北海道大学
HOKKAIDO UNIVERSITY

Applications of deep learning to accelerate label-free nerve imaging rate using coherent Raman rigid endoscopy – Construction of transfer learning method with fluorescence images and evaluation method for maximum imaging rate –

非線形ラマン硬性内視鏡による無標識神経イメージング高速化への深層学習の応用 – 蛍光画像を用いた転移学習法と最大撮像速度の評価法の構築 –

by Naoki Yamato

Doctoral Dissertation

Supervisor: Prof. Mamoru Hashimoto

Associate Supervisor: Prof. Hiroshi Uji-i
Prof. Hiroshi Hirata
Assoc. Prof. Nobuki Kudo

Laboratory for Biomedical Engineering
Course of Bioengineering and Bioinformatics
Division of Information Science and Technology
Graduate School of Information Science and Technology

Hokkaido University

Date 2023/3/23

abstract

An imaging system for nerve visualization/identification during surgery is expected to improve the quality of life after surgery. This is because postsurgical pain and body dysfunction are induced by nerve damage during surgery. Although nerve-sparing approaches based on anatomical knowledge and surgeon's experiments have been reported, it is unclear how well the peripheral nerves are preserved. The visibility of peripheral nerves is severely poor because peripheral nerves are transparent, colorless, and thin. Therefore, an intraoperative nerve identification tool is required to improve the nerve-sparing rate and evaluate the efficiency of nerve-sparing approaches. In this thesis, an imaging acceleration system for a coherent anti-Stokes Raman scattering (CARS) rigid endoscope is developed for nerve-sparing surgery. The nerves can be visualized in a label-free manner using CARS, which is sensitive to species of molecular vibration in samples. The imaging rate acceleration of the CARS rigid endoscope developed in this laboratory is needed for clinical application.

In this dissertation, the author shows the results of introducing image processing by deep learning to improve the nerve imaging rate of the CARS endoscope. The author developed the critical imaging rate (***CIR***) as a quantitative metric of imaging speed accelerated by deep learning. ***CIR*** is defined as the highest imaging rate to satisfy the image quality needed for medical images and is calculated from a harmonic mean of $\boldsymbol{CIR}_{\mathrm{PSNR}}$ and $\boldsymbol{CIR}_{\mathrm{SSIM}}$ which are the imaging rates satisfying criteria of PSNR=30 and SSIM=0.8, respectively. It is demonstrated that noise reduction of CARS endoscopic images made the imaging rate about five times faster using ***CIR*** (from 1.4 images/min to 7.0 images/min). Because noise reduction enhances

the quality of the image, it is possible to improve the imaging rate of CARS rigid endoscopy by enhancing the quality of noisy nerve images captured at high rates. In general, a large dataset is needed for training deep-learning models. However, it is time-consuming and laborious to prepare a large dataset. The author proposed transfer learning using CARS microscopic images before training models on CARS endoscopic images. CARS microscopy made it easier to search peripheral nerves because the CARS images with a high signal-to-noise ratio can be obtained at a high imaging rate using an objective lens with a higher numerical aperture. The author compared three denoising models (WIN5R, DenoiseNet, and Noise2Noise) to determine the optimal model for denoising of CARS nerve images and to conduct pre-training before fine-tuning with CARS endoscopic images. Noise2Noise showed the highest denoising performance between the three models quantitatively and qualitatively. After training on CARS microscopic images, the model was fine-tuned using a few CARS endoscopic images.

The author proposed a learning method to utilize images obtained with another modality as pre-training. Fluorescence microscopy was used as another modality to acquire a large dataset for nerve segmentation. Nerve segmentation is needed for displaying nerve areas to surgeons. The author prepared fluorescence images labeling lipids for pre-training nerve segmentation because CARS images have lipid information using $\mathbf{CH_2}$ vibration. U-Net, which is famous for semantic segmentation, was used. U-Net consists of an encoder part extracting feature maps from input images and a decoder part reconstructing output images from the feature maps. A VGG16 encoder, which learns to extract feature maps for image classification, was used in the encoder part of U-Net. It is found that both pre-training with fluorescence images and using the VGG16 encoder enhanced the segmentation performance significantly. In particular, pre-training with fluorescence images boosted the performance of the nerve segmentation from CARS images with a low signal-to-noise ratio. Therefore, the $\boldsymbol{CIR}_{segmentation}$ was improved about 47.5 times using semantic segmentation with deep learning than $\boldsymbol{CIR}_{denoising}$. Semantic segmentation of nerves in CARS endoscopic

images improved the imaging rate by about 47.5 times factor compared with denoising. These results open the opportunity for incorporating CARS rigid endoscopes into medical settings.

Contents

Introduction

Laparoscopic surgery (LS) has revolutionized surgery in terms of minimal invasiveness. Numerous surgeons have enthusiastically developed the procedure of LS since Mouret introduced it in 1987 [1]. LS is applied to many types of surgery (e. g. gastrectomy, hysterectomy and prostatectomy). In LS, a surgeon makes some holes for inserting laparoscopes and conducts dissection and excision while observing through endoscopic camera. LS benefits from lesser blood loss and shorter length of stay in the hospital compared to open surgery [2,3]. The oncological outcome of LS is similar to that of open surgery [3]. Therefore, LS is less invasive than open surgery.

In addition to laparoscopic surgery, nerve preservation is attracting attention to improve the prognosis after surgery. Nerve damage causes chronic postsurgical pain and body dysfunction [4–8]. Peripheral nerves are transparent, colorless, and thin. The preservation of peripheral nerves is based on the anatomical knowledge and experience of surgeons. However, the postsurgical symptoms which seem to be induced by nerve damage are reported [9, 10]. This reason is considered due to personal variability of peripheral nerve location. Furthermore, surgeons cannot confirm whether nerves are damaged during surgery or not because no surgical tools identifying nerve location exist. Thus, surgical support equipment identifying peripheral nerves is required for improving nerve preservation rate.

Various approaches for identifying peripheral nerves intraoperatively have been developed. Three types of approaches are introduced here; (1) electrical stimulation, (2) fluorescent dye, and (3) Label-free optical techniques. The first approach of an intraoperative electrical stimulation (IES) method can find out peripheral nerves by

observing the motion induced by nerve stimulation [11–13]. In this method, surgeons bring electrodes into contact with the tissue, apply a pulsed current, and monitor the mechanical response. This method allows the evaluation of nerve damage and the prediction of the postsurgical outcome but does not prevent nerve damage. The precise nerve location cannot be determined. Also, IES with the excess current has the risk of nerve damage [14]. It is desirable to identify peripheral nerve locations directly for nerve preservation. The second approach of a fluorescent dye is developed to label and visualize peripheral nerves specifically [15–20]. The molecular weight of fluorescent dye is designed to be less than 500 Da to reach nerves through the blood-brain barrier and blood-nerve barrier. Many animal experiences using mice, rats, and pigs are demonstrated. Both of central nervous system and peripheral nervous system are visualized. However, penetration of the blood-brain barrier is a concern for safe clinical use [21]. The third approach of label-free optical techniques is expected to be suitable for nerve identification in terms of invasiveness [22–24]. Raman scattering spectroscopy can recognize peripheral nerves from surrounding tissues in a label-free manner using the Raman spectrum, which is sensitive to molecular composition and chemical bonds. It takes a long time to obtain the Raman spectrum because the efficiency of Raman scattering is enormously low. Therefore, it can't acquire 2-dimensional images at fast acquisition rates. However, 2-dimensional images are desirable for clinical use because surgeons become familiar with the diagnosis using 2-dimensional images. Nerve visualization tools to provide 2-dimensional images are required.

Coherent Raman scattering (CRS) including coherent anti-Stokes Raman scattering (CARS) and stimulated Raman scattering (SRS) is more suitable for observing nerves while getting 2-dimension images. The CRS has the 10^5-10^7 times higher efficiency than spontaneous Raman scattering [25–27]. The CRS has the potential to provide 2-dimensional nerve images in a label-free manner at a real-time imaging rate. Several CRS endoscopes have been reported [28–36]. For now, the imaging rates are still slow in observing biological samples and far from practical use. Therefore,

the imaging rate is needed to accelerate for use in medical settings.

The purpose of this study is to overcome the drawback by combining image processing with deep learning. The development of deep learning accelerated rapidly since AlexNet won a competition of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [37]. This trend is called the third artificial intelligence (AI) boom [38]. Tremendous deep-learning models are developed for several fields of image processing (classification, object detection, semantic segmentation, and denoising). However, the quantitative evaluation of imaging rate acceleration has not been reported. Most of the studies are about comparisons between models. The author proposed a new metric, critical imaging rate (CIR), to evaluate an imaging rate quantitatively. CIR is defined as the highest imaging rate which satisfies the image quality needed for medical images. As image processing with deep learning, the author has focused on denoising and semantic segmentation. In training for semantic segmentation, a learning method to utilize fluorescence images as pre-training is proposed to compensate for the amount of dataset. In this study, CARS endoscopy with deep-learning image processing is demonstrated in terms of accelerating the imaging rate.

The outline of the thesis is as follows:

In chapter 1, the author reviews coherent Raman scattering (CRS) applications for medical imaging. CRS is used to visualize molecular mapping in a label-free manner based on molecular vibration. It is expected for medical applications because label-free imaging makes it highly compatible with biological applications. Two promising areas for medical applications are summarized. The first area is a cancer diagnosis. This benefits from the time savings due to a label-free manner. The label-free manner eliminates the time required for the staining process. CRS also can obtain images equivalent to conventional hematoxylin and eosin (H&E) staining images which are usually used for definitive diagnosis. The second area is endoscopic applications. The miniaturized CRS endoscope enables *in situ* diagnosis during surgery.

In chapter 2, the imaging acceleration with deep learning-based denoising is shown. The denoising process can suppress denoise in images, and the signal-to-

noise ratio in images increases. The increase in the signal-to-noise ratio allows for shorter exposure times when acquiring an image with a certain signal-to-noise ratio. To evaluate the degree of imaging acceleration quantitatively, the author proposed a new evaluation metric, critical imaging rate (CIR). CIR is defined as an imaging rate satisfying an image quality needed for medical imaging.

In chapter 3, automated nerve extraction from CARS images is demonstrated. CARS images using symmetric CH_2 stretching vibration visualize not only nerve but also non-nervous tissues. For nerve-sparing surgery, the distribution of only nerves, not lipid-rich tissues, is required. To display nerve distribution, nerve extraction from CARS images is demonstrated. The author proposed a transfer learning method using fluorescence images. Lipid-stained fluorescence images were prepared to obtain images equivalent to CARS images. The usefulness of the transfer learning method using fluorescence was evaluated.

In chapter 4, the imaging acceleration with deep learning-based nerve extraction is demonstrated. The previous chapter evaluates the usefulness of the transfer learning method for CARS images with a high SN ratio. Nerve extraction from CARS images with a low SN ratio means the imaging acceleration for understanding the nerve distribution. For evaluation of imaging acceleration, *CIR* based on F_1 score, which evaluates the similarity between segmented images and ground truth images, is used.

Reference

[1] C. A. Blum and D. B. Adams, "Who did the first laparoscopic cholecystectomy?," *Journal of minimal access surgery*, vol. 7, pp. 165–168, July 2011.

[2] S. Kitano and N. Shiraishi, "Current status of laparoscopic gastrectomy for cancer in Japan," *Surgical endoscopy*, vol. 18, pp. 182–185, Feb. 2004.

[3] B. Rocco, D.-V. Matei, S. Melegari, J. C. Ospina, F. Mazzoleni, G. Errico, M. Mastropasqua, L. Santoro, S. Detti, and O. de Cobelli, "Robotic vs open prostatectomy in a laparoscopically naive centre: a matched-pair analysis," *BJU international*, vol. 104, no. 7, pp. 991–995, 2009.

[4] J. E. Terrell, D. E. Welsh, C. R. Bradford, D. B. Chepeha, R. M. Esclamado, N. D. Hogikyan, and G. T. Wolf, "Pain, quality of life, and spinal accessory nerve status after neck dissection," *The Laryngoscope*, vol. 110, pp. 620–626, Apr. 2000.

[5] L. Erisen, B. Basel, J. Irdesel, M. Zarifoglu, H. Coskun, O. Basut, I. Tezel, I. Hizalan, and S. Onart, "Shoulder function after accessory nerve-sparing neck dissections," *Head & neck*, vol. 26, pp. 967–971, Nov. 2004.

[6] A. Pietrangeli, P. Pugliese, M. Perrone, I. Sperduti, M. Cosimelli, and B. Jandolo, "Sexual dysfunction following surgery for rectal cancer - a clinical and neurophysiological study," *Journal of experimental & clinical cancer research: CR*, vol. 28, p. 128, Sept. 2009.

[7] A. N. Anaizi, E. A. Gantwerker, M. L. Pensak, and P. V. Theodosopoulos, "Facial nerve preservation surgery for koos grade 3 and 4 vestibular schwannomas," *Neurosurgery*, vol. 75, pp. 671–5; discussion 676–7; quiz 677, Dec. 2014.

[8] S. D. Kundu, K. A. Roehl, S. E. Eggener, J. A. V. Antenor, M. Han, and W. J. Catalona, "Potency, continence and complications in 3,477 consecutive radical retropubic prostatectomies," *The Journal of urology*, vol. 172, pp. 2227–2231, Dec. 2004.

[9] O.-S. Barnoiu, E. Garcia Galisteo, F. Baron Lopez, R. Vozmediano Chicharro, J. Soler Martinez, J. M. Del Rosal Samaniego, J. Machuca Santacruz, and V. Baena Gonzalez, "Prospective urodynamic model for prediction of urinary incontinence after robot-assisted radical prostatectomy," *Urologia internationalis*, vol. 92, no. 3, pp. 306–309, 2014.

[10] E. Haglind, S. Carlsson, J. Stranne, A. Wallerstedt, U. Wilderäng, T. Thorsteinsdottir, M. Lagerkvist, J.-E. Damber, A. Bjartell, J. Hugosson, P. Wiklund, G. Steineck, and LAPPRO steering committee, "Urinary Incontinence and Erectile Dysfunction After Robotic Versus Open Radical Prostatectomy: A Prospective, Controlled, Nonrandomised Trial," *European urology*, vol. 68, pp. 216–225, Aug. 2015.

[11] J. Rehman, G. J. Christ, A. Kaynan, D. Samadi, and J. Fleischmann, "Intraoperative electrical stimulation of cavernosal nerves with monitoring of intracorporeal pressure in patients undergoing nerve sparing radical prostatectomy," *BJU international*, vol. 84, pp. 305–310, Aug. 1999.

[12] C. P. Nelson, J. E. Montie, E. J. McGuire, G. Wedemeyer, and J. T. Wei, "Intraoperative nerve stimulation with measurement of urethral sphincter pressure changes during radical retropubic prostatectomy: a feasibility study," *The Journal of urology*, vol. 169, pp. 2225–2228, June 2003.

[13] A. Katahira, H. Niikura, Y. Kaiho, H. Nakagawa, K. Kurokawa, Y. Arai, and N. Yaegashi, "Intraoperative electrical stimulation of the pelvic splanchnic nerves during nerve-sparing radical hysterectomy," *Gynecologic oncology*, vol. 98, pp. 462–466, Sept. 2005.

[14] M. Hermann, C. Hellebart, and M. Freissmuth, "Neuromonitoring in thyroid surgery: prospective evaluation of intraoperative electrophysiological responses for the prediction of recurrent laryngeal nerve injury," *Annals of surgery*, vol. 240, pp. 9–17, July 2004.

[15] S. L. Gibbs-Strauss, K. A. Nasr, K. M. Fish, O. Khullar, Y. Ashitate, T. M. Siclovan, B. F. Johnson, N. E. Barnhardt, C. A. Tan Hehir, and J. V. Frangioni, "Nerve-highlighting fluorescent contrast agents for image-guided surgery," *Molecular imaging*, vol. 10, pp. 91–101, Apr. 2011.

[16] C. Wang, C. Wu, J. Zhu, R. H. Miller, and Y. Wang, "Design, synthesis, and evaluation of coumarin-based molecular probes for imaging of myelination," *Journal of medicinal chemistry*, vol. 54, pp. 2331–2340, Apr. 2011.

[17] M. A. Whitney, J. L. Crisp, L. T. Nguyen, B. Friedman, L. A. Gross, P. Steinbach, R. Y. Tsien, and Q. T. Nguyen, "Fluorescent peptides highlight peripheral nerves during surgery in mice," *Nature biotechnology*, vol. 29, pp. 352–356, Apr. 2011.

[18] V. E. Cotero, T. Siclovan, R. Zhang, R. L. Carter, A. Bajaj, N. E. LaPlante, E. Kim, D. Gray, V. P. Staudinger, S. Yazdanfar, and C. A. Tan Hehir, "Intraoperative fluorescence imaging of peripheral and central nerves through a myelin-selective contrast agent," *Molecular imaging and biology: MIB: the official publication of the Academy of Molecular Imaging*, vol. 14, pp. 708–717, Dec. 2012.

[19] C. A. Massaad, G. Zhang, L. Pillai, A. Azhdarinia, W. Liu, and K. A. Sheikh, "Fluorescently-tagged anti-ganglioside antibody selectively identifies peripheral nerve in living animals," *Scientific reports*, vol. 5, p. 15766, Oct. 2015.

[20] J. Cha, A. Broch, S. Mudge, K. Kim, J.-M. Namgoong, E. Oh, and P. Kim, "Real-time, label-free, intraoperative visualization of peripheral nerves and micro-vasculatures using multimodal optical imaging techniques," *Biomedical optics express*, vol. 9, pp. 1097–1110, Mar. 2018.

[21] E. M. Walsh, D. Cole, K. E. Tipirneni, K. I. Bland, N. Udayakumar, B. B. Kasten, S. L. Bevans, B. M. McGrew, J. J. Kain, Q. T. Nguyen, E. L. Rosenthal, and J. M. Warram, "Fluorescence Imaging of Nerves During Surgery," *Annals of surgery*, vol. 270, pp. 69–76, July 2019.

[22] T. Minamikawa, Y. Harada, N. Koizumi, K. Okihara, K. Kamoi, A. Yanagisawa, and T. Takamatsu, "Label-free detection of peripheral nerve tissues against adjacent tissues by spontaneous Raman microspectroscopy," *Histochemistry and cell biology*, vol. 139, pp. 181–193, Jan. 2013.

[23] T. Minamikawa, Y. Harada, and T. Takamatsu, "Ex vivo peripheral nerve detection of rats by spontaneous Raman spectroscopy," *Scientific reports*, vol. 5, p. 17165, 2015.

[24] Y. Kumamoto, Y. Harada, H. Tanaka, and T. Takamatsu, "Rapid and accurate peripheral nerve imaging by multipoint Raman spectroscopy," *Scientific reports*, vol. 7, p. 845, Apr. 2017.

[25] D. Pestov, G. O. Ariunbold, X. Wang, R. K. Murawski, V. A. Sautenkov, A. V. Sokolov, and M. O. Scully, "Coherent versus incoherent Raman scattering: molecular coherence excitation and measurement," *Optics Letters*, vol. 32, pp. 1725–1727, June 2007.

[26] G. I. Petrov, R. Arora, V. V. Yakovlev, X. Wang, A. V. Sokolov, and M. O. Scully, "Comparison of coherent and spontaneous Raman microspectroscopies for noninvasive detection of single bacterial endospores," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, pp. 7776–7779, May 2007.

[27] H. Rigneault and P. Berto, "Tutorial: Coherent Raman light matter interaction processes," *APL Photonics*, vol. 3, no. 9, p. 091101, 2018.

[28] F. Légaré, C. L. Evans, F. Ganikhanov, and X. S. Xie, "Towards CARS Endoscopy," *Optics express*, vol. 14, pp. 4427–4432, May 2006.

[29] M. Balu, G. Liu, Z. Chen, B. J. Tromberg, and E. O. Potma, "Fiber delivered probe for efficient CARS imaging of tissues," *Optics express*, vol. 18, pp. 2380–2388, Feb. 2010.

[30] B. G. Saar, R. S. Johnston, C. W. Freudiger, X. S. Xie, and E. J. Seibel, "Coherent Raman scanning fiber endoscopy," *Optics letters*, vol. 36, pp. 2396–2398, July 2011.

[31] A. Lukic, S. Dochow, H. Bae, G. Matz, I. Latka, B. Messerschmidt, M. Schmitt, and J. Popp, "Endoscopic fiber probe for nonlinear spectroscopic imaging," *Optica*, vol. 4, no. 5, pp. 496–501, 2017.

[32] A. Lombardini, V. Mytskaniuk, S. Sivankutty, E. R. Andresen, X. Chen, J. Wenger, M. Fabert, N. Joly, F. Louradour, A. Kudlinski, and H. Rigneault, "High-resolution multimodal flexible coherent Raman endoscope," *Light, science & applications*, vol. 7, p. 10, May 2018.

[33] K. Hirose, S. Fukushima, T. Furukawa, H. Niioka, and M. Hashimoto, "Invited Article: Label-free nerve imaging with a coherent anti-Stokes Raman scattering rigid endoscope using two optical fibers for laser delivery," *APL Photonics*, vol. 3, p. 092407, Sept. 2018.

[34] P. Zirak, G. Matz, B. Messerschmidt, T. Meyer, M. Schmitt, J. Popp, O. Uckermann, R. Galli, M. Kirsch, M. J. Winterhalder, and A. Zumbusch, "A rigid coherent anti-Stokes Raman scattering endoscope with high resolution and a large field of view," *APL Photonics*, vol. 3, no. 9, p. 092409, 2018.

[35] E. Pshenay-Severin, H. Bae, K. Reichwald, G. Matz, J. Bierlich, J. Kobelke, A. Lorenz, A. Schwuchow, T. Meyer-Zedler, M. Schmitt, B. Messerschmidt, and J. Popp, "Multimodal nonlinear endomicroscopic imaging probe using a double-core double-clad fiber and focus-combining micro-optical concept," *Light, science & applications*, vol. 10, p. 207, Oct. 2021.

[36] D. Septier, V. Mytskaniuk, R. Habert, D. Labat, K. Baudelle, A. Cassez, G. Brévalle-Wasilewski, M. Conforti, G. Bouwmans, H. Rigneault, and A. Kudlinski, "Label-free highly multimodal nonlinear endoscope," *Optics Express*, vol. 30, no. 14, pp. 25020–25033, 2022.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, May 2017.

[38] H. Fujita, "AI-based computer-aided diagnosis (AI-CAD): the latest review to read first," *Radiological physics and technology*, vol. 13, pp. 6–19, Mar. 2020.

Chapter 1

Application of coherent Raman scattering toward medical instruments

1.1 Coherent Raman scattering

Coherent Raman scattering (CRS), including coherent anti-Stokes Raman scattering (CARS) and stimulated Raman scattering (SRS), has the potential to identify the material in biological samples in a label-free manner. The "Raman effect" was discovered in 1928 [1]. After that, the "SRS process" and "CARS process" were observed in 1962–1965 [2, 3]. The energy diagrams are shown in Fig. 1.1. CRS occurs when the difference ($\omega_p - \omega_S$) in angular frequency between two excitation lights (ω_p and ω_p) matches the molecular vibration (Ω) in samples. Molecular vibration is sensitive to the composition and the chemical bonds in molecules. The majority of the molecular vibration information is in the fingerprint spectral region (~500 cm^{-1} to 1,800 cm^{-1}) [4]. The other information exists in the CH-/OH-stretch region (~2,700 cm^{-1} to 3,300 cm^{-1}). In coherent anti-Stokes Raman scattering (CARS), the nonresonant signal which has the same wavelength as the resonant signal is emitted (Fig. 1.1b). In stimulated Raman scattering (SRS), the stimulated Raman loss (SRL) for the pump

excitation light and the stimulated Raman gain (SRG) for the Stokes excitation light are generated. SRL and SRG have the 10^{-6} order intensity of the excitation light. In general, the intensity change in this order is detected by the lock-in amplifier.

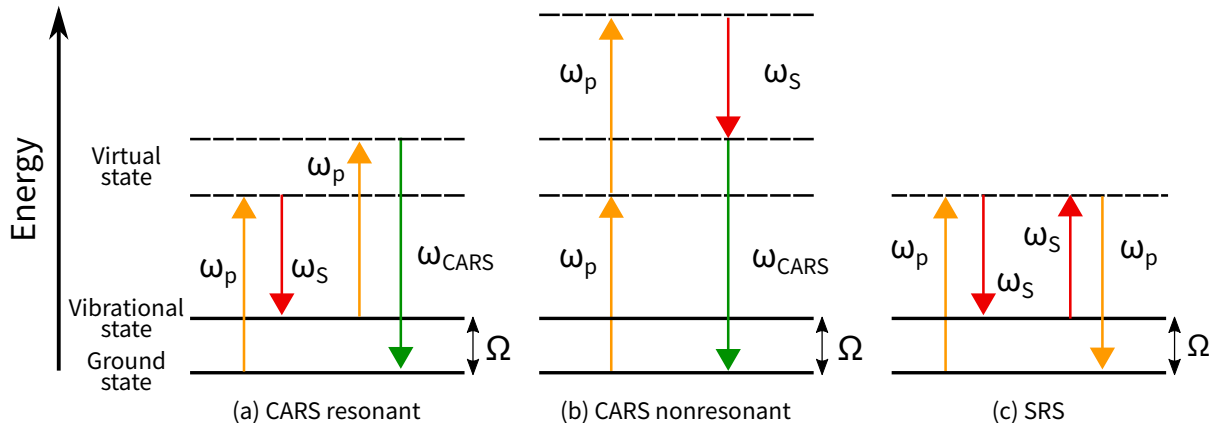


Figure 1.1: Energy diagrams of coherent Raman scattering. ω_p and ω_S are the angular frequency of the pump and Stokes excitation lights, respectively. The molecular vibration is shown as Ω. (a) the energy diagram of coherent anti-Stokes Raman scattering (CARS). The angular frequency of resonant CARS has $2\omega_p - \omega_S$. (b) the energy diagram of four-wave mixing with the same wavelength of resonant CARS signals. (c) the energy diagram of stimulated Raman scattering (SRS).

