set as the maximum of the local maximum values and the minimum value, $I_{\min}$, was set as the minimum of the local minimum values. For the elements, where the minima and maxima of the pattern matched their calculated locations in the averaged cross section, the contrast value was calculated as: $(I_{\max} - I_{\min})/(I_{\max} + I_{\min})$. For the elements where the minima and maxima were not at their expected positons, thus the modulation of the element was not preserved, we set the contrast to 0. Based on this experimental analysis, the calculated contrast values are given Table S4 and the MTFs for the input image and the output image of the deep neural network (trained on Masson's trichrome lung tissue) are compared to each other in Supplementary Fig. 5(e).

References

1. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in (2016), pp. 770–778.
2. K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15 (IEEE Computer Society, 2015), pp. 1026–1034.
3. K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, eds., Lecture Notes in Computer Science (Springer International Publishing, 2016), pp. 630–645.
4. D. Han, Kim, Jiwhan, and Kim, Junmo, "[1610.02915] Deep Pyramidal Residual Networks," https://arxiv.org/abs/1610.02915.
5. S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," IEEE Signal Process. Mag. **20**, 21–36 (2003).
6. W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in (2016), pp. 1874–1883.
7. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," Comput. Vis. Image Underst. **110**, 346–359 (2008).
8. "Find Image Rotation and Scale Using Automated Feature Matching - MATLAB & Simulink," https://www.mathworks.com/help/vision/examples/find-image-rotation-and-scale-using-automated-feature-matching.html.
9. A. Kingston, A. Sakellariou, T. Varslot, G. Myers, and A. Sheppard, "Reliable automatic alignment of tomographic projection data by passive auto-focus," Med. Phys. **38**, 4934–4945 (2011).
10. D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in (2014).
11. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," IEEE Trans. Image Process. **13**, 600–612 (2004).
12. J. Rosen, N. Siegel, and G. Brooker, "Theoretical and experimental demonstration of resolution beyond the Rayleigh limit by FINCH fluorescence microscopic imaging," Opt. Express **19**, 26249–26268 (2011).