1.1.1 Nonlinear process

Dipole moment per unit volume ($\boldsymbol{P}$) induced by incident light ($\boldsymbol{E}$) are given by [5]

$$\boldsymbol{P} = \epsilon_0[\chi^{(1)}\boldsymbol{E} + \chi^{(2)}\boldsymbol{EE} + \chi^{(3)}\boldsymbol{EEE} + \cdots], \tag{1.1}$$

where ϵ_0 is the permittivity in free space and $\chi^{(1)}$ is the linear suspceptibility. The $\chi^{(2)}$ and $\chi^{(3)}$ indicate the second- and third-order nonlinear optical susceptibilities, respectively. Coherent Raman scattering is induced as the third-order nonlinear optics. The third-order nonlinear polarization is represented as

$$\boldsymbol{P}^{(3)} = \epsilon_0\chi^{(3)}\boldsymbol{EEE}. \tag{1.2}$$

1.1.2 Coherent anti-Stokes Raman scattering process

CARS is one of the third-order nonlinear processes. Two laser beams with angular frequencies ($\omega_p > \omega_S$) excite a molecular vibration with Ω meeting the relationship:

$$\Omega = \omega_p - \omega_S . \tag{1.3}$$

Light with different angular frequencies from excitation laser beams is generated from anti-Stokes shifted radiation. The frequency is written by

$$\omega_{CARS} = 2\omega_p - \omega_S = \omega_p + \Omega. \tag{1.4}$$

The wave vector of CARS polarization must satisfy the phase-matching condition written as

$$\boldsymbol{k}_{CARS} = 2\boldsymbol{k}_p - \boldsymbol{k}_S . \tag{1.5}$$

In CARS imaging, a high NA objective lens could relax the strict phase-matching condition [6].

1.1.3 Stimulated Raman scattering process

The SRS process needs two laser beams with different angular frequencies ($\omega_p > \omega_S$) to excite molecular vibration (Ω). An energy transfer from the high-frequency laser (ω_p) to the low-frequency laser (ω_S) is induced by the SRS process. The energy loss of the high-frequency laser and the energy gain of the low-frequency laser are called stimulated Raman loss (SRL) and stimulated Raman gain (SRG), respectively. The frequency of the SRL signal is written as

$$\omega_{SRL} = \omega_p + \omega_S - \omega_S = \omega_p. \tag{1.6}$$

The nonlinear polarization induced by the SRL process is provided with

$$\boldsymbol{P}^{(3)}_{SRL}(t, \boldsymbol{r}) = \chi^{(3)} E_p E_S E_S \exp(-i(\omega_p t - 2\pi \boldsymbol{k}_p \cdot \boldsymbol{r})). \tag{1.7}$$

It is found that the phase-matching condition is easily satisfied in the SRS process from Eq. 1.7.

CRS allows for fast real-time imaging of biological samples. CRS shows 10^5-10^7 times higher scattering efficiency than spontaneous Raman scattering [7–9]. Microscopy applications of CARS and SRS were reported in 1982–2000 [10–12] and in 2007–2009 [13–15], respectively. CRS microscopy shows the ability to observe biological samples at video-rate [16–18]. Since then, several applications for biological samples have been reported.

1.2 Application toward medical instruments

1.2.1 Cancer diagnosis

CARS microscopy can differentiate breast cancer and lung cancer from other tissues [19–21]. CARS images at the wavenumber 2845 cm^{-1} (symmetric CH_2 stretching band) are used for the classification analysis. Cell nuclei appear to be dark holes because rich protein exists in nuclei. In particular, the shape and density of the nucleus are key diagnostic features. The feature of fibrils and cell morphology (inhomogeneous cytoplasm and cell nuclei) is obtained from CARS images and is used for differentiating the subtypes of cancer.

SRS microscopy is introduced as a diagnostic tool alternative to conventional hematoxylin and eosin (H&E) staining procedures in brain cancer surgery [22]. It is called stimulated Raman histology (SRH). In SRH, the virtual-colored H&E images having two color channels are generated from two Raman shifts (2845 cm^{-1} and 2930 cm^{-1}). 2845 cm^{-1} corresponds to symmetric CH_2 stretching vibration which is rich in lipids. 2930 cm^{-1} corresponds to CH_3 stretching vibration which is mainly included

in proteins and DNA. The one channel with reddish-pink color is 2845 cm^{-1} channel, and the other channel with dark-blue/violet color is 2930 cm^{-1} minus 2845 cm^{-1}. These colors are determined to mimic conventional H&E images. The virtual-colored H&E images generated from two Raman shifts enable the detection of key diagnostic histologic features. In addition to brain cancer [23–26], it is reported that SRH is useful in other types of cancer [27, 28].

1.2.2 Endoscopy

Various CRS endoscopes have been developed for biomedical applications [29–37]. The specifications of CRS endoscopes are summarized in Table. 1.1. They are listed in the order of publication in Table. 1.1. The CARS endoscopes are plotted in Fig. 1.3. The horizontal and vertical axes in Fig. 1.3 show the frame rate (frame per second; fps) and the field of view (FOV), respectively. The graph shows the trade-off relationship that the FOV decreases as the frame rate and NA increase. The author reviews these articles in terms of FOV, fps and scanning method. Three types of scanners shown in Fig. 1.2 are adopted in miniatured CRS endoscopes (stage scanning, galvanometer scanners in free-space or with bundle fiber, and spiral scanning).

Table 1.1: Summary of coherent Raman scattering endoscopy. fps; frame per seconds, FOV; field of view.

citation	fps	FOV	scanning method	sample
[29] in 2006	0.061	29 μm×29 μm	stage	polystyrene beads
[30] in 2010	0.50	300 μm×300 μm	galvano	mouse ear skin
[31] in 2011	7.0	80 μm circle	spiral	mouse skin
[32] in 2017	0.10	500 μm×500 μm	galvano (bundle fiber)	human skin
[33] in 2018	0.15	320 μm circle	spiral	human colon
[34] in 2018	0.050	450 μm×450 μm	galvano	rat sciatic nerve
[35] in 2018	0.10	250 μm circle	galvano	murine spinal cord
[36] in 2021	0.020	180 μm circle	spiral	human epithelial tissue
[37] in 2022	0.40	200 μm circle	spiral	ox liver

The first CRS endoscope reported in 2006 [29] adopts stage scanning (Fig. 1.2a). The two excitation lights are delivered through the non-polarization-maintaining

single-mode silica fiber. The 4 mm diameter endoscope head without a scanner is fixed and the sample was raster scanned by a piezo stage. The frame rate and FOV are 0.061 and 29 μm×29 μm, respectively. The frame rate is slow due to the use of stage scanning. The CARS imaging is demonstrated using 0.75 μm polystyrene beads.

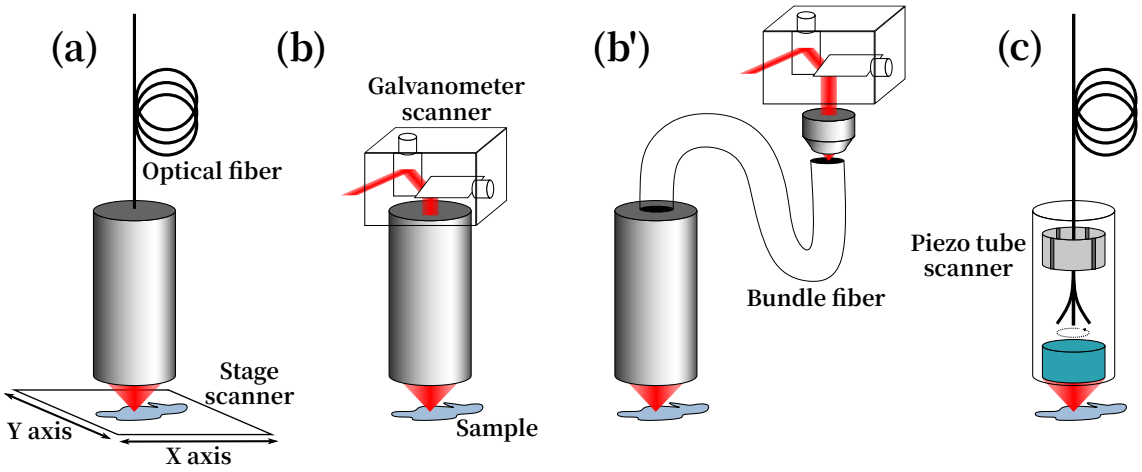


Figure 1.2: Scanner methods of CRS endoscopes. (a) stage scanning. Optical fiber delivers the excitation light and CRS light. (b) and (b') galvanometer scanning (GS). In b, GS is equipped with endoscopes. In b', the excitation light and CRS light are transported with bundle fiber. (c) spiral scanning. The fiber end is resonantly vibrated by a piezo tube with 4 electrodes.

Galvanometer scanners (Fig. 1.2b and b') have been adopted for several CRS endoscopes [30,32,34,35]. Galvanometer scanners are famous for acquiring 2-dimensional images in free-space optics. Each of the two mirrors rotates to acquire 2-dimensional images. The first CRS endoscope with galvanometer scanners was reported in 2010 [30]. The diameter and length are 3 cm and 25 cm, respectively. The size gets relatively large due to the use of the normal objective lens. The CRS endoscope using galvanometer scanners with bundle fiber (Fig. 1.2b') is reported in 2017 [32]. The endoscope head can be miniaturized using bundle fiber because the scanner part is not included in the endoscope head. After that, the CRS endoscopes using galvanometer scanners are reported [34, 35].

Spiral scanning (Fig. 1.2c) also has been widely used for miniatured CRS endo-

scopes [31, 33, 36, 37]. In spiral scanning, the fiber end is resonantly vibrated by an electrical piezo tube with 4 electrodes. The resonant frequency of the fiber cantilever is expressed by

$$f = \frac{\beta d}{4\pi l^2}\sqrt{\frac{E}{\rho}}, \tag{1.8}$$

where d is the radius of the cantilever, l is the length of the cantilever, E is the Young's modules, ρ is the mass density, and β is a resonance coefficient dependent on the vibration mode number and boundary conditions [38]. $\beta \approx 3.52$ is used for the zeroth-order vibration mode. It is widely used not only in CRS endoscopes but also in other optical imaging applications (optical coherence tomography, fluorescence imaging, second harmonic generation) due to the benefits of compact head size and fast imaging rate [38–47].

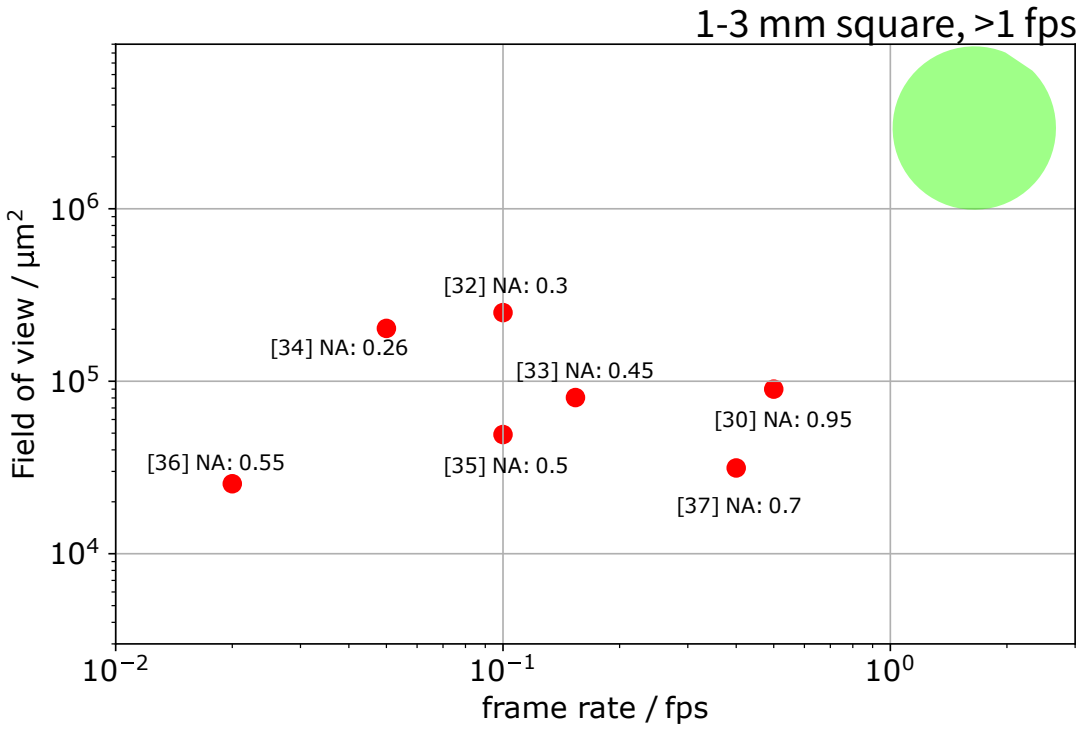


Figure 1.3: Benchmark of coherent anti-Stokes Raman scattering endoscopy. The NA is shown on the right side of the citation number. The graph indicates the trade-off relationship between FOV and fps. As the NA increases, the fps increase because CARS is induced more efficiently using the higher objective lens. However, the FOV is limited using the higher NA objective lens.

Reference

[1] C. V. Raman and K. S. Krishnan, "A New Type of Secondary Radiation," *Nature*, vol. 121, pp. 501–502, Mar. 1928.

[2] G. Eckhardt, R. W. Hellwarth, F. J. McClung, S. E. Schwarz, D. Weiner, and E. J. Woodbury, "Stimulated Raman Scattering From Organic Liquids," *Physical review letters*, vol. 9, pp. 455–457, Dec. 1962.

[3] P. D. Maker and R. W. Terhune, "Study of Optical Effects Due to an Induced Polarization Third Order in the Electric Field Strength," *Physics Review*, vol. 137, pp. A801–A818, Feb. 1965.

[4] C. H. Camp, Jr and M. T. Cicerone, "Chemically sensitive bioimaging with coherent Raman scattering," *Nature photonics*, vol. 9, pp. 295–305, Apr. 2015.

[5] R. W. Boyd, *Nonlinear Optics*. Academic Press, Mar. 2020.

[6] M. Hashimoto and T. Araki, "Three-dimensional transfer functions of coherent anti-Stokes Raman scattering microscopy," *Journal of the Optical Society of America. A, Optics, image science, and vision*, vol. 18, pp. 771–776, Apr. 2001.

[7] D. Pestov, G. O. Ariunbold, X. Wang, R. K. Murawski, V. A. Sautenkov, A. V. Sokolov, and M. O. Scully, "Coherent versus incoherent Raman scattering: molecular coherence excitation and measurement," *Optics Letters*, vol. 32, pp. 1725–1727, June 2007.

[8] G. I. Petrov, R. Arora, V. V. Yakovlev, X. Wang, A. V. Sokolov, and M. O. Scully, "Comparison of coherent and spontaneous Raman microspectroscopies for noninvasive detection of single bacterial endospores," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, pp. 7776–7779, May 2007.

[9] H. Rigneault and P. Berto, "Tutorial: Coherent Raman light matter interaction processes," *APL Photonics*, vol. 3, no. 9, p. 091101, 2018.

[10] M. D. Duncan, J. Reintjes, and T. J. Manuccia, "Scanning coherent anti-Stokes Raman microscope," *Optics letters*, vol. 7, pp. 350–352, Aug. 1982.

[11] A. Zumbusch, G. R. Holtom, and X. Sunney Xie, "Three-Dimensional Vibrational Imaging by Coherent Anti-Stokes Raman Scattering," *Physical review letters*, vol. 82, no. 20, pp. 4142–4145, 1999.

[12] M. Hashimoto, T. Araki, and S. Kawata, "Molecular vibration imaging in the fingerprint region by use of coherent anti-Stokes Raman scattering microscopy with a collinear configuration," *Optics letters*, vol. 25, pp. 1768–1770, Dec. 2000.

[13] E. Ploetz, S. Laimgruber, S. Berner, W. Zinth, and P. Gilch, "Femtosecond stimulated Raman microscopy," *Applied physics. B, Lasers and optics*, vol. 87, pp. 389–393, May 2007.

[14] C. W. Freudiger, W. Min, B. G. Saar, S. Lu, G. R. Holtom, C. He, J. C. Tsai, J. X. Kang, and X. S. Xie, "Label-free biomedical imaging with high sensitivity by stimulated Raman scattering microscopy," *Science*, vol. 322, pp. 1857–1861, Dec. 2008.

[15] P. Nandakumar, A. Kovalev, and A. Volkmer, "Vibrational imaging based on stimulated Raman scattering microscopy," *New journal of physics*, vol. 11, p. 033026, Mar. 2009.

[16] C. L. Evans, E. O. Potma, M. Puoris'haag, D. Côté, C. P. Lin, and X. S. Xie, "Chemical imaging of tissue in vivo with video-rate coherent anti-Stokes Raman scattering microscopy," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, pp. 16807–16812, Nov. 2005.

[17] B. G. Saar, C. W. Freudiger, J. Reichman, C. M. Stanley, G. R. Holtom, and X. S. Xie, "Video-Rate Molecular Imaging in Vivo with Stimulated Raman Scattering," *Science*, vol. 330, no. 6009, pp. 1368–1370, 2010.

[18] Y. Ozeki, W. Umemura, Y. Otsuka, S. Satoh, H. Hashimoto, K. Sumimura, N. Nishizawa, K. Fukui, and K. Itoh, "High-speed molecular spectral imaging of tissue with stimulated Raman scattering," *Nature photonics*, vol. 6, pp. 845–851, Nov. 2012.

[19] Y. Yang, F. Li, L. Gao, Z. Wang, M. J. Thrall, S. S. Shen, K. K. Wong, and S. T. C. Wong, "Differential diagnosis of breast cancer using quantitative, label-free and molecular vibrational imaging," *Biomedical optics express*, vol. 2, pp. 2160–2174, Aug. 2011.

[20] L. Gao, Z. Wang, F. Li, A. A. Hammoudi, M. J. Thrall, P. T. Cagle, and S. T. C. Wong, "Differential diagnosis of lung carcinoma with coherent anti-Stokes Raman scattering imaging," *Archives of pathology & laboratory medicine*, vol. 136, pp. 1502–1510, Dec. 2012.

[21] S. Weng, X. Xu, J. Li, and S. T. C. Wong, "Combining deep learning and coherent anti-Stokes Raman scattering imaging for automated differential diagnosis of lung cancer," *Journal of biomedical optics*, vol. 22, no. 10, p. 106017, 2017.

[22] D. A. Orringer, B. Pandian, Y. S. Niknafs, T. C. Hollon, J. Boyle, S. Lewis, M. Garrard, S. L. Hervey-Jumper, H. J. L. Garton, C. O. Maher, J. A. Heth, O. Sagher, D. A. Wilkinson, M. Snuderl, S. Venneti, S. H. Ramkissoon, K. A. McFadden, A. Fisher-Hubbard, A. P. Lieberman, T. D. Johnson, X. S. Xie,

J. K. Trautman, C. W. Freudiger, and S. Camelo-Piragua, "Rapid intraoperative histology of unprocessed surgical specimens via fibre-laser-based stimulated Raman scattering microscopy," *Nature biomedical engineering*, vol. 1, p. 0027, Feb. 2017.

[23] T. C. Hollon, S. Lewis, B. Pandian, Y. S. Niknafs, M. R. Garrard, H. Garton, C. O. Maher, K. McFadden, M. Snuderl, A. P. Lieberman, K. Muraszko, S. Camelo-Piragua, and D. A. Orringer, "Rapid Intraoperative Diagnosis of Pediatric Brain Tumors Using Stimulated Raman Histology," *Cancer research*, vol. 78, pp. 278–289, Jan. 2018.

[24] T. C. Hollon, B. Pandian, A. R. Adapa, E. Urias, A. V. Save, S. S. S. Khalsa, D. G. Eichberg, R. S. D'Amico, Z. U. Farooq, S. Lewis, P. D. Petridis, T. Marie, A. H. Shah, H. J. L. Garton, C. O. Maher, J. A. Heth, E. L. McKean, S. E. Sullivan, S. L. Hervey-Jumper, P. G. Patil, B. G. Thompson, O. Sagher, G. M. McKhann, 2nd, R. J. Komotar, M. E. Ivan, M. Snuderl, M. L. Otten, T. D. Johnson, M. B. Sisti, J. N. Bruce, K. M. Muraszko, J. Trautman, C. W. Freudiger, P. Canoll, H. Lee, S. Camelo-Piragua, and D. A. Orringer, "Near real-time intraoperative brain tumor diagnosis using stimulated Raman histology and deep neural networks," *Nature medicine*, vol. 26, pp. 52–58, Jan. 2020.

[25] L. Di, D. G. Eichberg, K. Huang, A. H. Shah, A. M. Jamshidi, E. M. Luther, V. M. Lu, R. J. Komotar, M. E. Ivan, and S. H. Gultekin, "Stimulated Raman Histology for Rapid Intraoperative Diagnosis of Gliomas," *World neurosurgery*, vol. 150, pp. e135–e143, June 2021.

[26] L. Di, D. G. Eichberg, Y. J. Park, A. H. Shah, A. M. Jamshidi, E. M. Luther, V. M. Lu, R. J. Komotar, M. E. Ivan, and S. H. Gultekin, "Rapid Intraoperative Diagnosis of Meningiomas using Stimulated Raman Histology," *World neurosurgery*, vol. 150, pp. e108–e116, June 2021.

[27] B. Sarri, F. Poizat, S. Heuke, J. Wojak, F. Franchi, F. Caillol, M. Giovannini, and H. Rigneault, "Stimulated Raman histology: one to one comparison with standard hematoxylin and eosin staining," *Biomedical Optics Express*, vol. 10, pp. 5378–5384, Oct. 2019.

[28] Z. Liu, W. Su, J. Ao, M. Wang, Q. Jiang, J. He, H. Gao, S. Lei, J. Nie, X. Yan, X. Guo, P. Zhou, H. Hu, and M. Ji, "Instant diagnosis of gastroscopic biopsy via deep-learned single-shot femtosecond stimulated Raman histology," *Nature communications*, vol. 13, p. 4050, July 2022.

[29] F. Légaré, C. L. Evans, F. Ganikhanov, and X. S. Xie, "Towards CARS Endoscopy," *Optics express*, vol. 14, pp. 4427–4432, May 2006.

[30] M. Balu, G. Liu, Z. Chen, B. J. Tromberg, and E. O. Potma, "Fiber delivered probe for efficient CARS imaging of tissues," *Optics express*, vol. 18, pp. 2380–2388, Feb. 2010.

[31] B. G. Saar, R. S. Johnston, C. W. Freudiger, X. S. Xie, and E. J. Seibel, "Coherent Raman scanning fiber endoscopy," *Optics letters*, vol. 36, pp. 2396–2398, July 2011.

[32] A. Lukic, S. Dochow, H. Bae, G. Matz, I. Latka, B. Messerschmidt, M. Schmitt, and J. Popp, "Endoscopic fiber probe for nonlinear spectroscopic imaging," *Optica*, vol. 4, no. 5, pp. 496–501, 2017.

[33] A. Lombardini, V. Mytskaniuk, S. Sivankutty, E. R. Andresen, X. Chen, J. Wenger, M. Fabert, N. Joly, F. Louradour, A. Kudlinski, and H. Rigneault, "High-resolution multimodal flexible coherent Raman endoscope," *Light, science & applications*, vol. 7, p. 10, May 2018.

[34] K. Hirose, S. Fukushima, T. Furukawa, H. Niioka, and M. Hashimoto, "Invited Article: Label-free nerve imaging with a coherent anti-Stokes Raman scattering

rigid endoscope using two optical fibers for laser delivery," *APL Photonics*, vol. 3, p. 092407, Sept. 2018.

[35] P. Zirak, G. Matz, B. Messerschmidt, T. Meyer, M. Schmitt, J. Popp, O. Uckermann, R. Galli, M. Kirsch, M. J. Winterhalder, and A. Zumbusch, "A rigid coherent anti-Stokes Raman scattering endoscope with high resolution and a large field of view," *APL Photonics*, vol. 3, no. 9, p. 092409, 2018.

[36] E. Pshenay-Severin, H. Bae, K. Reichwald, G. Matz, J. Bierlich, J. Kobelke, A. Lorenz, A. Schwuchow, T. Meyer-Zedler, M. Schmitt, B. Messerschmidt, and J. Popp, "Multimodal nonlinear endomicroscopic imaging probe using a double-core double-clad fiber and focus-combining micro-optical concept," *Light, science & applications*, vol. 10, p. 207, Oct. 2021.

[37] D. Septier, V. Mytskaniuk, R. Habert, D. Labat, K. Baudelle, A. Cassez, G. Brévalle-Wasilewski, M. Conforti, G. Bouwmans, H. Rigneault, and A. Kudlinski, "Label-free highly multimodal nonlinear endoscope," *Optics Express*, vol. 30, no. 14, pp. 25020–25033, 2022.

[38] X. Liu, M. J. Cobb, Y. Chen, M. B. Kimmey, and X. Li, "Rapid-scanning forward-imaging miniature endoscope for real-time optical coherence tomography," *Optics letters*, vol. 29, pp. 1763–1765, Aug. 2004.

[39] M. T. Myaing, D. J. MacDonald, and X. Li, "Fiber-optic scanning two-photon fluorescence endoscope," *Optics letters*, vol. 31, pp. 1076–1078, Apr. 2006.

[40] C. J. Engelbrecht, R. S. Johnston, E. J. Seibel, and F. Helmchen, "Ultra-compact fiber-optic two-photon microscope for functional fluorescence imaging in vivo," *Optics express*, vol. 16, pp. 5556–5564, Apr. 2008.

[41] Y. Wu, Y. Leng, J. Xi, and X. Li, "Scanning all-fiber-optic endomicroscopy system for 3D nonlinear optical imaging of biological tissues," *Optics express*, vol. 17, pp. 7907–7915, May 2009.

[42] Y. Wu, J. Xi, M. J. Cobb, and X. Li, "Scanning fiber-optic nonlinear endomicroscopy with miniature aspherical compound lens and multimode fiber collector," *Optics letters*, vol. 34, pp. 953–955, Apr. 2009.

[43] Y. Zhao, H. Nakamura, and R. J. Gordon, "Development of a versatile two-photon endoscope for biological imaging," *Biomedical optics express*, vol. 1, pp. 1159–1172, Oct. 2010.

[44] Y. Zhang, M. L. Akins, K. Murari, J. Xi, M.-J. Li, K. Luby-Phelps, M. Mahendroo, and X. Li, "A compact fiber-optic SHG scanning endomicroscope and its application to visualize cervical remodeling during pregnancy," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 109, pp. 12878–12883, Aug. 2012.

[45] G. Ducourthial, P. Leclerc, T. Mansuryan, M. Fabert, J. Brevier, R. Habert, F. Braud, R. Batrin, C. Vever-Bizet, G. Bourg-Heckly, L. Thiberville, A. Druilhe, A. Kudlinski, and F. Louradour, "Development of a real-time flexible multiphoton microendoscope for label-free imaging in a live animal," *Scientific reports*, vol. 5, p. 18303, Dec. 2015.

[46] H.-C. Park, H. Guan, A. Li, Y. Yue, M.-J. Li, H. Lu, and X. Li, "High-speed fiber-optic scanning nonlinear endomicroscopy for imaging neuron dynamics in vivo," *Optics Letters*, vol. 45, pp. 3605–3608, July 2020.

[47] C. Wang, H. Liu, J. Ma, H. Cui, Y. Li, D. Wu, Y. Hu, D. Wu, Q. Fu, L. Liang, F. Yu, R. Wu, A. Wang, and L. Feng, "Spiral scanning fiber-optic two-photon endomicroscopy with a double-cladding antiresonant fiber," *Optics Express*, vol. 29, pp. 43124–43135, Dec. 2021.

Chapter 2

Evaluation of imaging rate acceleration with noise reduction using critical imaging rate

2.1 Introduction

. Denoising models with deep learning are applied to improve the imaging rate of the developed CARS rigid endoscope. The development of deep learning has accelerated rapidly since AlexNet won a competition of ILSVRC [1]. This development has benefited from the hardware development of Graphics Processing Units and learning technical progress. Deep learning shows outstanding results than conventional models in many types of image processing, such as image recognition, object detection, semantic segmentation, and denoising [1–9]. The author focuses on denoising because obtaining sufficient image quality from noisy images observed with a high imaging rate (short exposure time) means the improvement of the imaging rate. The denoising technique using deep learning is applied to nerve imaging with the CARS

This chapter is based on the following publication:
Naoki Yamato, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto "Improvement of nerve imaging speed with coherent anti-Stokes Raman scattering rigid endoscope using deep-learning noise reduction," Scientific reports **10**(1), 15212 (2020).

rigid endoscope.

In denoising, the author mainly devised learning and evaluation methods. First, as a learning method for a few CARS endoscopic images, CARS microscopic images were used as a pre-training dataset. Then, a few CARS endoscopic images were used to fine-tune the pre-trained models. In general, more than some thousands of images is needed for training. However, a few CARS endoscopic nerve images could be prepared because the imaging rate is slow and a considerable amount of time is required. On the other hand, it is faster to obtain nerve images using CARS microscopy because an objective lens with higher NA was used for microscopy. Therefore, the author pre-trained deep learning models with a large amount of CARS microscopic images before fine-tuning them with CARS endoscopic images. Second, the author proposes a new evaluation metric, named critical imaging rate (CIR) in order to evaluate quantitatively the improvement of the imaging rate. In denoising, the performance of a new proposed model is evaluated and compared with evaluation metrics that estimate the similarity between ground truth images and denoised images. It is required to assess the improvement of an imaging rate for accelerating CARS endoscopic imaging rate. The author evaluated quantitatively the acceleration of nerve imaging rate using conventional evaluation metrics and a criterion required for medical images. This chapter's content is mainly based on the paper [10].

2.2 Pre-training of three deep learning models for denoising nerve images using CARS microscopic images

In this section, the author shows pre-training results for denoising nerve images using CARS microscopic images. The author investigates which model is optimal for denoising nerve images. CARS microscopic nerve images were utilized for evaluating the optimal denoising model and for the pre-trainig before training with endoscopic images because it is easier to obtain a large dataset of nerve images using CARS microscopy.

2.2.1 Acquisition of nerve images using CARS microscopiy

The author purchased the rabbit urinary organs including prostates and peri-prostatic fasciae from the Japan Lamb Co., Ltd. The organs were stored in 4% paraformaldehyde phosphate buffer solution (16320145, FUJIFILM Wako Pure Chemical). It is found that the fixation with the formalin does not influence the CARS signal of lipids [11, 12]. The peri-prostatic fasciae were removed from the organs, washed with PBS to remove the formalin, and placed between two cover slips enclosed with PBS to avoid dryness.

The author obtained CARS nerve images using CARS microscopy [13] shown in Fig. 2.1. An 80 MHz mode-locked Ti:sapphire laser (Tsunami, Spectra-Physics) and high-speed wavelength-tunable mode-locked Ti:sapphire laser (Megaopt) were used as excitation light sources. The former is depicted as ω_p laser, and the latter is depicted as ω_S laser in Fig. 2.1. The synchronization system is used to reduce timing jitter between two laser pulses [14]. It is necessary to irradiate two pulses simultaneously in the sample to induce the CARS process effectively. This system detects timing jitter between two pulses and adjusts the cavity length in ω_S laser. Two laser beams were overlapped spatially at a long pass filter (LP02-785RU-25, Semrock) depicted

as DM in Fig. 2.1 and inserted into the CARS microscope through a galvanometer scanner. The object lens (CFI Apo LWD, × 25, NA = 1.1, WI, Nikon) shown as OL1 in Fig. 2.1 focused the excitation lights and collected the backscattered CARS signal. The CARS signal was separated from the excitation lights using a dichroic mirror (FF665-Di02-25×36, Semrock) and optical filters (F; FF01-591/6-25, FF01-680/SP-25, Semrock; 3rd560-640, Omega; FESH0650, Thorlabs) and detected with a photomultiplier tube (PMT; R9110, Hamamatsu). The CARS signal was amplified with a current-to-voltage conversion amplifier (As-905-1, NF). The amplified CARS signals were incorporated into a desktop computer by an analog-to-digital converter (PCI-6115, 12-bit, 10 MS/s/ch, National Instruments). The wavelengths of two laser sources were set as 709 nm and 888 nm to observe the $\mathbf{CH_2}$ symmetric stretching vibration of lipids at 2845 cm^{-1}. The powers at the sample plane were 31 mW for ω_p laser and 11 mW for ω_S laser. The CARS images have a field of view of 273 μm and a resolution of 500 × 500 pixels. A total of 9100 nerve images at 91 positions were obtained, and 100 nerve images were stored at each position. The imaging rate was 66.7 images per minute (exposure time; 0.9 s). An averaged images of 100 images for each position were used as a ground truth image. Therefore, ground truth images correspond to the imaging rate of 0.667 (exposure time; 90 s).

2.2.2 Training of denoising using CARS microscopic images from scratch

The author used three deep-learning models to investigate the optimal model for denoising of nerve images. Figure 2.2 shows the architectures of three denoising models. W5 (WIN5) [7] and DN (DenoiseNet) [8] consist of convolution layers, activation layers, and skip connections. ReLU (Rectified Linear Unit) [15] was used as activation layers. These architectures learn and generate the noise distribution, and subtract the noise distribution from input images. N2N (Noise2Noise) [9] has a similar structure to U-Net [3]. N2N consists of convolution layers, activation layers, max-pooling

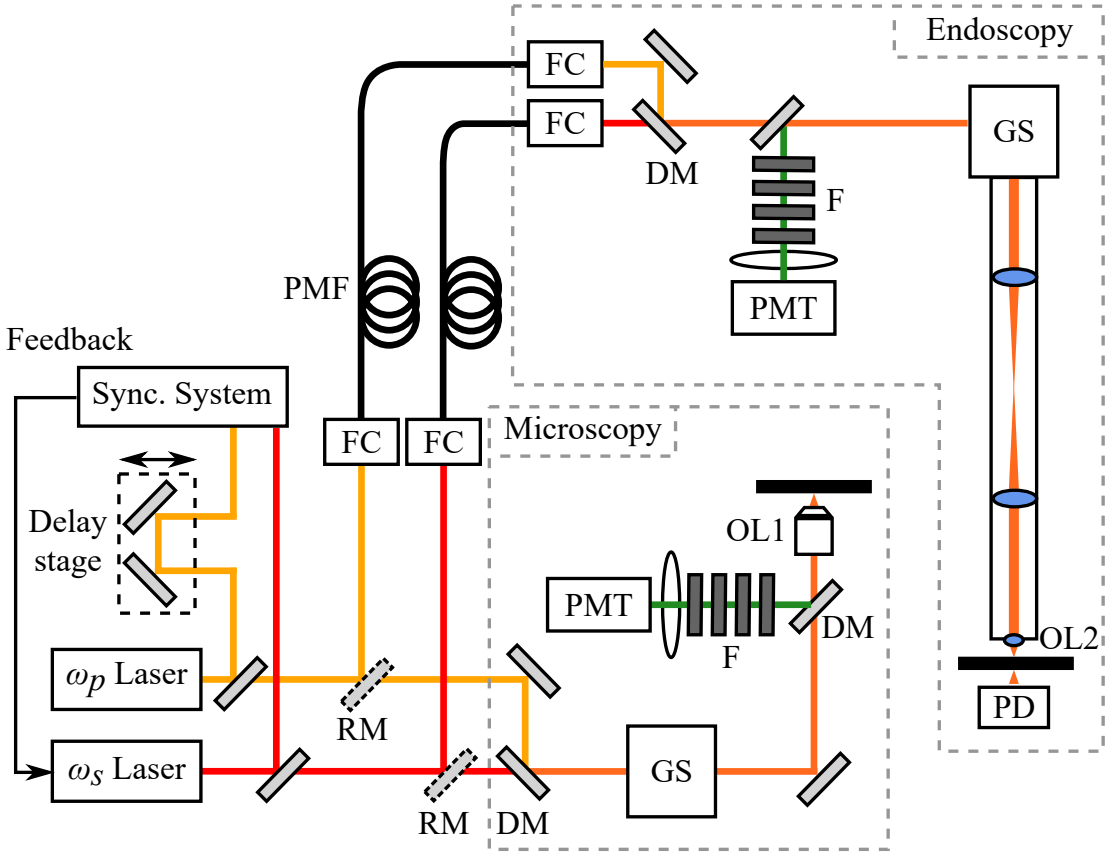


Figure 2.1: Schemantic of coherent Raman scattering microscopy and endoscopy [10]. In CARS microscopy, two ultrafast lasers are superimposed at DM. The excitation light is scanned by GS and focused by OL1. The backscattered CARS signal is separated by DM and detected with PMT. In CARS endoscopy, two ultrafast lasers are delivered through PMF. *DM* dichroic mirror, *F* optical filter, *FC* fiber coupler, *GS* Galvano mirror scanner, *OL* objective lens, *PD* photodiode, *PMF* polarization-maintaining fiber, *PMT* photomultiplier tube, *RM* removable mirror

layers, up-convolution layers and skip connections. In N2N, leaky ReLU with $\alpha = 0.1$ was used as activation layers. N2N does not require ground truth images for training. In the training of N2N, pairs of noisy images at the same position are used for a dataset. The performance of deep learning models depends on the hyperparameters, that is, the number (F) and size (K) of filters in the convolution layers and the number (L) of convolution layers. The author used a grid search to examine the best hyperparameters for this denoising task. The hyperparameters used for the grid search are summarized in Table. 2.1. N2N has twice as many filters on the encoder side in the first half as on the encoder side in the second half. F of N2N in Table. 2.1 indicates the number of filters in the convolution layers of the encoder part. The convolution layers in the decoder part except for the last three layers have 2F. F in the convolution layers in W5 and DN were fixed to 1. K in N2N was fixed to 3×3 because the performance deteriorates for larger than 3×3. The models were trained using every combination of hyperparameters and evaluated using validation datasets. The hyperparameters (F, K, L) showing the highest performance were (32, 5, 10) in DN, (48, 3, 18) in N2N, and (16, 3, 8) in W5.

Table 2.1: Hyperparameters are utilized for grid search in this work [10].

model	number of filters (F)	size of filter (K)	number of convolution layers (L)
DN	64, 32, 16 or 8	7×7, 5×5 or 3×3	25,20,15 or 10
N2N	128, 96, 64, 48 or 32	3×3	18
W5	128, 64, 32, 16 or 8	7×7, 5×5 or 3×3	10, 8, 6, 5 or 4

A mini-batch size of 32 and a loss function of mean squared error (MSE) which measures the difference between output images and ground truth images were used. MSE is given by

$$MSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{2.1}$$

In the equation, y_i and $\hat{y}_i$ mean the value at each pixel for output images ($\boldsymbol{y}$) and ground truth images ($\hat{\boldsymbol{y}}$). n is the number of all pixels. As an optimizer, Adam [16] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and a learning rate of 10^{-3} was adopted. The author used 81 nerve images as the training dataset, 5 nerve images as the validation

dataset, and 5 nerve images as the test dataset. The data augmentation of random shuffle and random cropping was used. The random cropping in the training of W5 and DN cut out 50 images with 100×100 pixels per image with 500×500 pixels. In N2N training, the images with 128×128 pixels were cropped. Therefore, 4050 nerve images for training, 250 nerve images for validation, and 250 nerve images for testing were used. The models were implemented with Pytorch framework [17] in Python on a desktop computer equipped with a Core i7-8900K CPU (Intel) and an RTX 2080Ti GPU card (NVIDIA).

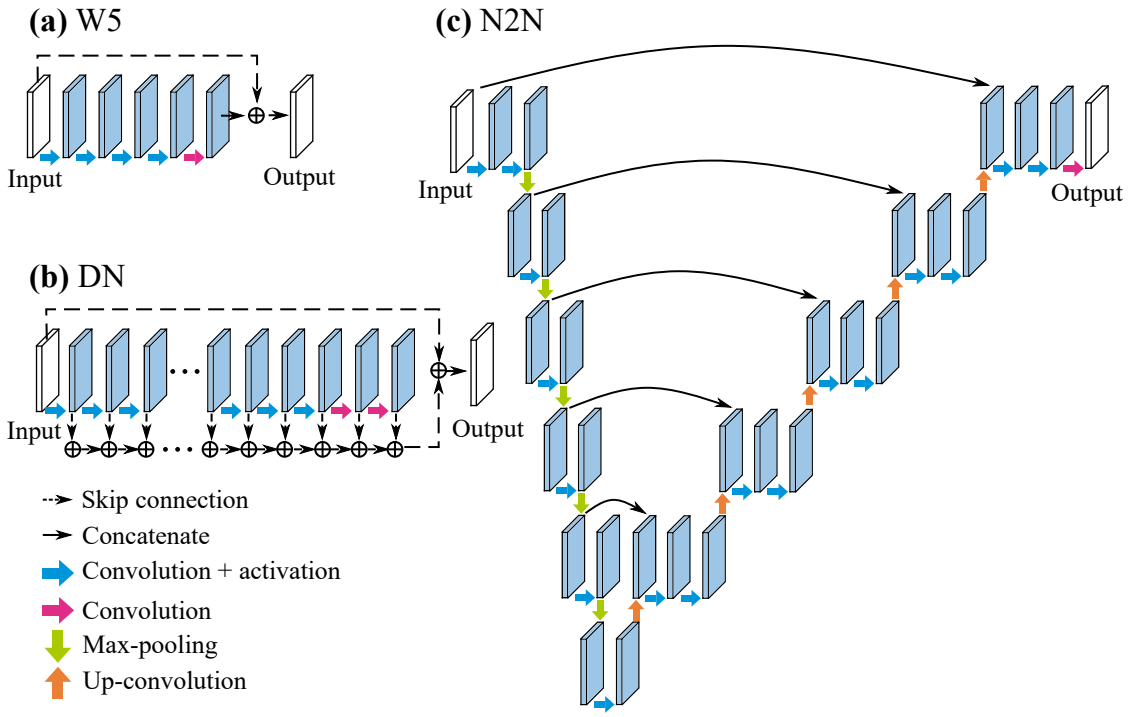


Figure 2.2: Schemantic of denoising models [10]. (a)WIN5 (W5) [7]. W5 has one skip connection. W5 learns the noise distribution. (b)DenoiseNet (DN) [8]. DN has several skip connections. As with W5, DN learns the noise distribution. (c)Noise2Noise (N2N) [9]. N2N is based on U-Net. This structure is good at learning shape information.

2.2.3 Evaluation of the quality of denoised images

The author used two evaluation metrics, peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [18], to assess the performance of each model. PSNR is expressed as

$$PSNR(\boldsymbol{y}, \hat{\boldsymbol{y}}) = 10log_{10}\frac{MAX^2}{MSE(\boldsymbol{y}, \hat{\boldsymbol{y}})}. \tag{2.2}$$

In this equation, MAX indicates the maximum value of each image. MSE has the same definition written in Eq. 2.1. SSIM is given by

$$SSIM(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{(2\mu_y\mu_{\hat{y}} + C_1)(2\sigma_{y\hat{y}} + C_2)}{(\mu_y^2 + \mu_{\hat{y}}^2 + C_1)(\sigma_y^2 + \sigma_{\hat{y}}^2 + C_2)}. \tag{2.3}$$

In this equation, μ and σ show mean and standard deviation values. C_1 and C_2 are constant values.

The Friedman test and two-tailed paired t-test were used to confirm the statistical difference between the denoising performance of models. The null hypothesis is that the mean ranks of the denoising performance are equal, and the alternative hypothesis is that at least one pair of mean metric is significantly different. The Friedman test was utilized to compare 4 models in Table 2.2. Multiple tests were used to compare the performance of denoising. The evaluation metrics between models were evaluated by a two-tailed paired t-test ($\alpha = 0.05$) with the Bonferroni-Holm method.

The CARS microscopic images denoised with the deep learning models and the conventional filter are shown in Fig. 2.3. The CARS images obtained at imaging rates of 33.3 images/min (a short exposure time of 1.8 s) for input and 0.7 images/min (a long exposure time of 90 s) for ground truth are shown in Fig. 2.3a–b. The denoised images processed with BM3D, DN, N2N, and W5 are Fig. 2.3c–f. Each result has two images: the left images show the full field of view, and the right images indicate the rectangle view in the left images. The fibrous shape in the lower half of the image is nerves. The denoised images in Fig. 2.3c–f have fewer impulsive noises than the noisy input image in Fig. 2.3a. The thick nerve fiber bundle labeled with green

arrowheads in Fig. 2.3 has good visibility for all denoised results. This indicates that the noise reduction is working well. On the other hand, the thin nerve fiber bundle labeled with yellow arrowheads cannot be recognized in Fig. 2.3c, d, and f. Only the nerve images processed by N2N show good visibility of thin nerve bundles.

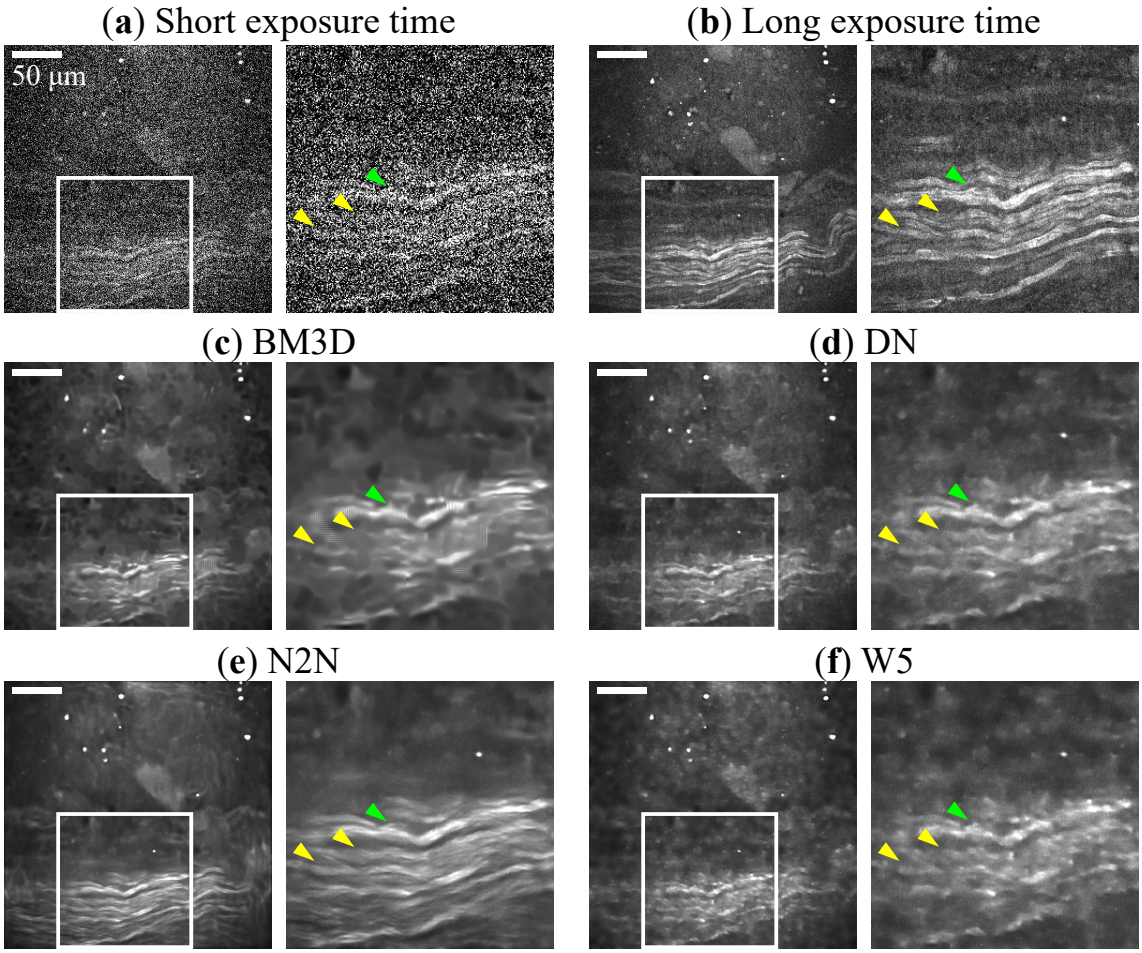


Figure 2.3: Denoising results of CARS microscopy images [10]. The images are for: (a) input observed with an imaging rate of 33.3 images/min (a short exposure time of 1.8 seconds), (b) ground truth observed with an imaging rate of 0.7 images/min (a long exposure time of 90 seconds), and images denoised with (c) the BM3D filter, (d) DN, (e) N2N, and (f) W5. Parts (a–f) are each composed of two images: the right image is a magnified image of the white rectangle in the left image. The image size is 270 × 270 μm, with 500 × 500 pixels. Nerves lie horizontally in the lower half of the images. The scale bar indicates 50 μm. The polarization direction of two laser beams is horizontal in images.

The averages and standard deviations of the two metrics are summarized in Table 2.2. The higher values of PSNR and SSIM indicate the higher similarity of denoised

Table 2.2: Comparison of denoising performance between BM3D filter and deep learning models [10]. The average and standard deviation of each metric over five test images are shown. The N2N shows a significant difference from the others in a paired t-test ($n = 5$, $P < 0.05$) using the Bonferroni-Holm correction. N2N showed the highest performance for both metrics. * means a significant difference after the Bonferroni-Holm correction ($\alpha < 0.05$)

Model	**Evaluation metrics**		Paired t-test	**P-value**	
	PSNR	SSIM		PSNR	SSIM
BM3D	31.04 ± 5.86	0.649 ± 0.207	N2N vs BM3D	0.045*	0.016*
DN	32.05 ± 6.02	0.671 ± 0.220	N2N vs DN	0.0059*	0.027*
N2N	**32.21 ± 6.04**	**0.679 ± 0.214**	N2N vs W5	0.00035*	0.016*
W5	31.96 ± 6.02	0.668 ± 0.221			

images to ground truth images. In other words, the higher performance of denoising has the higher values of the two metrics. Friedman test for 4 models indicated one of the significant differences ($P = 0.0018$ for PSNR and SSIM, respectively). All of the deep learning models have a higher performance than the BM3D filter. In the deep learning models, N2N outperformed the other two models with both metrics. According to the paired t-test ($n = 5$, $P < 0.05$) using the Bonferroni-Holm correction, the two metrics in N2N show a higher value than that in DN (PSNR: $P = 0.0059$, SSIM: $P = 0.027$) and W5(PSNR: $P = 3.5 \times 10^{-4}$, SSIM: $P = 0.016$). Therefore, N2N was used for transfer learning of CARS endoscopic images.

2.3 Transfer learning from microscopic images to endoscopic images

2.3.1 Acquisition of nerve images using CARS endoscopy

The author obtained CARS nerve images using CARS endoscopy shown in Fig. 2.1. The optical system was the same reported in [19]. The length and diameter of the rigid endoscope are 270 mm and 12 mm, respectively. The same laser sources described in section 2.2 were used as the excitation lights. Two polarization-maintaining single-

mode optical fibers (P1-630PM-FC-2 having a mode field diameter of 4.5 ± 0.5 μm at 630 nm and NA= 0.12 and P1-780PM-FC-2 a mode field diameter of 5.3 ± 1.0 μm at 850 nm and NA= 0.12, Thorlabs) deliver the two laser beams to the endoscope head individually. Two excitation lights were input by fiber couplers (PAF-X-5-B, Thorlabs) depicted as FC and collimated by zoom fiber collimators (ZC618FC-B, Thorlabs). The two laser beams are overlapped spatially after fiber couplers using a long pass filter (LP02-785RU-25, Semrock) and scanned with a pair of galvanometer mirrors (GVS002, Thorlabs). The excitation lights are focused on the sample with an objective lens (AC050-008-B-ML, Thorlabs) depicted as OL2 in Fig. 2.1. The CARS signals were detected with PMT through the same optical filters in CARS microscopy and amplified with a current-to-voltage conversion amplifier (As-905-1, NF). The CARS signals were incorporated into a desktop computer by an analog-to-digital converter (PCIe-6321, 16-bit, 250 kS/s, National Instruments). The powers of the two laser beams at the sample were 101 mW for ω_p laser and 55 mW for ω_S laser. The CARS endoscopic images have a field of view of 262 μm and a resolution of 500 × 500 pixels. A total of 500 nerve images at 5 positions were obtained, and 100 nerve images were stored at each position. The imaging rate was 37.5 images per minute (exposure time; 1.6 s). An averaged images of 100 images for each position were used as a ground truth image. Therefore, ground truth images correspond to the imaging rate of 0.374 (exposure time; 160 s). To confirm and adjust the sample position, a photodiode (PDA100A-EC, Thorlabs) depicted as PD in Fig. 2.1 was used.

2.3.2 Transfer learning from CARS microscopic images

The denoising of CARS endoscopic images is trained from the model which has the highest quality in denoising CARS microscopic images.

2.3.3 Evaluation of improvement of imaging rate

The CARS endoscopic images denoised with the deep learning models and the conventional filter are shown in Fig. 2.4. Figure 2.4b–c shows the denoised images from an input image observed at an imaging rate of 12.5 images/min (a short exposure time of 4.8 seconds) shown in Fig. 2.4a. The ground truth image obtained at an imaging rate of 0.4 images/min (a long exposure time of 160 seconds) is shown in Fig. 2.4d. The right images in each figure show the magnified view of the white rectangle in the left images. The fibrous shape in the center of the images is nerve fibers, and a spherical shape on the left side is lipid-rich tissues. The images processed with BM3D and N2N seem to be less noisy than the input image. However, The BM3D-processed image has the artifact with vertical and horizontal stripes labeled with the green narrowheads in Fig. 2.4. The intensity profiles along the color lines in Fig. 2.4a–b are shown in Fig. 2.4e. We can understand the higher denoising performance of N2N than that of BM3D according to the line profiles. The ratio of the signal A and B in 2.4e was 1.14 for BM3D, 1.41 for N2N, and 1.37 for the ground truth. Hence, N2N can keep a close value to the ground truth.

The author used nested cross-validation to assess the denoising performance for CARS endoscopic images. The evaluation metrics are summarized in Table 2.3. "with microscopy" means the results using only CARS microscopic images for the training, "with endoscopy" means the results using only CARS endoscopic images for the training, "with fine-tuning" means the results using CARS microscopic images for the pre-training and CARS endoscopic images for fine-tuning, and "with ensemble" means the results combining "with fine-tuning" with ensemble learning. The Friedman test ($\mathrm{P} < 0.05$) on 13 models indicated the significant difference (PSNR: $\mathrm{P} = 1.5 \times 10^{-7}$, SSIM: $\mathrm{P} = 2.4 \times 10^{-7}$) "with fine-tuning" shows significantly higher values than those of "with endoscopy" except for DN and the PSNR of W5R using the Bonferroni-Holm correction (DN: $\mathrm{P} = 0.020$ (PSNR), 0.15 (SSIM); N2N: $\mathrm{P} = 0.0017$ (PSNR), 0.0040 (SSIM); W5: $P = 0.023$ (PSNR), 0.0087 (SSIM)). It is found that ensemble learning

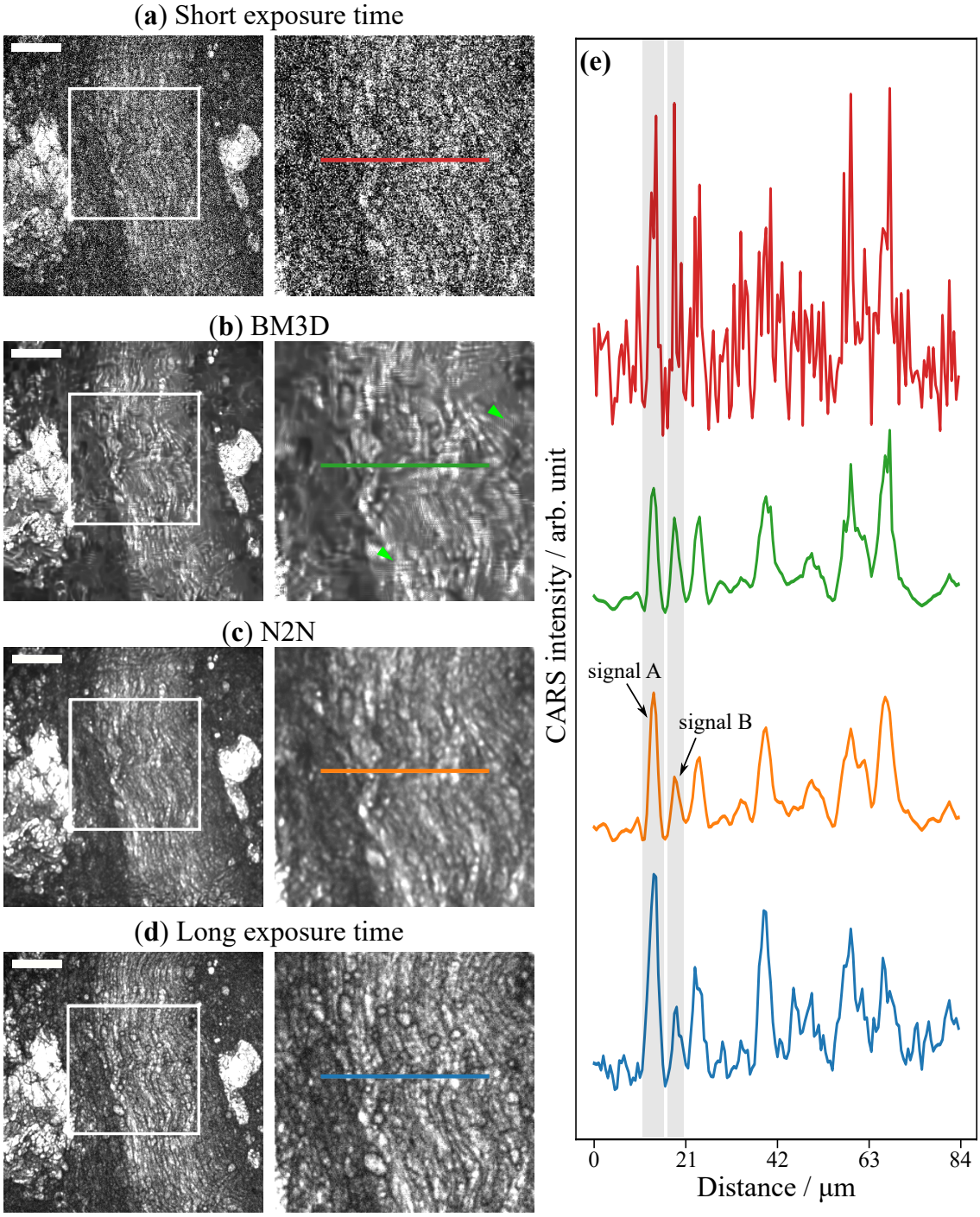


Figure 2.4: Results of denoised CARS endoscopy images [10]. The images are for (a) an input observed with an imaging rate of 12.5 images/min (a short exposure time of 4.8 s), denoised with (b) the BM3D filter and (c) N2N with ensemble, and (d) ground truth observed with an imaging rate of 0.4 images/min (a long exposure time of 160 s). Nerves lie at the center of the images with lipid-rich tissue on each side. The CARS images in the second column are images cropped from the white rectangles in the left images. The line profiles of the cropped images are shown in (e). The scale bar indicates 50 μm. The polarization direction of two laser beams is horizontal in images.

enhances denoising performance significantly (comparison between fine-tuning and ensemble in the same architecture, DN: P = 0.00070 (PSNR),0.0072 (SSIM); N2N: P = 0.00090 (PSNR), 0.0067 (SSIM); W5: P = 0.014 (PSNR), 0.000089 (SSIM)). Although P values below 0.05 exist, the no-significant difference is confirmed between the models using ensemble learning (DN versus N2N: P = 0.52 (PSNR), 0.033 (SSIM); DN versus W5R: P = 0.029 (PSNR), 0.83 (SSIM); N2N versus W5R: P = 0.064 (PSNR), 0.048 (SSIM)).

Table 2.3: Results of denoising CARS endoscopy images [10]. The CARS endoscopy images for input are at 12.5 images/min (the exposure time of 4.8 seconds). The performances of four learning styles were compared: with microscopy, with endoscopy, with fine-tuning, and with ensemble. The evaluation metrics of the models with fine-tuning were statistically significantly higher than that of the models with endoscopy, except for DN and PSNR of W5. The evaluation metrics of the models with ensemble were statistically significantly higher than that of the models with fine-tuning, except for the PSNR of W5. The comparison between the models with ensemble showed a no-significant difference, despite having even though P values were < 0.05. The models with microscopy had worse performance than the models with endoscopy in DN and N2N. All of the deep learning models with ensemble showed statistically significantly higher performance than BM3D. * means significant difference by the Bonferroni-Holm correction in a paired t-test ($n = 5$, $P < 0.05$).

Model	**Evaluation metrics**	
	PSNR	SSIM
BM3D	28.98 ± 1.12	0.690 ± 0.060
DN with microscopy	30.15 ± 1.73	0.690 ± 0.069
DN with endoscopy	30.41 ± 1.50	0.711 ± 0.052
DN with fine-tuning	30.53 ± 1.54	0.713 ± 0.053
DN with ensemble	**30.58 ± 1.69**	0.715 ± 0.058
N2N with microscopy	29.97 ± 1.64	0.680 ± 0.070
N2N with endoscopy	30.33 ± 1.50	0.710 ± 0.055
N2N with fine-tuning	30.51 ± 1.53	0.718 ± 0.054
N2N with ensemble	30.56 ± 1.67	**0.720 ± 0.059**
W5 with microscopy	30.31 ± 1.75	0.698 ± 0.066
W5 with endoscopy	30.41 ± 1.51	0.712 ± 0.0520
W5 with fine-tuning	30.47 ± 1.52	0.714 ± 0.052
W5 with ensemble	30.50 ± 1.66	0.714 ± 0.057

Paired t-test	**P-value**	
	PSNR	SSIM
DN		
endoscopy vs fine-tuning	0.020	0.15
fine-tuning vs ensemble	0.00070*	0.0072*
N2N		
endoscopy vs fine-tuning	0.0017*	0.0040*
fine-tuning vs ensemble	0.00090*	0.0067*
W5		
endoscopy vs fine-tuning	0.023	0.0087*
fine-tuning vs ensemble	0.014	0.000089*
Comparison of ensemble models		
DN vs N2N	0.52	0.033
DN vs W5	0.029	0.83
N2N vs W5	0.064	0.048
BM3D vs with ensemble		
BM3D vs DN	0.0032*	0.00014*
BM3D vs N2N	0.0030*	0.0000019*
BM3D vs W5	0.0032*	0.00029*

The author proposed a critical imaging rate (*CIR*) as a new metric to estimate the improvement of the imaging rate quantitatively. The *CIR* is composed of a harmonic mean of $\boldsymbol{CIR}_{\mathrm{PSNR}}$ and $\boldsymbol{CIR}_{\mathrm{SSIM}}$, which is calculated at the imaging rates meeting **PSNR** = 30 dB and **SSIM** = 0.8, respectively. The *CIR* is given by

$$\boldsymbol{CIR} = \frac{2}{\frac{1}{\boldsymbol{CIR}_{\mathrm{PSNR}}} + \frac{1}{\boldsymbol{CIR}_{\mathrm{SSIM}}}}. \tag{2.4}$$

Generally, a standard criterion of **PSNR** $\geq$ 30 [20] or **SSIM** $\geq$ 0.8 [21] is used to indicate the high denoising performance. Therefore, the harmonic mean of averaging the imaging rates of $\boldsymbol{CIR}_{\mathrm{PSNR}}$ and $\boldsymbol{CIR}_{\mathrm{SSIM}}$ is utilized for the evaluation. Figure 2.5a–b shows the relationships between the imaging rates and each evaluation metric. The blue and orange plots indicate the imaging rates for raw CARS images and denoised CARS images using N2N "with ensemble". N2N models were trained for denoising at each imaging rate. At the star plots at 12.5 images/min, the denoising model with deep learning improves to the evaluation metrics of 30.56 ± 1.67 dB (PSNR) and 0.720 ± 0.059 (SSIM) from the evaluation metrics of 19.85 ± 1.40 dB (PSNR) and 0.232 ± 0.027 (SSIM). The denoising of CARS nerve images provides large improvements at high imaging rates. The $\boldsymbol{CIR}_{PSNR}$ and $\boldsymbol{CIR}_{SSIM}$ for the raw CARS images (blue bars) and the denoised CARS images (orange bars) are summarized in Fig. 2.5c. The $\boldsymbol{CIR}_{PSNR}$ and $\boldsymbol{CIR}_{SSIM}$ increased to 12.5 and 4.8 images/min from 1.6 and 1.2 images/min, respectively. As a result, it was demonstrated that the denoising speeds up the *CIR* by a factor of 5 from 1.4 images/min to 7.0 images/min.

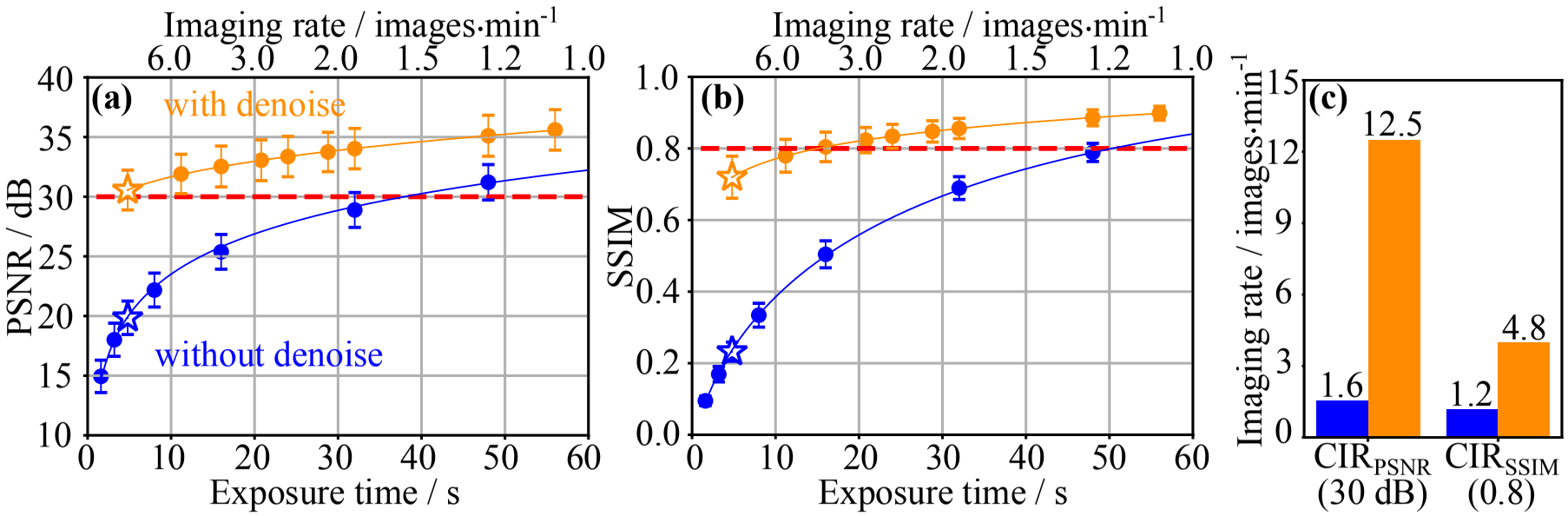


Figure 2.5: Evaluation of *CIR* with plots of PSNR (a) and SSIM (b) [10]. Some modifications are added from [10]. The blue solid circles are the values of evaluation metrics for raw images observed with different imaging rates (exposure times). The orange solid circles are the values of the evaluation metrics for images denoised with N2N with ensemble. The criteria **PSNR** = 30 dB and **SSIM** = 0.8 are plotted as the red dashed lines. The star markers in (a) and (b) correspond the evaluation metrics at 12.5 images/min including Fig. 2(c). (c) The $\mathbf{CIR}_{\mathbf{PSNR}}$ and $\mathbf{CIR}_{\mathbf{SSIM}}$ for **PSNR** = 30 dB and **SSIM** = 0.8 with (orange) and without denoising (blue), respectively.

The author employed the fine-tuning strategy using CARS microscopic images as the pre-training dataset. Besides, the author confirmed how the amount of pre-training images affects on the denoising performance of N2N with fine-tuning. The results of fine-tuned N2N changing the amount of pre-training images are shown in Fig. 2.6 The shown results correspond to test images obtained using CARS endoscopy at 12.5 images/min (exposure time of 4.8 seconds). The evaluation metrics for N2N trained from scratch are shown as blue plots, and The evaluation metrics for fine-tuned N2N are shown as green plots. The blue plots are the results of N2N with endoscopy, the green plots are N2N with fine-tuning shown in Table 2.3. The amount of pre-training images was changed from 10 to 70. The pre-training images were randomly chosen from 81 images 5 times. Each plot and error bar means the average and the standard deviation between 5 times training. The average of evaluation metrics is improved as the pre-training images increase. Therefore, the more pre-training images the model learns, the more the effect of pre-training is demonstrated. Of course, it will come to a head at some pre-training images.

The nerve shape recognition of deep learning models seems to be essential for denoising of nerve images. W5 has a skip connection and some convolution layers. In W5, the output images are generated by adding the output of the last convolution layer and the input images. W5 learns to cancel out the noise in the input images. W5 is considered to learn little information about the nerve shape due to the shallow layers. Although DN has similar architecture to W5, DN has more skip connections to cancel out the noise precisely. The outputs from the deeper laers seems to restore specific textures in the input images. N2N is composed of encoder-decoder architecture similar to U-Net, which is famous with the high shape recognition ability. The encoder of N2N extracts the features of the input image at various resolution and the extracted features are converted to the output images by the decoder. The evaluation metrics of the models “with microscopy” are compared with that “with endoscopy” in each architecture in Table 2.3. Although W5 showed small differences (PSNR: 0.10; SSIM: 0.014) between “with microscopy” and “with endoscopy”,

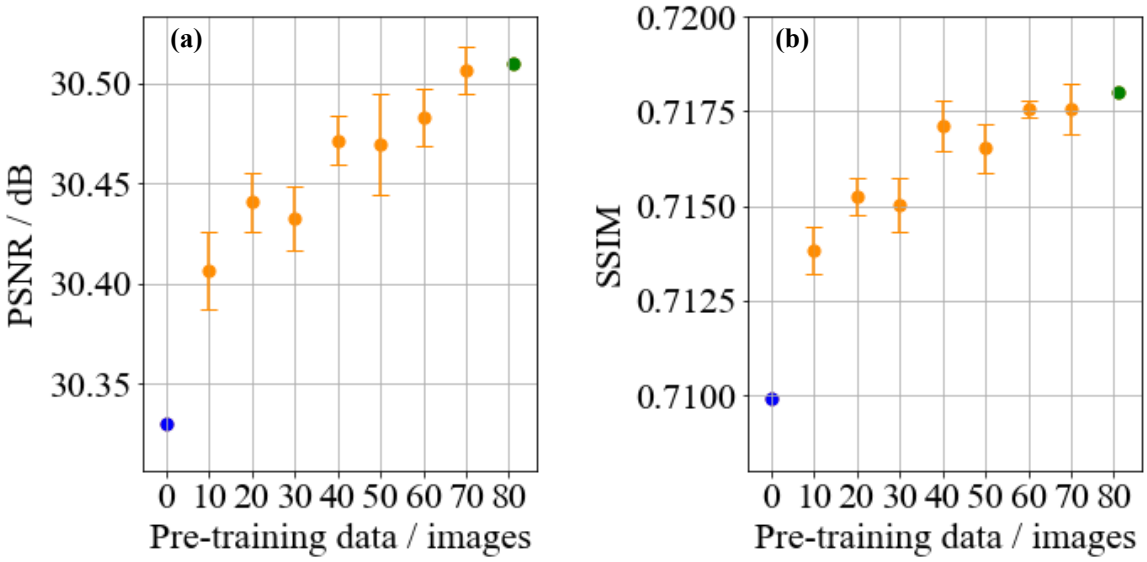


Figure 2.6: Results of N2N with fine-tuning for the amount of pre-training images of CARS microscopy [10]. (a) PSNR (b) SSIM. The plot and error bar means the average and standard deviation between 5 times training, respectively. The blue plots show the result of N2N from scratch. The green plots show the result of fine-tuned N2N.

DN showed large differences (PSNR: 0.26, SSIM: 0.021), and N2N also showed large differences (PSNR: 0.36, SSIM: 0.030) in Table 2.3. The nerve images acquired with CARS endoscopy seems to be different from those acquired with CARS microscopy due to the difference of NA (microscopy; NA = 1.1, endoscopy; NA = 0.26). Sectioned nerve images are obtained with CARS microscopy using the high NA objective lens. On the other hand, nerve fibers obtained with CARS endoscopy seem to overlap due to the low axial resolution by the low NA objective lens. The author considered that the evaluation metrics of DN and N2N "with microscopy" became lower values because of the differences in the axial resolution. In other words, DN and N2N learned the shape feature about nerves. The author believes that N2N will superior to learn shape information by utilizing its architecture, and will be more effective for applying to the observation of specific organs.

There are two advantages to implement a denoising approach with deep learning for medical settings. First, the processing speed of deep learning models is faster

than that of the conventional filter. The processing times with each model are summarized in Table. 2.4. The processing time of N2N is shorter than that of DN due to the pooling layers in N2N. At each pooling layer the image size is halved. Therefore, computational costs are significantly reduced. For ensemble learning, the four models could be implemented using parallel processing. The processing time for denoising with deep learning models was negligible because the computation time is considerably shorter than the exposure time. Second, the biggest advantage of N2N is the possibility of updating the model with retraining during surgery. Fine-tuning of the denoising model is needed when introducing the model for the first time, for infrequent surgery, or when the conditions of the optical system change. It is difficult to prepare adequate data and fine-tune the model at all such times because obtaining the ground truth data for a long time is laborious. A learning style that does not involve observing the ground truth allows for in situ fine-tuning in a situation where an image is taken two times at the same position. The possibility of maintaining the model by fine-tuning is important for surgical use. Currently, off-line processing is being performed, but we will demonstrate the improvement of imaging rate by on-line processing in the future.

Table 2.4: Denoising processing time for one image with BM3D filter and three deep learning architectures [10]. The average and standard deviation from 1000 measurements are shown.

model	averaged time	parameters
BM3D	5048 ± 35 ms	-
DN	20.0 ± 1.9 ms	0.224 M
DN with ensemble	79.4 ± 0.6 ms	0.897 M
N2N	15.2 ± 1.7 ms	1.342 M
N2N with ensemble	59.6 ± 3.0 ms	5.366 M
W5	3.3 ± 0.7 ms	0.019 M
W5 with ensemble	12.6 ± 0.6 ms	0.075 M

2.4 Conclusion

The author demonstrated that deep-learning-based denoising accelerates the nerve imaging rate using CARS rigid endoscope. Deep learning models for denoising were compared using CARS microscopic images to determine the optimal model for denoising of nerve images. N2N has a quantitatively and qualitatively high performance of denoising for CARS microscopic images. N2N trained with a fine-tuning strategy restored the detailed structure faithfully to the ground truth image. The *CIR* after denoising with deep learning increased to 7.0 images/min ($\boldsymbol{CIR}_{\mathrm{PSNR}}$: 12.5 images/min, $\boldsymbol{CIR}_{\mathrm{SSIM}}$: 4.0 images/min) from 1.4 images/ min ($\boldsymbol{CIR}_{\mathrm{PSNR}}$: 1.6 images/min, $\boldsymbol{CIR}_{\mathrm{SSIM}}$: 1.2 images/min).

The author adopted the fine-tuning strategy for training deep-learning models with CARS microscopic images as the pre-training data. The fine-tuning strategy from CARS microscopic images to CARS endoscopic images was significantly useful for most cases by learning denoising CARS images obtained with the same size under the same photomultiplier conditions. Thus, the author concluded that it is effective to employ pre-training using nerve images obtained with CARS microscopy which is less time-consuming and laborious than CARS endoscopy. In addition to the effect of pre-training, the author investigated the dependence of the performance on the amount of pre-training images (data is shown in the Appendix). The more large dataset of nerve images will improve the denoising performance according to the result in the Appendix. As with other models, generative adversarial networks (GAN), which boost the performance by competing using two networks of generator and discriminator, are expected to offer superior performance.

The author proposed the *CIR* as a new evaluation metric for assessing the imaging rate of denoised and row CARS images. The definition of the *CIR* is a harmonic mean of $\boldsymbol{CIR}_{\mathrm{PSNR}}$ and $\boldsymbol{CIR}_{\mathrm{SSIM}}$, where a criterion of PSNR= 30 and SSIM = 0.8 are satisfied. The criterion is considered to be needed for medical images. The PSNR, SSIM, and mean squared error (MSE) are used as evaluation metrics to assess the denoising

performance. However, these metrics cannot assess the imaging rate directly. Thus, the author proposed *CIR* combining these metrics.

Reference

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, May 2017.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Springer International Publishing, 2015.

[4] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6230–6239, IEEE, 2017.

[5] T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, A. Dovzhenko, O. Tietz, C. Dal Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox, and O. Ronneberger, "U-Net: deep learning for cell counting, detection, and morphometry," *Nature methods*, vol. 16, pp. 67–70, Jan. 2019.

[6] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Advances*

in Neural Information Processing Systems, vol. 29, Curran Associates, Inc., Mar. 2016.

[7] P. Liu and R. Fang, "Wide Inference Network for Image Denoising via Learning Pixel-distribution Prior," July 2017.

[8] T. Remez, O. Litany, R. Giryes, and A. M. Bronstein, "Deep class-aware image denoising," in *2017 International Conference on Sampling Theory and Applications (SampTA)*, pp. 138–142, July 2017.

[9] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2Noise: Learning Image Restoration without Clean Data," Mar. 2018.

[10] N. Yamato, H. Niioka, J. Miyake, and M. Hashimoto, "Improvement of nerve imaging speed with coherent anti-Stokes Raman scattering rigid endoscope using deep-learning noise reduction," *Scientific reports*, vol. 10, p. 15212, Sept. 2020.

[11] R. Galli, O. Uckermann, E. Koch, G. Schackert, M. Kirsch, and G. Steiner, "Effects of tissue fixation on coherent anti-Stokes Raman scattering images of brain," *Journal of biomedical optics*, vol. 19, no. 7, p. 071402, 2013.

[12] S. M. Levchenko, X. Peng, L. Liu, and J. Qu, "The impact of cell fixation on coherent anti-stokes Raman scattering signal intensity in neuronal and glial cell lines," *Journal of biophotonics*, vol. 12, p. e201800203, Jan. 2019.

[13] H. Cahyadi, J. Iwatsuka, T. Minamikawa, H. Niioka, T. Araki, and M. Hashimoto, "Fast spectral coherent anti-Stokes Raman scattering microscopy with high-speed tunable picosecond laser," *Journal of biomedical optics*, vol. 18, p. 096009, Sept. 2013.

[14] T. Minamikawa, N. Tanimoto, M. Hashimoto, T. Araki, M. Kobayashi, K. Fujita, and S. Kawata, "Jitter reduction of two synchronized picosecond mode-

locked lasers using balanced cross-correlator with two-photon detectors," *Applied physics letters*, vol. 89, p. 191101, Nov. 2006.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), pp. 807–814, Omnipress, June 2010.

[16] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Curran Associates Inc., Oct. 2017.

[18] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, vol. 13, pp. 600–612, Apr. 2004.

[19] K. Hirose, S. Fukushima, T. Furukawa, H. Niioka, and M. Hashimoto, "Invited Article: Label-free nerve imaging with a coherent anti-Stokes Raman scattering rigid endoscope using two optical fibers for laser delivery," *APL Photonics*, vol. 3, p. 092407, Sept. 2018.

[20] R. Shyam Sunder, C. Eswaran, and N. Sriraam, "Medical image compression using 3-D Hartley transform," *Computers in biology and medicine*, vol. 36, pp. 958–973, Sept. 2006.

[21] S. Pudlewski and T. Melodia, "Compressive Video Streaming: Design and Rate-Energy-Distortion Analysis," *IEEE Transactions on Multimedia*, vol. 15, pp. 2072–2086, Dec. 2013.

Chapter 3

Nerve segmentation using transfer learning from fluorescence images to CARS images

3.1 Introduction

. Automated nerve extraction from CARS endoscopic images by computers will reduce the burden on surgeons in nerve-sparing surgery and contribute to further progress toward safer surgery. It is necessary to distinguish the nerve areas from other lipid-rich tissues in CARS images because the CARS images using the $\mathbf{CH_2}$ stretching vibration contains information about all lipids as well as nerves. Training to recognize new images is a barrier to the introduction of new modalities into medical settings. Assisting in image interpretation will reduce the burden on the surgeon and will contribute to improving the safety of endoscopic surgery.

In this chapter, the author mainly devised a pre-training method using fluorescence images. The author demonstrated automated nerve segmentation from CARS

This chapter is based on the following publication:
Naoki Yamato, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto "Nerve Segmentation with Deep Learning from Label-Free Endoscopic Images Obtained Using Coherent Anti-Stokes Raman Scattering," Biomolecules **10**(7), 1012 (2020).

images with deep-learning models. Generally speaking, deep-learning models require a large dataset. However, it is laborious and time-consuming to prepare a large number of CARS endoscopic nerve images because it takes a long time to find nerves using CARS endoscopy. Therefore, the author proposed transfer learning using fluorescence images of nerves as pre-training data. The author prepared the fluorescence images labeling lipid in order to have similar information to CARS images. This chapter's content is mainly based on the paper [1].

3.2 Transfer learning of Segmentation from fluorescent nerve images to CARS nerve images

In this section, the author investigates nerve extraction using fluorescence images as a pre-training dataset. Although a large dataset is required for the training of deep learning models, it is laborious and time-consuming to obtain a large number of CARS endoscopic nerve images. This is because the sample has a few peripheral nerves which are similar to an actual tissue, and it is hard to find peripheral nerves using the CARS endoscope. CARS endoscope has a low NA objective lens to observe a large field at once. The higher the NA, the more efficient the CARS process. Therefore, it requires a long time to confirm whether nerves exist or not. On the other hand, the speed of fluorescence imaging is faster than that of CARS endoscopy and it is easy to handle. The author prepared a large amount of fluorescence nerve images and utilized that as a pre-training dataset.

3.2.1 Acquisition of nerve images using fluorescent and CARS endoscopic images

The rabbit urinary organs described in chapter 2 were used as the sample. The storage and preparation for the observation were the same. For fluorescence imaging, a staining process described below was added.

fluorescence imaging

The author doubly stained the prostate fasciae resected from the urinary organs to prepare fluorescence images for a training dataset. FluoroMyelin™ Green F34651 (Ex/Em: 479/598 nm, Thermo Fisher Scientific, Waltham, MA, USA) was used to visualize the myelin sheath.

Hoechst 33342 (Ex/Em: 352/461 nm, Dojindo) stains cell nuclei. The staining procedure was the following 3 steps. First, the author diluted 20 L of FluoroMyelin™ Green and 20 L of Hoechst 33342 to 300-fold in PBS. Second, the fasciae were placed in the staining solution for 20 minutes at room temperature. Third, the solution was washed out using an ultrasonic washing machine for 10×3 minutes in PBS. After that, the fasciae were enclosed between two coverslips using a procedure similar to that used for CARS imaging.

A confocal fluorescence microscope (Eclipse Ti, Nikon, Tokyo, Japan) was used to prepare z-stack fluorescence images of nerves. The timing-shifted mode in the microscope detected the fluorescence signals of FluoroMyelin™ Green and Hoechst 33342. The wavelength of the excitation and detection light for FluoroMyelin™ Green was 488 nm and 605–1000 nm, respectively. That for Hoechst 33342 was 405 nm and 417–477 nm, respectively. The excitation lights were focused by an objective lens (Plan Apo, ×10, NA = 0.45, Nikon). The stepping width for each stack was 1 μm. The fluorescence images have a field of view (1270×1270 μm^2) and a resolution (1024×1024 pixels). One hundred fifty-three fluorescence images of nerves were acquired.

Each fluorescence image was divided into 16 parts with no overlapping. The author removed the fluorescence images that were difficult to identify nerves. The fluorescence dataset of 1818 images with the FOV of (317×317 μm^2) and the resolution of 256×256 pixels was prepared.

The fluorescence images for example are shown in Fig. 3.1. A FluoroMyelin™ Green image (Fig. 3.1a), a Hoechst image (Fig. 3.1b), and a transmission image

(Fig. 3.1c) are shown. Although the peripheral nerves were visualized clearly in Fig. 3.1a, non-nerve tissues were also visible. The ground truth image shown in Fig. 3.1d was manually prepared; the non-nerve tissues were removed from Fig. 3.1a and those images were binarized by thresholding. The Hoechst image was used to distinguish the blood vessels and nerves in the neurovascular bundles. The author focused on the nuclei density indicated by white arrowheads in 3.1b.

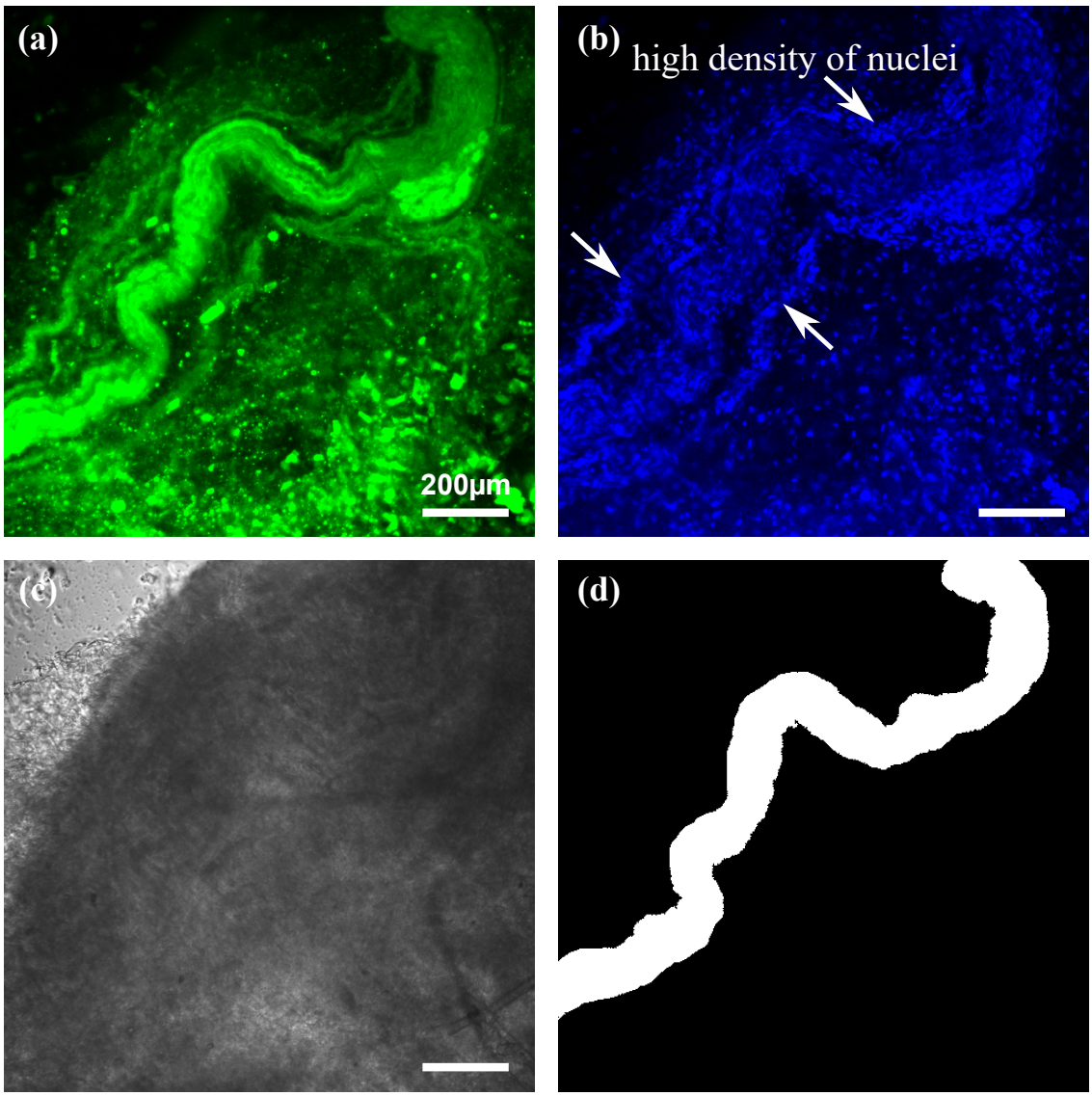


Figure 3.1: Confocal fluorescence microscopy imaging of rabbit peri-prostatic nerve [1]. (**a**) FluoroMyelin^{TM} Green image. (**b**) Hoechst image. (**c**) Transmission image. (**d**) Ground truth image prepared based on fluorescence images. The scale bar is 200 μm.

CARS imaging

The author prepared the CARS images of peripheral nerves using developed CARS rigid endoscope [2, 3]. The same optical setup in chapter 2 was used. A total of 24 CARS images from two rabbit prostates were obtained. Two barrels were used to observe peripheral nerves. The one barrel has a length of 550 mm and an objective lens with a focal length of 7.5 mm. A total of 16 CARS images with a FOV of 317×317 μm^2 and a resolution of 256×256 pixels were prepared. The other barrel has a length of 270 mm and an objective lens with a focal length of 3.1 mm. Three CARS images with a FOV of 317×317 μm^2 and a resolution of 256×256 pixels and 5 CARS images with a FOV of 317×317 μm^2 and a resolution of 256×256 pixels were acquired. The five CARS images were resized to a resolution of 512×512 pixels using symmetric padding and halved to a half resolution of 256×256 pixels using the area averaging method. These CARS images were acquired for an exposure time of 330 s (the former 19 CARS images) and 160 s (the latter 5 CARS images).

The acquired CARS images are shown in Fig. 3.2. The transmission and CARS images are placed in the left and middle columns, respectively. The images with nerves, non-nerves and both tissues are shown in the upper, middle and lower rows, respectively. The fibrous structure with the high intensity is peripheral nerves in 3.2(b and h). The spherical structure seems to be lipid-rich tissues in 3.2(e and h). The CARS image in Fig. 3.2(h) has both nerve and non-nerve tissues. In transmission images in Fig. 3.2(a and g), the characteristic fibrous structure of nerves cannot be seen. This is because the peripheral nerves are almost transparent. Therefore, the contrast enhancement of peripheral nerves is required in medical settings to confirm nerve positions. The ground truth images for nerve segmentation are shown in Fig. 3.2(c, f and i).

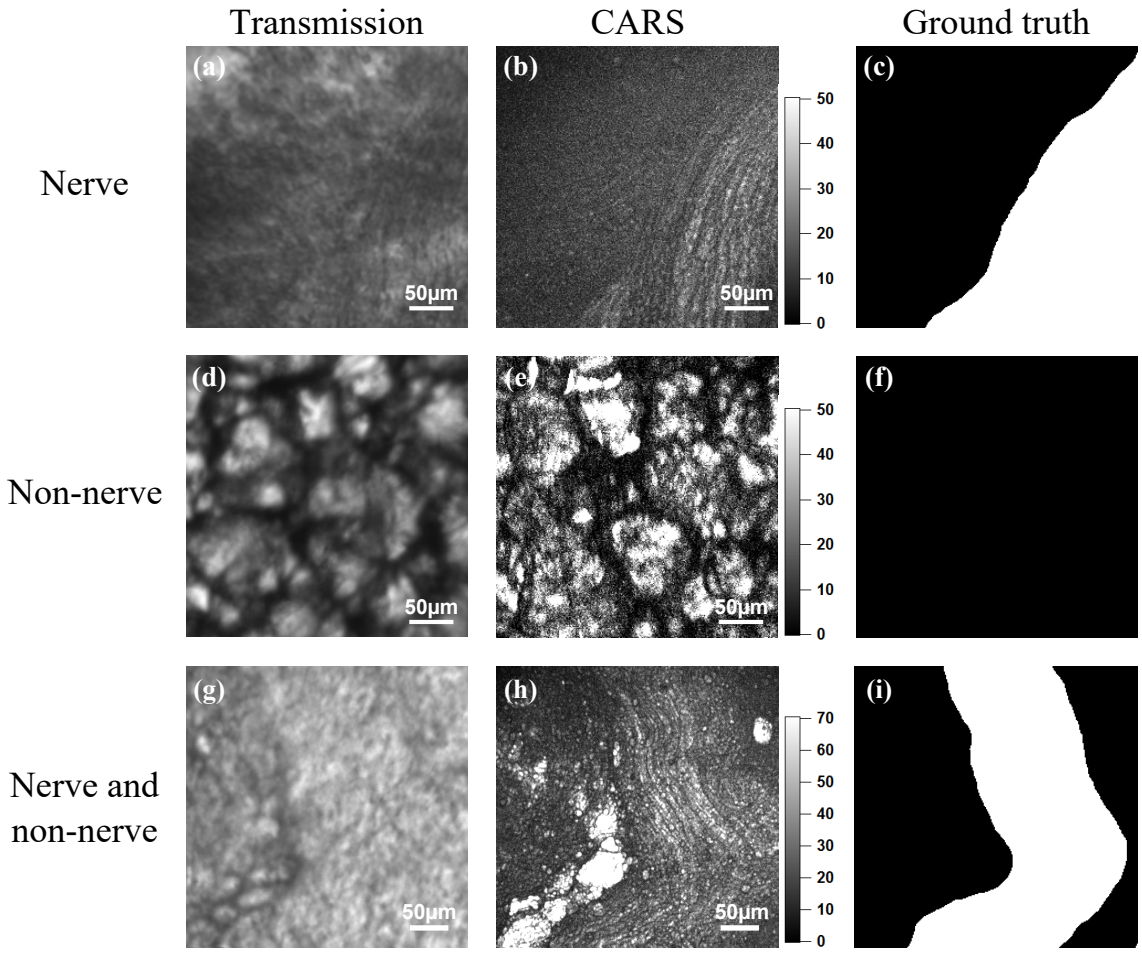


Figure 3.2: CARS endoscopic imaging of rabbit peri-prostatic fascia [1]. (**a**,**d**,**g**) Transmission images. (**b**,**e**,**h**) CARS rigid endoscopy images. (**c**,**f**,**i**) Ground truth images. (**a**-**c**) A sample containing only nerves emitting large CARS signals. (**d**-**f**) A sample containing only non-nerve tissue emitting strong CARS signals. (**g**-**i**) A sample containing both nerves and non-nerve tissues emitting CARS signals.

3.2.2 Transfer learning method

U-Net [4,5], which is famous in semantic segmentation, was used for nerve extraction. The architecture of U-Net is shown in Fig. 3.3. U-Net has an encoder part labeled pink and a decoder part labeled blue. The author applied the weight of a VGG16 encoder as the encoder part of U-Net [6]. VGG16, one of the classification models, has simple architecture and shows high performance. The final layer of U-Net outputs the probability of nerve with a value of 0–1. In this study, the author used a threshold of 0.5 to binarize the output images. Pixels with a value greater than 0.5 are identified as nerves.

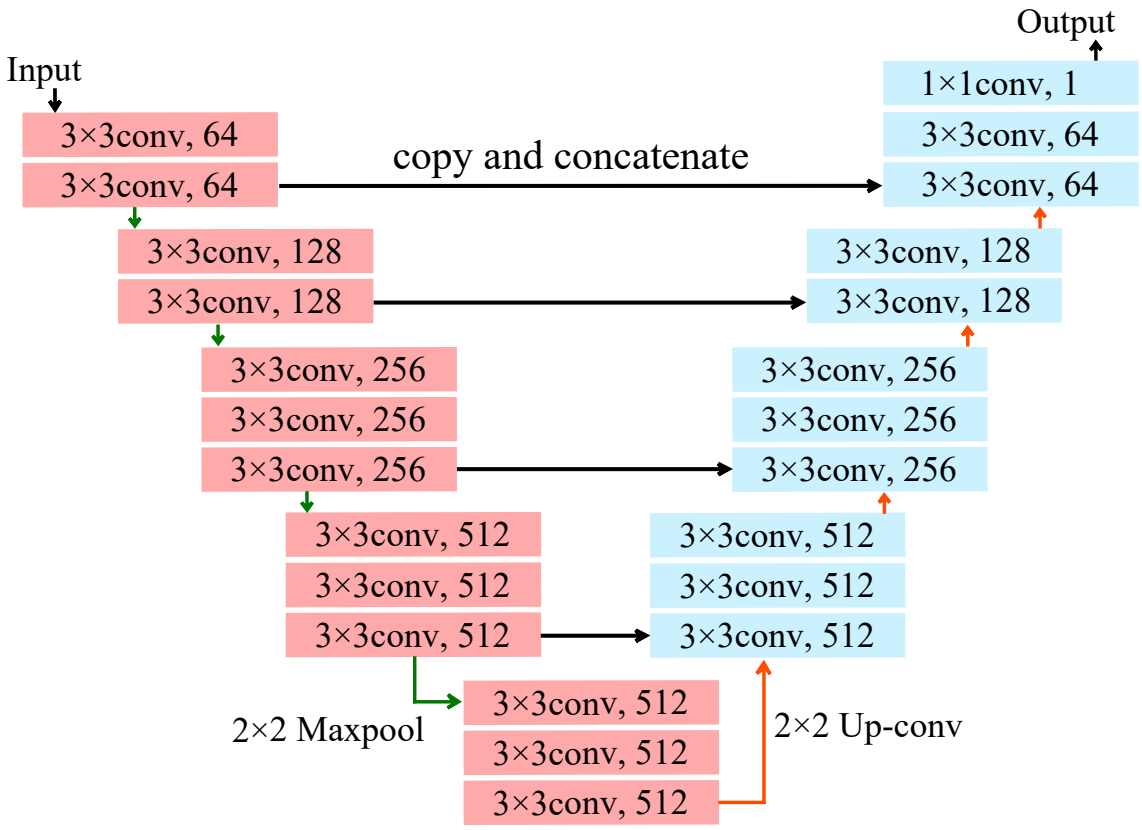


Figure 3.3: Network architecture of U-Net with VGG16 encoder [1].

A loss function used for training segmentation models is given by

$$Loss = 1 - sensitivity \times specificity, \tag{3.1}$$

$$sensitivity = \frac{TP}{TP + FN}, \tag{3.2}$$

$$specificity = \frac{TN}{TN + FP}. \tag{3.3}$$

In the equation, true positive (TP) and true negative (TN) means the number of pixels correctly predicted as nerves and non-nerves, false positive (FP) and false negative (FN) means the number of pixels wrongly predicted as nerves and non-nerves, respectively. The Adam optimizer [7] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a learning rate of 0.001 was used. The author utilized He's initialization method [8] to initialize the parameters of convolution layers. The maximum number of epochs that determined the number of the training process was set as 200. The early stopping with the number 20 was also used, where the training was finished when validation loss was not updated for 20 consecutive epochs. The data augmentation of horizontal and vertical flip and rotation (90 and 180 degrees) was used to prevent overfitting.

3.2.3 Evaluation of the effect of transfer learning

Multiple tests were conducted for assessing the performance of three schemes (Scheme I–III). Scheme I means the training of U-Net using CARS images from scratch. Scheme II means the training of U-Net with VGG16 encoder using CARS endoscopic images. Scheme III means the pre-training of U-Net with VGG16 encoder using fluorescence images and fine-tuning CARS endoscopic images. The Bonferroni-Holm method corrects a significant level for multiple comparisons. The performance between Scheme III and III' was tested. Scheme III' means the addition of ensemble learning and median filtering to Scheme III.

The evaluation metrics of mean accuracy and F_1 value were used to quantitatively

assess the nerve segmentation quality. These metrics are expressed as

$$meanaccuracy = \frac{sensitivity + specificity}{2}, \tag{3.4}$$

$$F_1value = \frac{2 \times sensitivity \times specificity}{sensitivity + specificity}, \tag{3.5}$$

$$precision = \frac{TP}{TP + FP}. \tag{3.6}$$

Sensitivity and precision were calculated from all output images in a one-fold test set. This is because TP and FN become 0 for the images including only non-nerve areas. The paired t-test (two-sided test) was used to compare the performance on the same dataset.

The metric of F_1 value is a key indicator for evaluating performance. F_1 value, called the Dice coefficient, is composed of a harmonic mean of sensitivity (recall) and precision. A higher value of both metrics is needed to extract nerve areas without excess or deficiency.

The pre-training of the fluorescence images was assessed using 10-fold cross-validation. The 1,818 fluorescence images were divided into 10 folds randomly. One fold in 10 folds was used as the test set, which was not related to the pre-training. The other fold and the eight folds were assigned as the validation and training sets, respectively. The pre-training was repeated until each fold in nine folds except for the test set was used as the validation set. Therefore, the training was conducted 9 times. The author evaluated the performance by the averaged metrics of the nine models.

The training with the CARS endoscopic images was assessed using 8-fold nested cross-validation [9]. The 24 CARS endoscopic images were divided into 8 folds randomly. The training was repeated until each fold in eight folds was used as the test and validation sets. Therefore, the training was conducted 56 times. The author

evaluated the performance by the averaged metrics of the 56 models.

In scheme III', ensemble learning and median filter were applied. An overview of ensemble learning was depicted in Fig. 3.4. A total of 56 models were generated after 8-fold nested cross-validation. These models were divided into 8 groups (each group is marked with a red rectangle); each group had the same test set and a slightly different training set. The output images for each group were made by a majority decision of the outputs from the 7 models and filtered by a median filter with a 37×37 kernel.

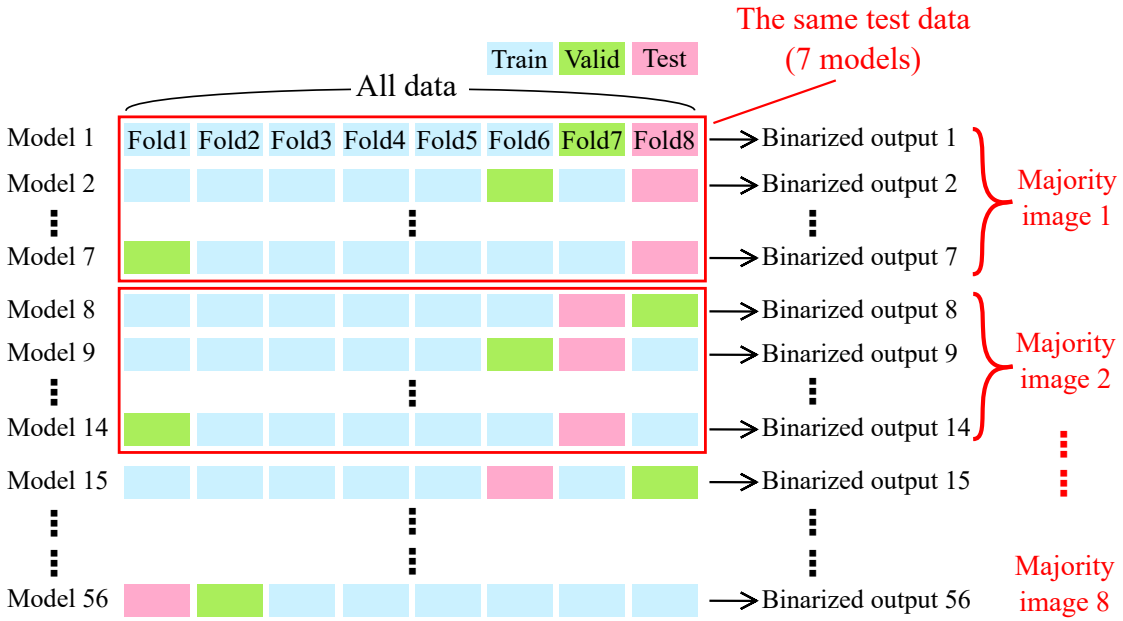


Figure 3.4: Overview of ensemble learning (majority voting strategy) [1].

The results of nerve segmentation with three learning schemes are shown in Fig. 3.5. The columns of Input and Ground truth show the input and ground truth images of CARS images. The column of Scheme I shows the output images of the U-Net trained with only CARS endoscopic images from scratch. The column of Scheme II presents the output images of the U-Net with the VGG16 model whose decoder was trained with CARS endoscopic images. The column of Scheme III displays the output images of the U-Net with the VGG16 model whose decoder was trained with fluorescence images and CARS endoscopic images. The column of Scheme III' shows the output images with ensemble learning and median filtering applied to the images of Scheme III. In Scheme I, excess non-nerve areas are determined to be nerve areas. Conversely, there were cases where many regions were predicted to be non-nerve tissues (not shown). Scheme II and Scheme III improve the segmentation quality than Scheme I. These results indicate that applying the VGG16 encoder (Scheme II) improves nerve extraction and pre-training with the fluorescence images (Scheme III) further boosts the performance. In addition, the post-processing (Scheme III') enhances the smoothness of nerve extraction.

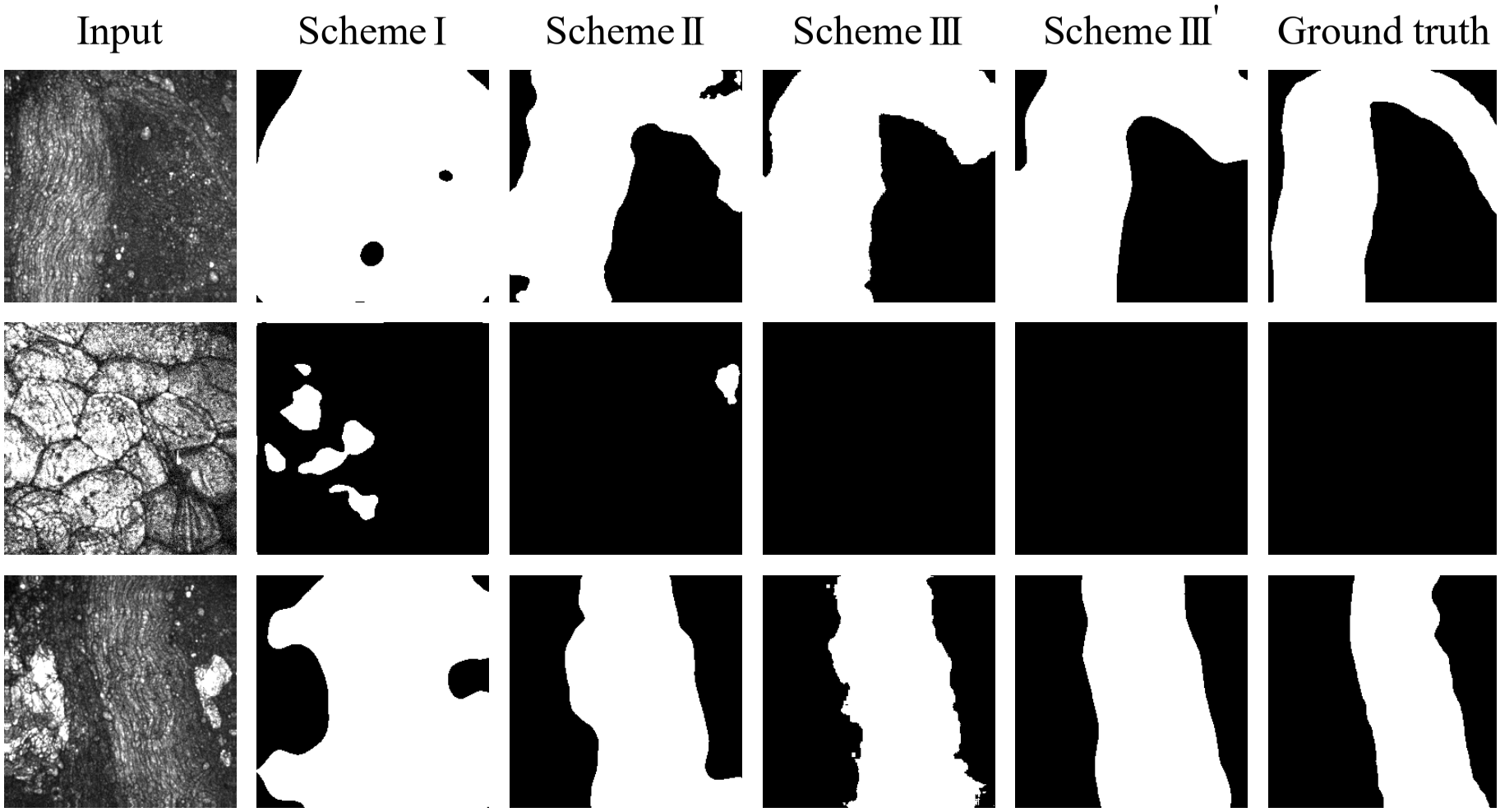


Figure 3.5: Results of nerve segmentation with four learning schemes [1].

The evaluation metrics are summarized in Table 3.1. The four evaluation metrics of the mean accuracy and F_1 value, and sensitivity, specificity, and precision are shown. Scheme II indicates a significantly higher performance than Scheme I in all evaluation metrics (Bonferroni-Holm corrected significance level: p = 0.0167 for specificity and p = 0.0250 for the other metrics). Scheme III also shows a significantly better segmentation quality than Scheme I (Bonferroni-Holm corrected significance level: p = 0.0250 for specificity and p = 0.0167 for the other metrics). These results indicate that the VGG16 encoder improves the performance of nerve extraction. The value of the mean accuracy and F_1 value in Scheme III has significantly higher values than that in Scheme II (Bonferroni-Holm corrected significance level: p = 0.05). The other metrics (sensitivity, specificity and precision) of Scheme III tended to be higher than those of Scheme II. In addition to the VGG16 encoder, the training with fluorescence images enhances the performance. Although Scheme III' has no-significant differences from Scheme III, the average and the standard deviations of all evaluation metrics for Scheme III' were better than those of Scheme III. The ensemble learning and post-processing in Scheme III' tend to improve the performance from Scheme III.

Table 3.1: Evaluation results of nerve segmentation with four learning schemes [1].

Evaluation metric	**learning scheme**				***p* value**			
	I	**II**	**III**	**III'**	**I vs II**	**I vs III**	**II vs III**	**III vs III'**
sensitivity	0.689 ±0.391	0.949 ±0.056	0.962 ±0.054	0.977 ±0.025	<0.01	<0.01	0.14	0.16
specificity	0.843 ±0.147	0.930 ±0.046	0.937 ±0.054	0.947 ±0.030	<0.01	<0.01	0.33	0.19
precision	0.469 ±0.239	0.719 ±0.123	0.752 ±0.118	0.772 ±0.061	<0.01	<0.01	0.06	0.13
mean accuracy	0.766 ±0.156	0.939 ±0.026	0.950 ±0.031	0.962 ±0.014	<0.01	<0.01	0.03	0.06
F_1 value	0.469 ±0.243	0.809 ±0.083	0.837 ±0.085	0.860 ±0.034	<0.01	<0.01	0.03	0.05

The effect of pre-training with fluorescence images was investigated. Fluorescence images offer similar images to CARS images because the fluorescence images map the lipid distribution which is also visualized in CARS images using $\mathbf{CH_2}$ symmetric stretching vibration. The author explored nerve segmentation with a small dataset of CARS endoscopic images by compensating the amount of dataset using fluorescence images. It is found that pre-training with fluorescence images made the quality of nerve segmentation higher in accuracy. Such pre-training using the other imaging modality might be useful for introducing coherent Raman imaging into medical settings. In addition to nerve visualization, label-free CARS imaging has the potential for *in situ* definitive diagnosis because the key feature of cell nuclei is visualized in a label-free manner [10–12]. Medical applications of CARS imaging will increase in the future. Of course, in medical applications, image analysis and diagnostic assistance with deep learning are expected. It is very laborious to prepare a large number of CARS images from scratch. The transfer learning from fluorescence images would be an alternative to acquiring a large amount of CARS images. A stock of fluorescent images already acquired can be used for pre-training, and it is easier to prepare fluorescence images than CARS images because fluorescence microscopy is widespread enough. Therefore, the transfer learning method from fluorescence images will ease the barriers to the adoption of CARS imaging in medical settings.

More multiple models with ensemble learning will improve the performance of nerve segmentation. The performance of the majority decision shown in Table 3.1 was calculated from seven models. In the actual implementation, all 56 models can be used for the majority decision. It takes 549 ms to conduct the majority decision of 56 models using the present PC environment. Therefore, it is expected that the performance of the majority decision from 56 models outperforms the present results. One of the other problems to be solved is the long acquisition time of CARS imaging. The waiting time to acquire CARS images may interfere with surgery. The author believes that nerve segmentation from CARS images with a low signal-to-noise ratio will bring CARS rigid endoscopes to practical use. When the waiting time decreases,

the processing time would be a problem. There are two solutions for reducing the processing time. The first solution is reducing the number of convolutional layers. This is called hyperparameter tuning. The U-Net models with various numbers of convolution layers and depths are shown in Fig. 3.6. The U-Net with depth 4 is a base model. The U-Net2 and U-Net3 have a lower number of convolutional layers than the base model. The models with depth 3 have a lower number of max pooling layers. It is important to choose several layers that balance performance and processing time.

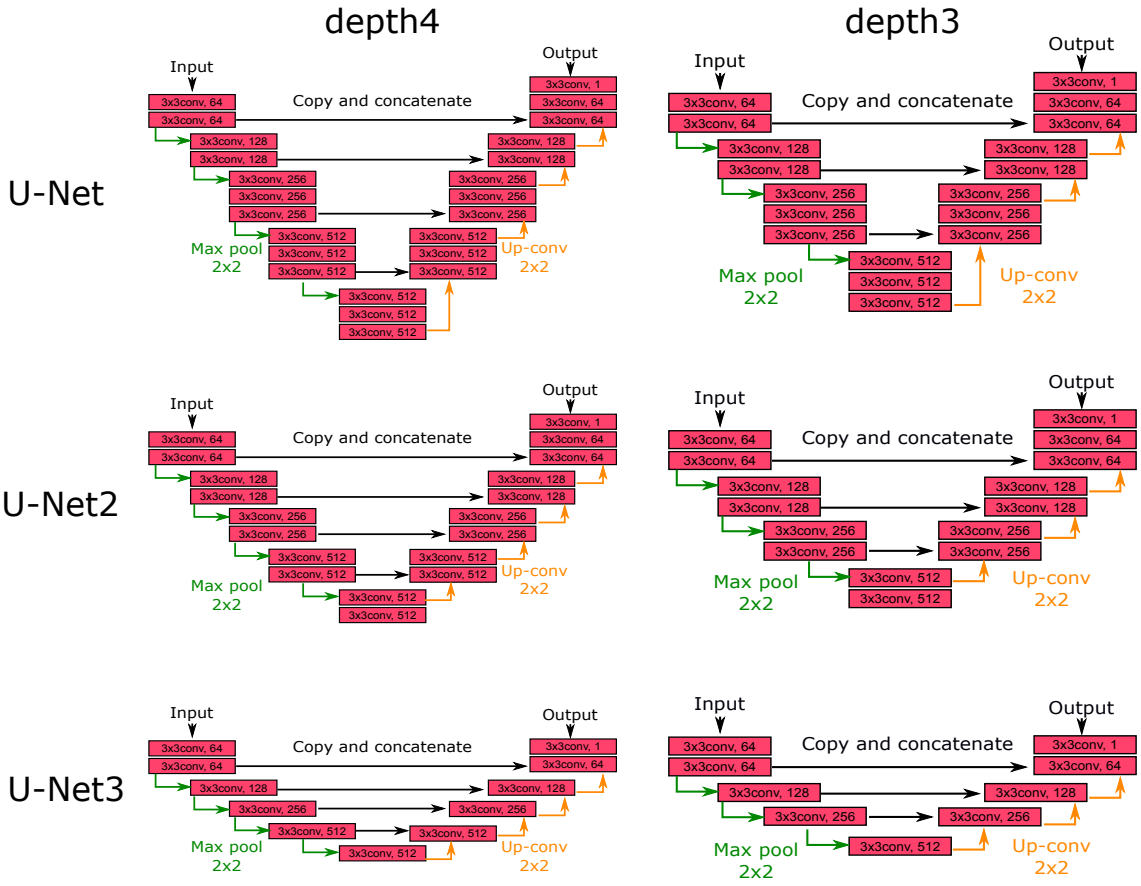


Figure 3.6: U-Net models for hyperparameter tuning. U-Net with depth 4 is a base model.

The second solution is using depthwise separable convolution layers shown in Fig. 3.7 [13]. The standard convolution is replaced by two convolutions (depthwise convolution and pointwise convolution). When the kernel size is 3, the computational costs are reduced by a factor of about 9.

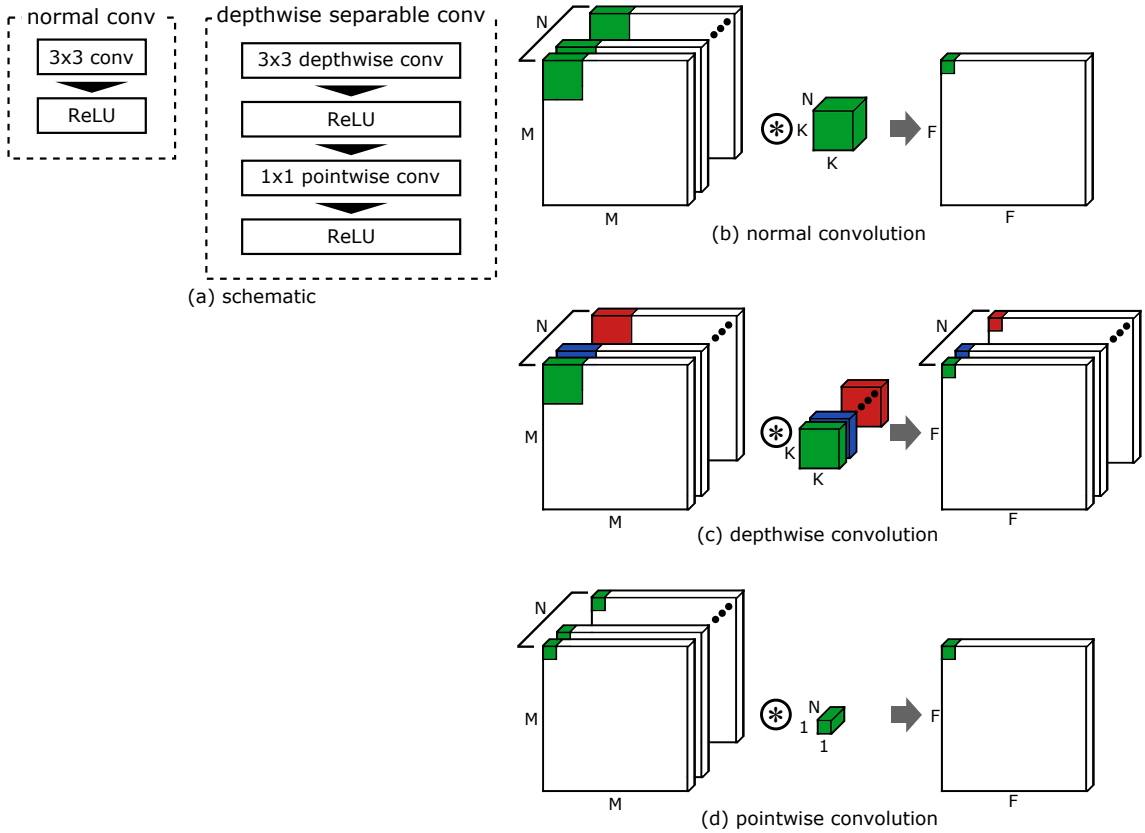


Figure 3.7: Schematic of depthwise separable convolution. F; the size of output images, K; the kernel size, M; the size of input images, N; the number of channels for input images, P; the number of channels for output images. The computational cost of standard convolution is $((K \times K \times N) \times F \times F) \times P$. The computational cost of depthwise convolution is $((K \times K) \times F \times F) \times N$. The computational cost of pointwise convolution is $((1 \times 1 \times N) \times F \times F) \times P$.

The results for hyperparameter tuning with the depthwise separable convolution using fluorescence images are summarized in Table 3.2. The average of F_1 scores are shown. The values in the parentheses after the F_1 scores are the ratio of processing time when U-Net with the standard convolution is set to 1. In comparison between U-Net with standard convolution with BN and U-Net with depthwise separable convolution, the results show that the processing time was reduced while performance was maintained.

The combination of hyperparameter tuning and depthwise convolution reduced the processing time by about one-third.

Table 3.2: The result for hyperparameter tuning with depthwise separable convolutions using fluorescence images.

model	U-Net	U-Net2	U-Net3
standard convolution with BN	0.853 (1.00)	0.846 (0.85)	0.835 (0.53)
depthwise separable convolution	0.858 (0.54)	0.851 (0.50)	0.844 (0.35)
standard convolution with BN (depth 3)	0.834 (0.77)	0.837 (0.67)	0.819 (0.43)
depthwise separable convolution (depth 3)	0.838 (0.48)	0.827 (0.45)	0.809 (0.31)

3.3 Conclusion

The nerve segmentation of CARS endoscopic images using the deep learning model was demonstrated. The usage of the VGG16 encoder part which was trained with the ImageNet dataset improved the nerve segmentation performance. The enhanced performance of nerve segmentation by transfer learning is similar to a previous report [6]. further improved the performance. The high performance of nerve segmentation (a mean accuracy of 0.962 and an F_1 value of 0.860) was achieved. A tendency for the output images to have larger nerve areas than ground truth images was confirmed. In terms of nerve preservation, this tendency seems to be preferable to underestimating the nerve areas.

Deep learning achieved remarkably high performance of automated nerve segmentation from label-free CARS rigid endoscopic images. CARS images have not only nerves but also other tissues distribution because $\mathbf{CH}_2$ symmetric stretching vibration is included in other tissues. The evaluation metrics (a mean accuracy of 0.962 and an F_1 value of 0.860) indicate the deep learning model trained with a few CARS images allowed for predicting correctly nerve areas. The author believes that CARS imaging with automated nerve segmentation will contribute to intraoperative nerve-sparing.

Reference

[1] N. Yamato, M. Matsuya, H. Niioka, J. Miyake, and M. Hashimoto, "Nerve Segmentation with Deep Learning from Label-Free Endoscopic Images Obtained Using Coherent Anti-Stokes Raman Scattering," *Biomolecules*, vol. 10, p. 1012, July 2020.

[2] K. Hirose, T. Aoki, T. Furukawa, S. Fukushima, H. Niioka, S. Deguchi, and M. Hashimoto, "Coherent anti-Stokes Raman scattering rigid endoscope toward robot-assisted surgery," *Biomedical optics express*, vol. 9, no. 2, pp. 387–396, 2018.

[3] K. Hirose, S. Fukushima, T. Furukawa, H. Niioka, and M. Hashimoto, "Invited Article: Label-free nerve imaging with a coherent anti-Stokes Raman scattering rigid endoscope using two optical fibers for laser delivery," *APL Photonics*, vol. 3, p. 092407, Sept. 2018.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Springer International Publishing, 2015.

[5] T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, A. Dovzhenko, O. Tietz, C. Dal Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Di-

ester, T. Brox, and O. Ronneberger, "U-Net: deep learning for cell counting, detection, and morphometry," *Nature methods*, vol. 16, pp. 67–70, Jan. 2019.

[6] C. Balakrishna, S. Dadashzadeh, and S. Soltaninejad, "Automatic detection of lumen and media in the IVUS images using U-Net with VGG16 Encoder," June 2018.

[7] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *the IEEE international conference on computer vision (ICCV)*, pp. 1026–1034, 2015.

[9] S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection," *BMC bioinformatics*, vol. 7, p. 91, Feb. 2006.

[10] Y. Yang, F. Li, L. Gao, Z. Wang, M. J. Thrall, S. S. Shen, K. K. Wong, and S. T. C. Wong, "Differential diagnosis of breast cancer using quantitative, label-free and molecular vibrational imaging," *Biomedical optics express*, vol. 2, pp. 2160–2174, Aug. 2011.

[11] L. Gao, Z. Wang, F. Li, A. A. Hammoudi, M. J. Thrall, P. T. Cagle, and S. T. C. Wong, "Differential diagnosis of lung carcinoma with coherent anti-Stokes Raman scattering imaging," *Archives of pathology & laboratory medicine*, vol. 136, pp. 1502–1510, Dec. 2012.

[12] S. Weng, X. Xu, J. Li, and S. T. C. Wong, "Combining deep learning and coherent anti-Stokes Raman scattering imaging for automated differential diagnosis of lung cancer," *Journal of biomedical optics*, vol. 22, no. 10, p. 106017, 2017.

[13] A. G. Howard, M. Zhu, B. Chen, D. Kelenichenko, W. Wang, and others, "Efficient convolutional neural networks for mobile vision.".

Chapter 4

Accelerating CARS endoscopic imaging rate using nerve segmentation

4.1 Introduction

. In this chapter, the author investigates the acceleration of the nerve imaging rate using segmentation. In general, semantic segmentation is used to label multiple objects in images with pixel-pixel level for a wide range of applications [1]. In the previous chapter, nerve segmentation is demonstrated for CARS endoscopic images obtained at a low imaging rate. The demonstration of nerve segmentation from CARS images with a considerably low signal-to-noise ratio obtained at a high imaging rate means the acceleration of the imaging rate. For segmentation, the author evaluates the acceleration of the imaging rate using the *CIR* with F_1 value. This chapter is based on the following publication:

This chapter is based on the following publication:
Naoki Yamato, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto Naoki Yamato, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto, "Near real-time nerve visualization using coherent Raman scattering rigid endoscope and deep learning-based image processing for nerve-sparing surgery," Proc. SPIE, Biomedical Vibrational Spectroscopy 2022: Advances in Research and Industry, 11957, 119570B, (2022).

Naoki Yamato, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto, "Near real-time nerve visualization using coherent Raman scattering rigid endoscope and deep learning-based image processing for nerve-sparing surgery," Proc. SPIE, Biomedical Vibrational Spectroscopy 2022: Advances in Research and Industry, 11957, 119570B, (2022).

4.2 Training of nerve segmentation from images with low signal-to-noise ratio

4.2.1 CARS imaging

Additional CARS images were obtained using the same setup in chapter 3. To evaluate the improvement of the imaging rate, 100 nerve images at an imaging rate of 0.625 frames per second were obtained at each position. As a result, 3600 images at 36 positions were acquired. The CARS images were divided into 6 folds (each fold has 600 images at 6 positions).

4.2.2 Training of nerve segmentation

The U-Net architecture [2,3] was used for training nerve segmentation. As in chapter 2, the weight of a VGG16 encoder was used as the encoder part of U-Net [4]. The batch normalization layers were applied to the convlution layers except for the VGG16 encoder. The threshold of 0.5 was applied to binarize the predicted images. The loss function is provided by Eq. 3.1. The Adam optimizer [5] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a learning rate of 0.001 and mini-batch number of 32 were used. The He's initialization method [6] was utilized to initialize the parameters of convolution layers. The maximum number of epochs was set as 200. The early stopping with the number 20 was also used. The data augmentation of shuffle, random cropping, random flip (horizontal and vertical) and rotation (90 and 180 degrees) was used to prevent overfitting. The size of cropping was 192×192 pixels (the original size of the images is 256×256 pixels). By random cropping, the author cropped 16 images per image for training fluorescence images and 16 images per image for training CARS images, respectively.

The fluorescence images were used as pre-training. A total of 1818 fluorescence images were divided into 10 folds. The performance of nerve segmentation was evaluated using 10-fold cross-validation. In 10-fold cross-validation, 9 times of training

were conducted (different validation set was used at each training). The author evaluated the performance by F_1 value. The model showing the highest F_1 value was used for transfer learning of CARS images.

For training CARS images, 6-fold nested cross-validation was used. The author compared 5 training schemes using the averaged evaluation metrics over a total of 30 models. Scheme I means that a U-Net model was trained with CARS images only from scratch. Scheme II means that a U-Net model was pre-trained with fluorescence images and fine-tuned with CARS images. Scheme III means that a U-Net with a VGG16 encoder was pre-trained with or without fluorescence images and fine-tuned with CARS images. It is different to use CARS images with a low signal-to-noise ratio (SNR). In training CARS images, the accumulation number of input images was randomly selected from 100 (exposure time; 160 s) to 1 (exposure time; 1.6 s). It is called random averaging in this chapter. And the frames used for random averaging were randomly extracted from 100 frames. It is called a random frame in this chapter.

4.3 Evaluation of the improvement of imaging rate

The result of nerve segmentation for each scheme is shown in Fig. 4.1 and Table 4.1. The horizontal and vertical axes mean the exposure time and the averaged F_1 values, respectively. The standard deviation is not shown in Fig. 4.1 and is summarized in Table 4.1 because the visibility becomes lowered. Scheme I is depicted as UNetBN_fromScratch (blue plots) in Fig. 4.1. This result indicates that the nerve segmentation for low SNR images has a sufficient performance over F_1 value of 0.8. Scheme II is depicted as UNetBN_fluo_alltuning and UNetBN_fluo_fixhalf (orange and green plots) in Fig. 4.1. The "alltuning" means that all weights of convolution layers were updated in training CARS images. On the other hand, the "fixhalf" means that weights in the first half of the convolution layers of U-Net were fixed and those in the second half of the convolution layers were updated in training CARS images. Scheme II shows a significantly higher performance than Scheme I

for all exposure times. It is found that the pre-training using fluorescence images is effective to improve the performance of nerve segmentation for high SNR images as well as low SNR images. Scheme III is depicted as VGGUNetBN_fixhalf and VGGUNetBN_fixhalf_fluo_alltuning (red and purple plots) in Fig. 4.1. The U-Net with VGG encoder shows lower performance than Scheme II (UNetBN_fluo_alltuning). This trend is considered due to the difference between datasets in training VGG and nerve segmentation. VGG16 is trained for classification tasks using an ImageNet dataset with high SNR. The low SNR images are not included in this dataset. Therefore, it is considered that the VGG16 encoder does not work well for low SNR images. As a result, Scheme II using fluorescence and CARS images show the highest performance for low SNR images. For nerve segmentation, $\boldsymbol{CIR}_{F1}$ is defined as the fastest imaging rate satisfying $\boldsymbol{F}_1 \geq 0.8$. This threshold is a criterion required for medical images [7–10]. The $\boldsymbol{F}_1$ value for Scheme II already got over the criterion of 0.8 at the shortest exposure time. The $\boldsymbol{CIR}_{F1}$ is calculated as 0.625 fps from the shortest exposure time of 1.6 s.

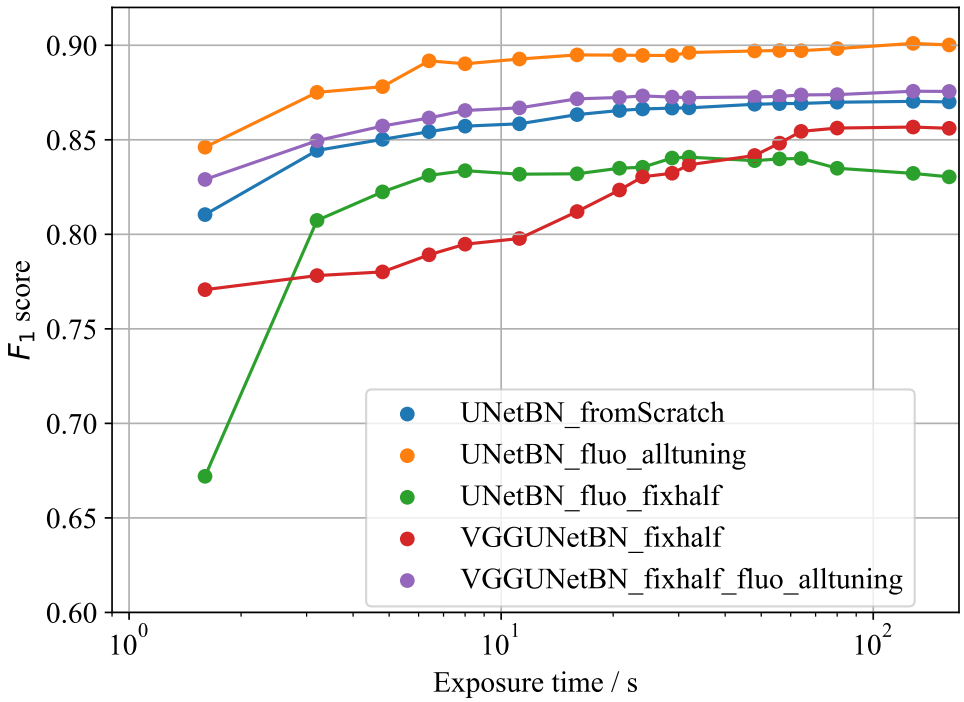


Figure 4.1: Results of nerve segmentation with three learning schemes. Scheme I is depicted as UNetBN_fromScratch (blue plots). Scheme II is depicted as UNetBN_fluo_alltuning and UNetBN_fluo_fixhalf (orange and green plots). Scheme III is depicted as VGGUNetBN_fixhalf and VGGUNetBN_fixhalf_fluo_alltuning (red and purple plots). The averaged F_1 values are plotted (the standard deviations are not shown because the visibility becomes lowered). The standard deviations are summarized in Table 4.1.

Table 4.1: Summary of the evaluation metric of F_1 values. The average and standard deviation of the metric are shown in the upper row and lower row for each scheme. Each average value was calculated over 6 tests (each test has 5 models with different validation datasets).

exposure time [s]	1.6	3.2	4.8	6.4	16	32	48	64	128	160
UNetBN fromScratch	0.810	0.844	0.850	0.854	0.863	0.867	0.869	0.869	0.870	0.870
	±0.149	±0.121	±0.119	±0.117	±0.115	±0.116	±0.116	±0.117	±0.116	±0.116
UNetBN fluo all	0.846	0.875	0.878	0.892	0.895	0.896	0.897	0.897	0.901	0.900
	±0.127	±0.097	±0.094	±0.066	±0.073	±0.082	±0.082	±0.084	±0.073	±0.073
UNetBN fluo fix	0.672	0.807	0.822	0.831	0.832	0.841	0.839	0.840	0.832	0.830
	±0.223	±0.133	±0.119	±0.111	±0.115	±0.116	±0.117	±0.122	±0.124	±0.123
VGGUNetBN fix	0.771	0.778	0.780	0.789	0.812	0.837	0.842	0.854	0.857	0.856
	±0.121	±0.112	±0.117	±0.119	±0.122	±0.104	±0.105	±0.087	±0.089	±0.091
VGGUNetBN fluo all	0.829	0.850	0.857	0.862	0.872	0.872	0.873	0.874	0.876	0.876
	±0.137	±0.124	±0.123	±0.118	±0.108	±0.113	±0.115	±0.115	±0.112	±0.110

The segmentation results are shown in Fig. 4.2 and 4.3. Figure 4.2 and 4.3 show the predicted images with low and high F_1 values, respectively. The ground truth images are in the lowest row. The exposure time of the input CARS images increases from the left column to the right column. You can see that the nerves in input images become clearer in proportion to the exposure time in Fig. 4.2 and 4.3. The models using the fixed weights of the encoder part in UNetBN_fluo_fixhalf and VGGUNetBN_fixhalf show the deteriorated predicted images for all exposure times. In Fig. 4.3, better segmentation performance was obtained, regardless of the learning schemes.

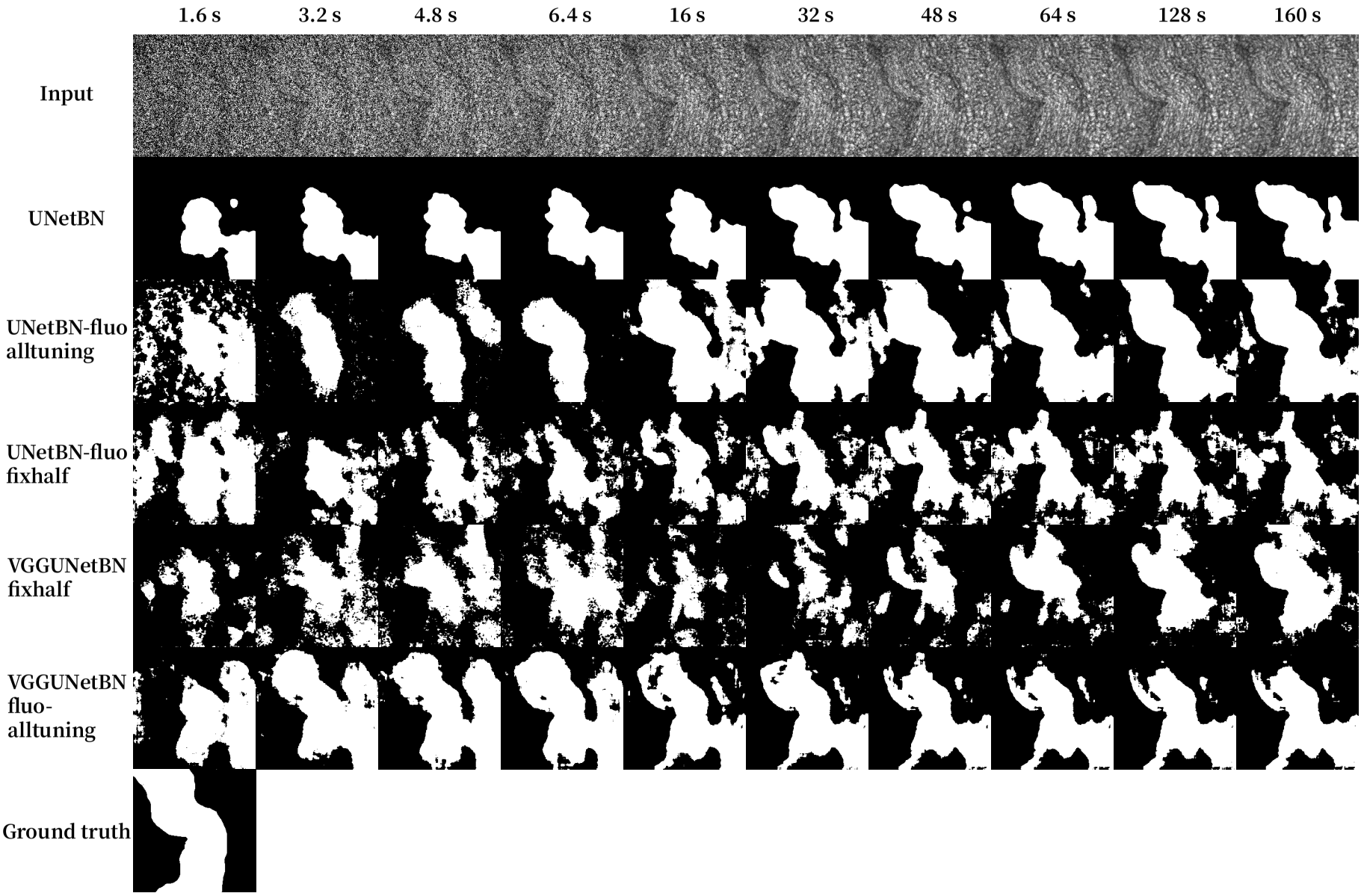


Figure 4.2: The segmentation result with lower F_1 values.

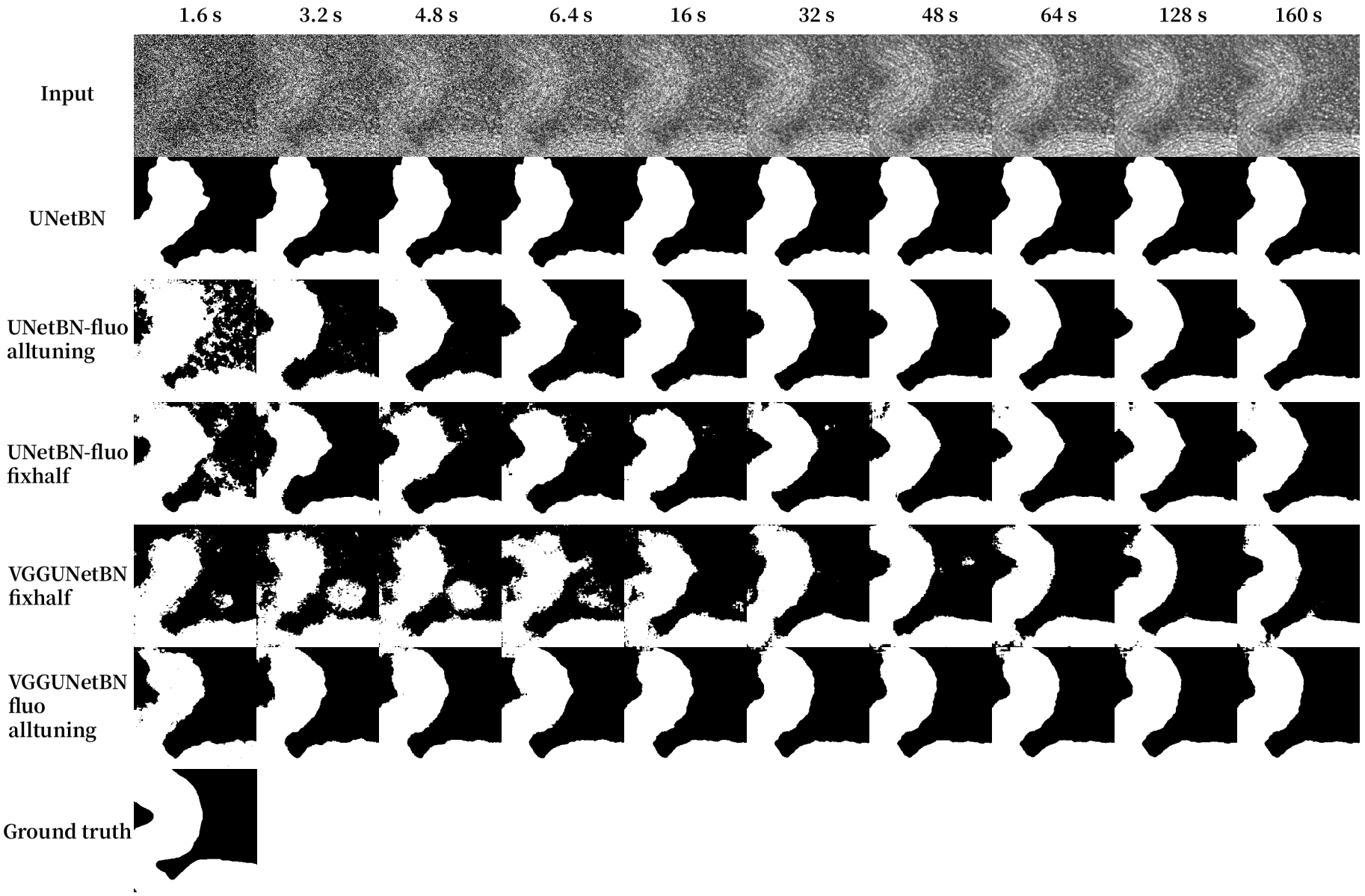


Figure 4.3: The segmentation result with higher F_1 values.

To investigate the dependence on the number of CARS images, the models trained with 24, 12, and 6 images were prepared. These results are summarized in Fig. 4.4 and Table 4.2, 4.3, and 4.4. The author compared the average F_1 values for the same three test sets (each test set has 2 validation sets). Scheme I which shows the highest performance in Fig. 4.1, also has better performance in all numbers of training in Fig. 4.4(a)–(c). On the other hand, Scheme III (VGGUNetBN_fluo_fixhalf_alltuning) depends on the number of training datasets.

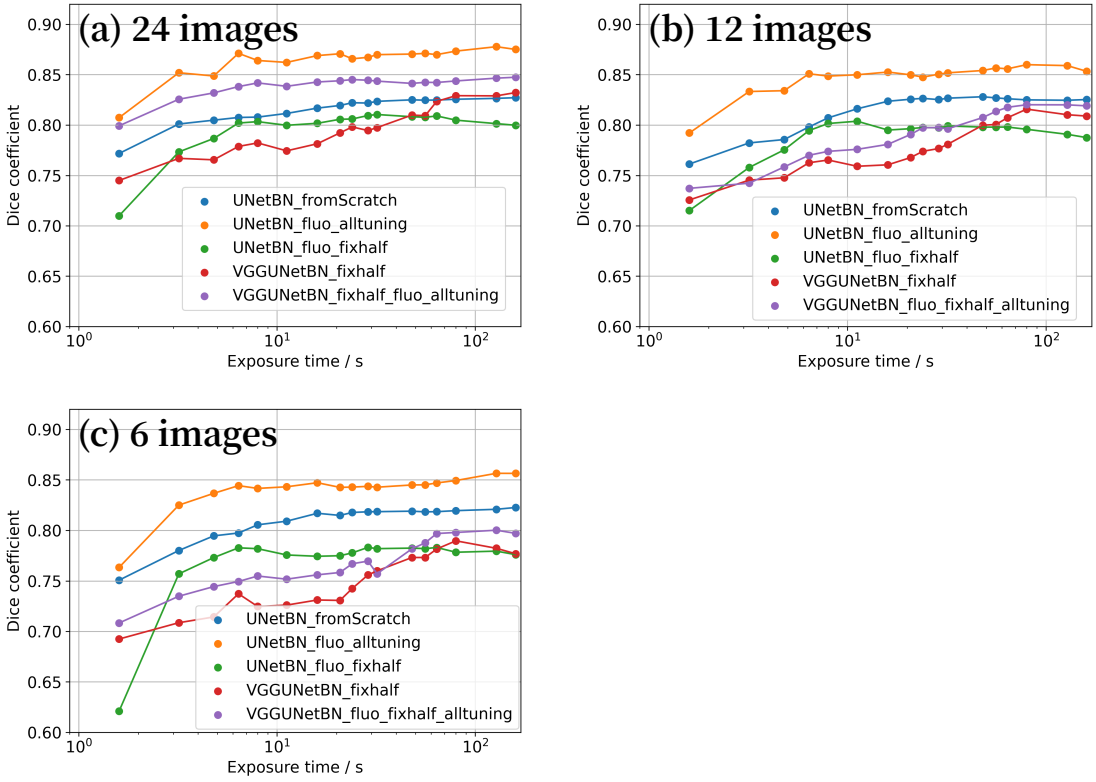


Figure 4.4: Results of nerve segmentation with various training sets. (a) the result for training 24 images. (b) the result for training 12 images. (c) the result for training 6 images.

Table 4.2: Summary of the evaluation metric of F_1 values for training 24 images. The average and standard deviation of the metric are shown in the upper row and lower row for each scheme. Each average value was calculated over 3 tests (each test has 2 models with different validation datasets).

exposure time [s]	1.6	3.2	4.8	6.4	16	32	48	64	128	160
UNetBN fromScratch	0.772	0.801	0.805	0.808	0.817	0.824	0.825	0.825	0.826	0.827
	±0.034	±0.043	±0.047	±0.042	±0.036	±0.034	±0.035	±0.035	±0.036	±0.036
UNetBN fluo all	0.816	0.852	0.859	0.871	0.876	0.873	0.873	0.874	0.876	0.872
	±0.063	±0.052	±0.048	±0.033	±0.017	±0.019	±0.022	±0.028	±0.027	±0.030
UNetBN fluo fix	0.710	0.773	0.787	0.802	0.802	0.810	0.808	0.809	0.801	0.800
	±0.057	±0.046	±0.050	±0.047	±0.033	±0.026	±0.024	±0.020	±0.026	±0.027
VGGUNetBN fix	0.745	0.767	0.765	0.779	0.781	0.797	0.810	0.823	0.829	0.832
	±0.033	±0.039	±0.024	±0.028	±0.039	±0.030	±0.036	±0.036	±0.038	±0.039
VGGUNetBN fluo all	0.799	0.826	0.832	0.838	0.843	0.844	0.841	0.842	0.847	0.847
	±0.023	±0.020	±0.022	±0.016	±0.017	±0.021	±0.023	±0.022	±0.021	±0.021

Table 4.3: Summary of the evaluation metric of F_1 values for training 12 images. The average and standard deviation of the metric are shown in the upper row and lower row for each scheme. Each average value was calculated over 3 tests (each test has 2 models with different validation datasets).

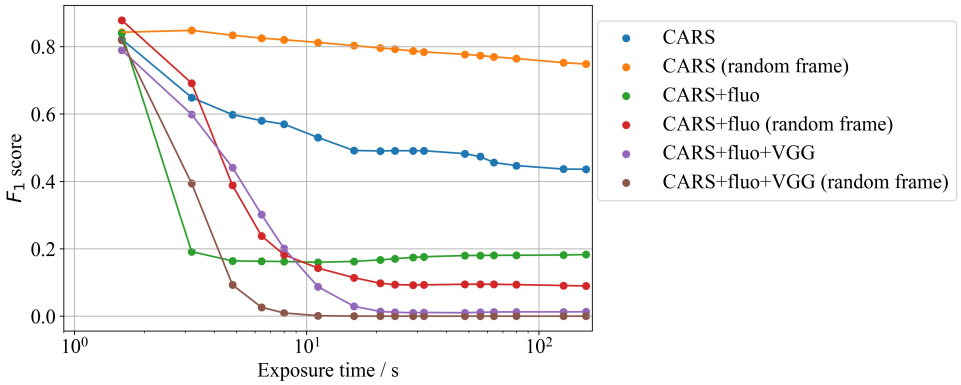
exposure time [s]	1.6	3.2	4.8	6.4	16	32	48	64	128	160
UNetBN fromScratch	0.761	0.782	0.786	0.798	0.824	0.827	0.828	0.826	0.825	0.825
	±0.050	±0.055	±0.060	±0.053	±0.040	±0.037	±0.038	±0.038	±0.039	±0.039
UNetBN fluo all	0.792	0.833	0.834	0.851	0.853	0.852	0.854	0.856	0.859	0.853
	±0.058	±0.040	±0.038	±0.037	±0.032	±0.028	±0.028	±0.027	±0.040	±0.047
UNetBN fluo fix	0.715	0.758	0.775	0.795	0.795	0.799	0.798	0.798	0.791	0.788
	±0.035	±0.048	±0.050	±0.049	±0.042	±0.032	±0.031	±0.031	±0.035	±0.037
VGGUNetBN fix	0.726	0.745	0.748	0.763	0.761	0.781	0.800	0.807	0.810	0.809
	±0.060	±0.046	±0.028	±0.035	±0.041	±0.041	±0.039	±0.041	±0.030	±0.037
VGGUNetBN fluo all	0.737	0.742	0.758	0.770	0.781	0.796	0.807	0.818	0.820	0.819
	±0.039	±0.039	±0.028	±0.029	±0.038	±0.040	±0.040	±0.039	±0.033	±0.041

Table 4.4: Summary of the evaluation metric of F_1 values for training 6 images. The average and standard deviation of the metric are shown in the upper row and lower row for each scheme. Each average value was calculated over 3 tests (each test has 2 models with different validation datasets).

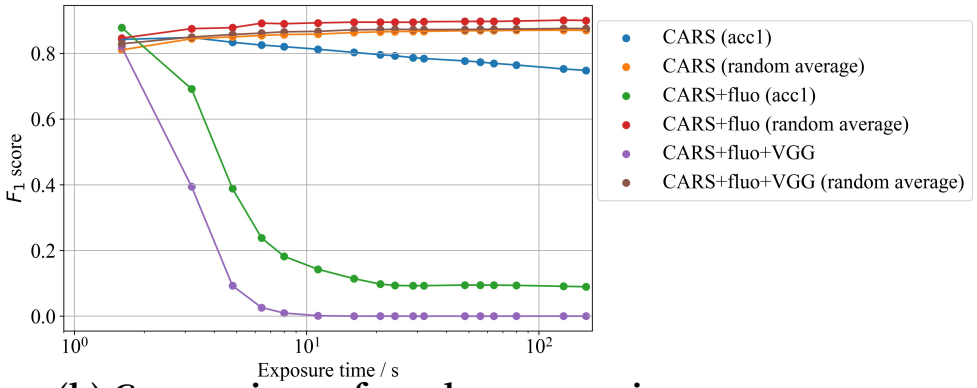
exposure time [s]	1.6	3.2	4.8	6.4	16	32	48	64	128	160
UNetBN fromScratch	0.751	0.780	0.795	0.797	0.817	0.819	0.819	0.819	0.821	0.823
	±0.208	±0.184	±0.169	±0.167	±0.145	±0.147	±0.150	±0.151	±0.147	±0.141
UNetBN fluo all	0.763	0.825	0.837	0.844	0.847	0.843	0.845	0.847	0.857	0.856
	±0.197	±0.138	±0.132	±0.124	±0.112	±0.126	±0.122	±0.123	±0.111	±0.111
UNetBN fluo fix	0.621	0.757	0.773	0.783	0.774	0.782	0.782	0.783	0.779	0.776
	±0.225	±0.155	±0.138	±0.136	±0.155	±0.157	±0.155	±0.158	±0.159	±0.159
VGGUNetBN fix	0.692	0.709	0.714	0.737	0.731	0.760	0.773	0.782	0.782	0.777
	±0.154	±0.131	±0.116	±0.112	±0.124	±0.099	±0.095	±0.087	±0.103	±0.112
VGGUNetBN fluo all	0.708	0.735	0.744	0.749	0.756	0.757	0.782	0.797	0.800	0.797
	±0.179	±0.131	±0.125	±0.140	±0.140	±0.139	±0.123	±0.107	±0.111	±0.115

The nerve segmentation has the potential to accelerate the imaging rate. The Scheme II using fluorescence images as pre-training (UNetBN_fluo_alltuning) in Fig. 4.1 indicates that the exposure time (1.6 s) of CARS images is sufficient to conduct nerve segmentation. The margin between scheme I ("UNetBN_fromScratch") and scheme II ("VUNetBN_fluo_alltuning") indicates the possibility of more acceleration for the exposure time (less than 1.6 s).

The effect of data augmentation using random frame and random averaging on nerve segmentation performance is investigated. The results of training using CARS images (only accumulation number 1) with or without random frames are shown in Fig. 4.5(a). When using only CARS images (labeled as "CARS" and "CARS (random frame)"), random frame boosts the nerve segmentation performance except for the shortest exposure time. In this training, the model learns from CARS images for only the shortest exposure time. The augmentation of random frames means using images with a slightly different noise distribution in the same imaging window. The use of many noise distribution patterns would boost the generalization of segmentation models. The results of training using CARS images without (only accumulation number 1) or with random averaging (accumulation number from 1 to 100) are shown in Fig. 4.5(b). Random averaging also improves the nerve segmentation performance, as does the random frame. For example, when using only CARS images without (blue) or with (orange) random averaging, the nerve extraction performance for CARS images obtained for a long exposure time increases. This trend is observed for the model "CARS+fluo" and "CARS+fluo+VGG". These results indicate that it is useful to use the augmentation of random frames and random averaging for small datasets.



(a) Comparison of random frame



(b) Comparison of random averaging

Figure 4.5: Comparison between training using different datasets. (a) comparing training with or without random frames. (b) comparing training with or without random averaging.

4.4 Comparison of imaging acceleration between denoise and segmentation

The author evaluated the improvement of imaging speed by noise reduction for the same dataset. $CIR_{segmentation}$ and $CIR_{denoise}$ were used to compare imaging rates. The $CIR_{denoise}$ was calculated from the same procedure in chapter 2. The exposure times of CARS images used for training were 1.6, 3.2, 4.8, 6.4, 8.0, 16, 24, 32, 48, 56, 64, 80, and 128 s. The averaged evaluation metrics are plotted in Fig. 4.6(a). The evaluation metrics for raw images and denoised images are plotted as blue and orange dots, respectively. The quality of denoised images increases for a short exposure time. The criteria of PSNR = 30 and SSIM = 0.8 are the red dashed lines in each plot. The CIR_{PSNR} and CIR_{SSIM} were calculated from the cross points between the fitting curves and the criterion. The CIR for raw images and denoised images is 0.68 images/min and 0.79 images/min, respectively. The CIR_{PSNR} and CIR_{SSIM} for raw images are 0.54 images/min and 0.92 images/min. The CIR_{PSNR} and CIR_{SSIM} for denoised images are 0.57 images/min and 1.28 images/min. The imaging speed becomes 1.2 times faster using denoising. The denoised CARS images are shown in Fig. 4.6(b). The fiber-like shape indicates a peripheral nerve. The denoised image for an exposure time of 1.6 s is more blurred compared with an exposure time of 48 s. The images satisfying SSIM=0.8 (the raw image obtained for the exposure time of 56 s and the denoised image obtained for the exposure time of 48 s in Fig. 4.6(b)) are indistinguishable from the GT image.

The degree of imaging rate acceleration is lower than in the previous report [11]. The author considers that the signal-to-noise ratio of input images is different. The averaged scores of PSNR for raw images at an exposure time of 1.6 s are 6.25 dB and 14.94 dB for new CARS images and the previous paper, respectively. More exposure time would be required to achieve sufficient signal-to-noise of CARS images. The author considers that the laser source' s pulsing condition was not much better than the previous report, which degrades the scattering efficiency of CARS. The power of

the excitation laser sources was similar to the previous report [11].

The improvement of the imaging rate using segmentation gets 47.5 times higher than using denoising comparing $CIR_{segmentation}$ and $CIR_{denoise}$. This result indicates nerve segmentation is more robust to the SN ratio of the input images.

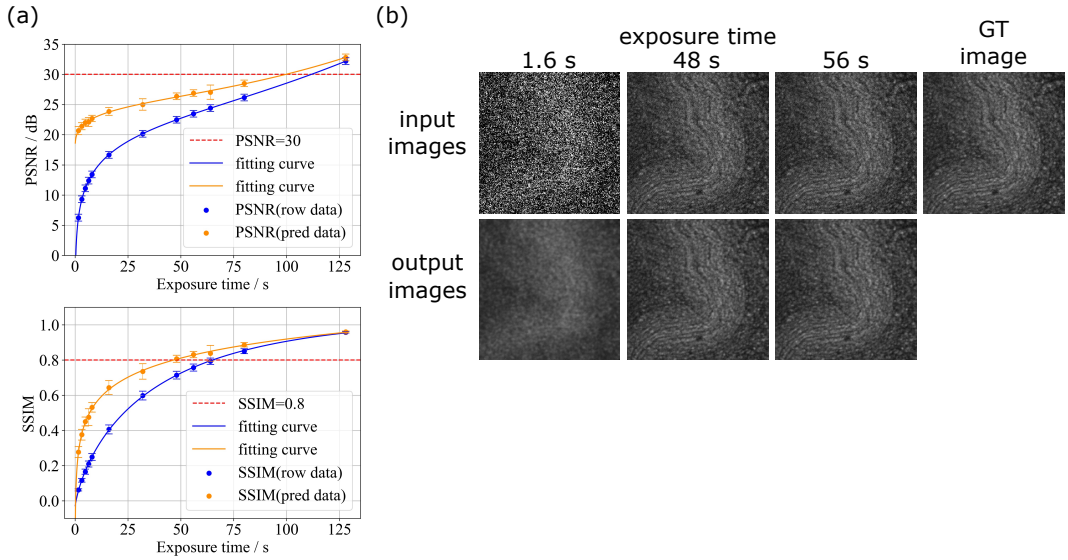


Figure 4.6: Denoising results of nerve images obtained with the CARS endoscopy [12]. (a) The averaged PSNR and SSIM for each exposure time. The points and error bars show the average and standard deviation of the evaluation metric, respectively. The blue and orange plots indicate the row image without denoising and the processed image with denoising. The red dashed lines are the criteria required for medical images (PSNR=30, SSIM=0.8). (b) Denoised CARS images. GT: ground truth, PSNR; peak signal-to-noise ratio, SSIM: structural similarity.

4.5 Generalization performance for the images without nerve tissues

The author evaluated the generalization performance of nerve segmentation using CARS images with non-nervous tissues. Five CARS images, including adipose tissues, were used. The averaged F_1 scores are summarized in Table 4.5. This result shows the higher generalization performance for non-nervous tissues.

Table 4.5: The result for hyperparameter tuning with depthwise separable convolutions using fluorescence images.

model	1.6 s	320 s
UNetBN fromScrach	0.994±0.007	0.993±0.008
UNetBN fluo alltuning	0.989±0.017	0.991±0.014
VGGUNetBN fixhalf fluo alltuning	0.989±0.013	0.990±0.010

4.6 Conclusion

The author applied nerve segmentation to CARS images with a low SNR. As in chapter 2, pre-training using fluorescence images enhanced the performance significantly. Unlike chapter 2, the usage of the VGG16 encoder did not contribute to the performance improvement in nerve segmentation. It is considered that a VGG16 encoder did not work well because the dataset of VGG16 has an exceedingly different SNR from the CARS images obtained at a high imaging rate. As a result, the nerve segmentation using the U-Net model trained with fluorescence images and CARS images shows the sufficient performance of F_1 value > 0.8 for CARS images with a low SNR. The segmentation boosted the imaging rate by a factor of about 5 times than denoising. Therefore, image processing with deep learning enables acceleration by a factor of 25 than row CARS endoscopic imaging.

Reference

[1] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, pp. 3523–3542, July 2022.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Springer International Publishing, 2015.

[3] T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald, A. Dovzhenko, O. Tietz, C. Dal Bosco, S. Walsh, D. Saltukoglu, T. L. Tay, M. Prinz, K. Palme, M. Simons, I. Diester, T. Brox, and O. Ronneberger, "U-Net: deep learning for cell counting, detection, and morphometry," *Nature methods*, vol. 16, pp. 67–70, Jan. 2019.

[4] C. Balakrishna, S. Dadashzadeh, and S. Soltaninejad, "Automatic detection of lumen and media in the IVUS images using U-Net with VGG16 Encoder," June 2018.

[5] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *the IEEE international conference on computer vision (ICCV)*, pp. 1026–1034, 2015.

[7] S. Pudlewski and T. Melodia, "Compressive Video Streaming: Design and Rate-Energy-Distortion Analysis," *IEEE Transactions on Multimedia*, vol. 15, pp. 2072–2086, Dec. 2013.

[8] K. K. Brock, S. Mutic, T. R. McNutt, H. Li, and M. L. Kessler, "Use of image registration and fusion algorithms and techniques in radiotherapy: Report of the AAPM Radiation Therapy Committee Task Group No. 132," *Medical physics*, vol. 44, pp. e43–e76, July 2017.

[9] K. Dolde, C. Dávid, G. Echner, R. Floca, C. Hentschke, F. Maier, N. Niebuhr, K. Ohmstedt, N. Saito, M. Alimusaj, B. Fluegel, P. Naumann, C. Dreher, M. Freitag, and A. Pfaffenberger, "4DMRI-based analysis of inter— and intrafractional pancreas motion and deformation with different immobilization devices," *Biomedical Physics & Engineering Express*, vol. 5, no. 2, p. 025012, 2019.

[10] Y. Zhang, E. Paulson, S. Lim, W. A. Hall, E. Ahunbay, N. J. Mickevicius, M. W. Straza, B. Erickson, and X. A. Li, "A Patient-Specific Autosegmentation Strategy Using Multi-Input Deformable Image Registration for Magnetic Resonance Imaging–Guided Online Adaptive Radiation Therapy: A Feasibility Study," *Advances in radiation oncology*, vol. 5, no. 6, pp. 1350–1358, 2020.

[11] N. Yamato, H. Niioka, J. Miyake, and M. Hashimoto, "Improvement of nerve imaging speed with coherent anti-Stokes Raman scattering rigid endoscope using deep-learning noise reduction," *Scientific reports*, vol. 10, p. 15212, Sept. 2020.

[12] N. Yamato, H. Niioka, J. Miyake, and M. Hashimoto, "Near real-time nerve visualization using coherent Raman scattering rigid endoscope and deep learning-based image processing for nerve-sparing surgery," in *Biomedical Vibrational Spectroscopy 2022: Advances in Research and Industry*, vol. 11957, pp. 63–68, SPIE, Mar. 2022.

Conclusion

The author demonstrated that denoising using deep learning improves the imaging rate of CARS rigid endoscope. Three denoising models were compared using CARS microscopic images to select the optimal model for denoising of nerve images. N2N shows the highest performance quantitatively and qualitatively. It is found that N2N fine-tuned with CARS endoscopic images restored the detailed nerve structure faithfully. The author proposed the critical imaging rate (*CIR*) to assess the improvement of the imaging rate. According to *CIR*, the imaging rate was improved by a factor of 5 times from 1.4 images/min to 7.0 images/min.

The nerve segmentation from CARS endoscopic images with deep learning was demonstrated. The VGG16 encoder improved the nerve segmentation performance. The enhanced performance of nerve segmentation by transfer learning is similar to a previous report [**?**]. A mean accuracy of 0.962 and an F_1 value of 0.860 mean the achievement of high performance. A trend for the predicted images to have larger nerve areas than ground truth images was confirmed. In terms of nerve preservation, this tendency seems to be preferable to underestimating the nerve areas.

The author applied nerve segmentation to CARS images with a low SNR. As in chapter 3, pre-training using fluorescence images enhanced the performance significantly. Unlike chapter 3, the usage of the VGG16 encoder did not contribute to the performance improvement in nerve segmentation. It is considered that a VGG16 encoder did not work well because the dataset of VGG16 has an exceedingly different SNR from the CARS images obtained at a high imaging rate. As a result, the nerve segmentation using the U-Net model trained with fluorescence images and CARS

images shows the sufficient performance of F_1 value > 0.8 for CARS images with a low SNR. Therefore, image processing with deep learning enables acceleration by a factor of 47.5 than raw CARS endoscopic imaging. The trade-off between FOV and frame rate would be overcome using deep learning shown in Fig. 4.7.

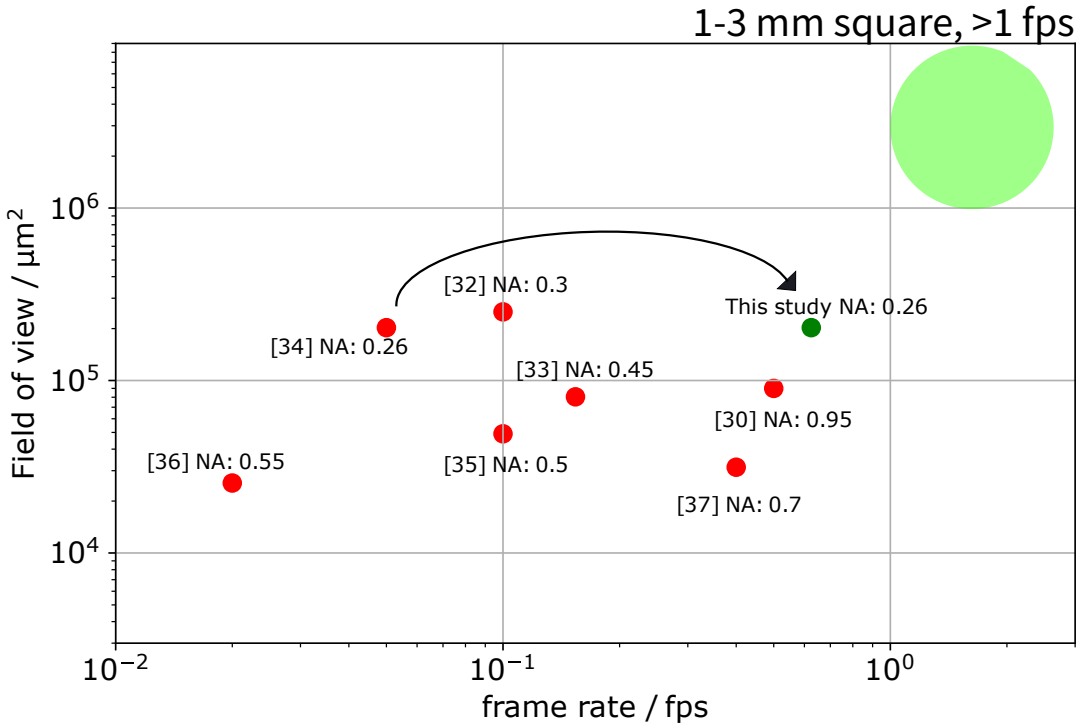


Figure 4.7: Benchmark of coherent anti-Stokes Raman scattering endoscopy. The NA is shown on the right side of the citation number. The graph indicates the trade-off relationship between FOV and fps. As the NA increases, the fps increase because CARS is induced more efficiently using the higher objective lens. However, the FOV is limited using the higher NA objective lens. The image processing with deep learning overcomes the trade-off relationship between FOV and fps.

Acknowledgement

This study was carried out at Laboratory of Biomedical Engineering, Division of Bioengineering and Bioinformatics, Graduate School of Information Science and Technology, Hokkaido University. I want to show my appreciation to everyone involved in my research.

I would like to express my most profound appreciation to Prof. Dr. Mamoru Hashimoto in Faculty of Information Science and Technology, Hokkaido University for supervising my six year's study, for providing the experimental devices and conditions, and for his outstanding advice and support. All of the experiences in this laboratory are precious for my work after graduation.

I would like to express my gratitude to Assoc. Prof. Dr. Nobuki Kudo in Faculty of Information Science and Technology, Hokkaido University for his advice, comment, and discussion from different disciplines in our group meeting.

I would like to show my greatest appreciation to Assist. Prof. Dr. Yuji Kato in Faculty of Information Science and Technology, Hokkaido University for providing his advice about optical fibers in our group meeting.

Prof. Dr. Hiroshi Uji-i and Prof. Dr. Hiroshi Hirata, members of the review committee, assessed my doctoral thesis. I have benefited from their advice, comment, question, and criticism.

I would like to show my most significant appreciation to Specially Appointed Assoc. Prof. Dr. Hirohiko Niioka in Institute for Datability Science, Osaka University for providing analytical advice, comment, suggestion, and technical support about deep learning.

I was financially supported by Japan Student Services Organization (JASSO) in my daily life in Master's course, and by Support Center for Advanced Telecommunications Technology Research (SCAT) and Japan Society of the promotion of science (JSPS) in my daily life and the study grant in Doctor's course. Also, I was supported for the attendance to SPIE international conference by Grants for Researchers Attending International Conferences in the NEC C&C Foundation (C&C) The study related to deep learning was partially supported by Hokkaido University DX Doctoral Fellowship, and Tateisi Science and Technology Foundation.

Finally, I would like to show my greatest appreciation to my parents.

Appendix

A. Source code of measurement GUI for deep learning using python

Figure A.1 shows a graphical user interface for acquiring CARS images and processing deep-learning models using python. The source code is shown on the next page. In ①, the scanning area with a galvanometer scanner (Xmin, Xmax, Ymin, Ymax), accumulation number (AccNum), repetition number (repeatNum), pixel size (pixelNum), and scanning speed (ScanFreq) are set. In ②, the checkbox selects the active image window shown in ③. CH0, CH1, and seg_is mean transmission image acquisition, CARS image acquisition and segmentation from CARS image, respectively. In ③, the acquired images are shown in each window. In ④, the color scale of the CARS image window is controlled. In ⑤, SignalAcquire, Abort, and CloseWindow mean starting acquiring images, stopping the scanning program, and closing the GUI panel, respectively. In ⑥, acquired images are saved in a directory of the textbox.

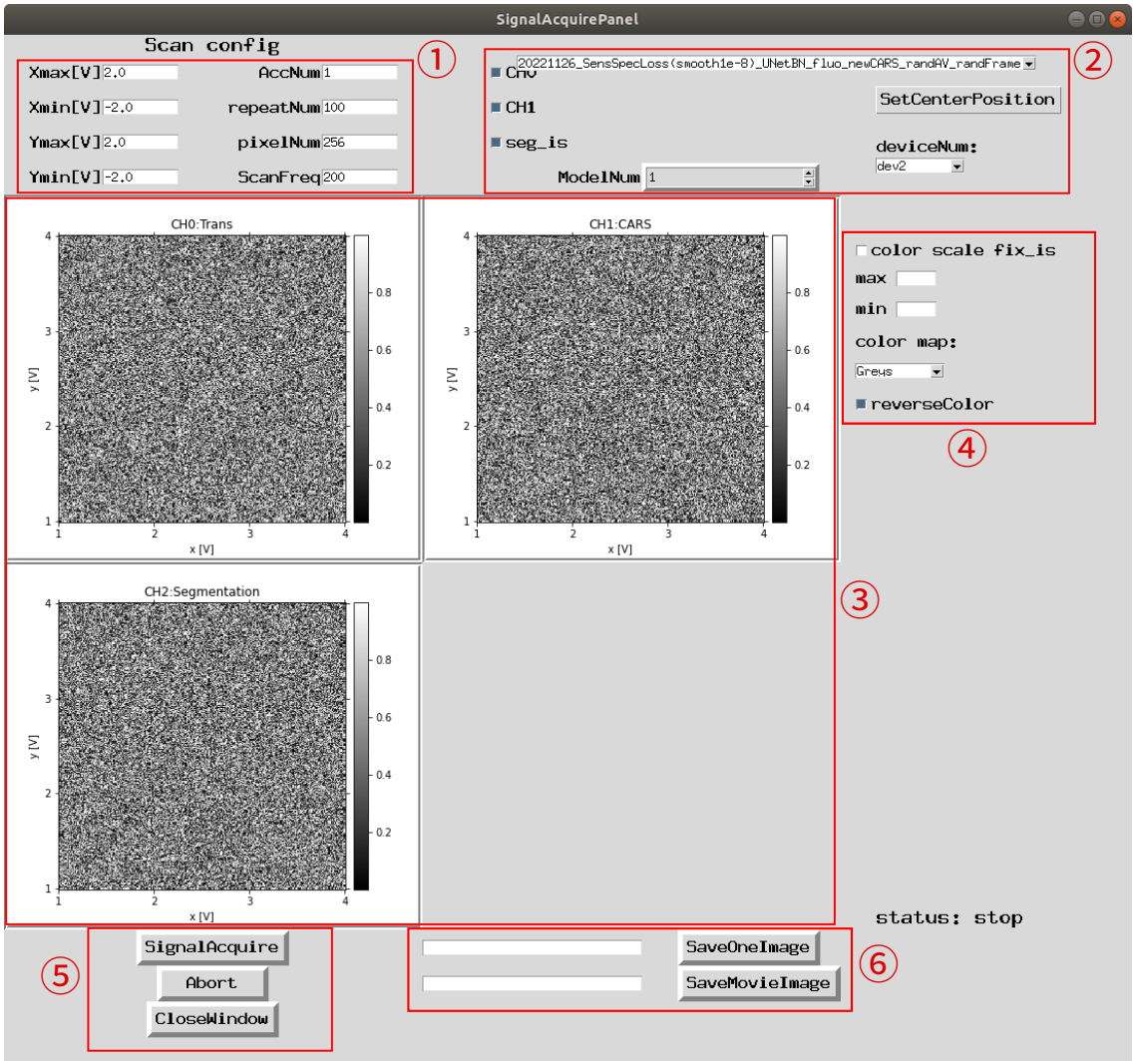


Figure A.1: GUI panel for acquiring CARS images and processing deep learning models using python.

```
import csv
from distutils.util import strtobool
import os
import pathlib
import time
import tkinter as tk
import tkinter.ttk as ttk
import threading

import cv2
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.cm as cm
from matplotlib.colors import Normalize
import mpl_toolkits.axes_grid1
import nidaqmx
from nidaqmx import constants
from nidaqmx.constants import AcquisitionType, Edge, Slope
from nidaqmx.stream_readers import AnalogMultiChannelReader
from nidaqmx.stream_writers import AnalogMultiChannelWriter
from nidaqmx.stream_readers import AnalogSingleChannelReader
import numpy as np
import torch

from segmentation_model20210516 import UNet_VGG, UNets, UNet

def WriteConstVoltage(DevName, channel, voltage):
    '''set constant voltage for 1 channel
    Args:
        DevName (str): device name (ex. dev1, dev2)
        channel (str): analog output channel (ex. ao0, ao1)
        voltage (float): output voltage [V]
    '''
    with nidaqmx.Task() as task:
        task.ao_channels.add_ao_voltage_chan('{}/{}'.format(DevName, channel))
        task.write(voltage)

class Application(tk.Frame):
    def __init__(self, master):
        super().__init__(master)
        self.master = master
        # window size
        self.width = 1100
```

```
44          self.height = 1000
45          # image window size
46          self.canvas_width = 400
47          self.canvas_height = 350
48          # interval for updating image window
49          self.after_time = 100 # [ms]
50
51          self.master.geometry('{}x{}'.format(self.width,self.height))
52          self.master.title('SignalAcquirePanel')
53
54          # initialize img
55          self.img_list = []
56          for i in range(3):
57              self.img_list.append(np.random.rand(256 * 256).reshape([256, 256]))
58          self.img_save_factor = 50
59          self.createWidgets()
60
61      def createWidgets(self):
62          '''create GUI panel'''
63          # setting style
64          s = ttk.Style()
65          s.configure('my.TLabel', font=('RictyDiminished-Regular', 12, 'bold'))
66          s.configure('my.TButton', font=('RictyDiminished-Regular', 12, 'bold'))
67          s.configure('my.TCheckbutton', font=('RictyDiminished-Regular', 12, 'bold'))
68
69          # scan configuretaion
70          ScanRange_label = ttk.Label(self.master,
71                                      font=('RictyDiminished', '14', 'bold'),
72                                      text='Scan config')
73          ScanRange_label.grid(row=0, column=0, columnspan=4)
74
75          label_pad = 8
76          # Xmax (galvano)
77          self.Xmax_label = ttk.Label(self.master,
78                                      style='my.TLabel',
79                                      text='Xmax[V]',
80                                      padding=[0,label_pad,0,label_pad])
81          self.Xmax_label.grid(row=1, column=0, sticky=tk.E)
82          self.Xmax_str = tk.StringVar()
83          self.Xmax_entry = ttk.Entry(self.master,
84                                      textvariable=self.Xmax_str,
85                                      width=10)
86          self.Xmax_entry.grid(row=1, column=1, sticky=tk.W)
87
```

```
88          # Xmin (galvano)
89          self.Xmin_label = ttk.Label(self.master,
90                                      style='my.TLabel',
91                                      text='Xmin[V]',
92                                      padding=[0,label_pad,0,label_pad])
93          self.Xmin_label.grid(row=2, column=0, sticky=tk.E)
94          self.Xmin_str = tk.StringVar()
95          self.Xmin_entry = ttk.Entry(self.master,
96                                      textvariable=self.Xmin_str,
97                                      width=10)
98          self.Xmin_entry.grid(row=2, column=1, sticky=tk.W)
99
100         # Ymax (galvano)
101         self.Ymax_label = ttk.Label(self.master,
102                                     style='my.TLabel',
103                                     text='Ymax[V]',
104                                     padding=[0,label_pad,0,label_pad])
105         self.Ymax_label.grid(row=3, column=0, sticky=tk.E)
106         self.Ymax_str = tk.StringVar()
107         self.Ymax_entry = ttk.Entry(self.master,
108                                     textvariable=self.Ymax_str,
109                                     width=10)
110         self.Ymax_entry.grid(row=3, column=1, sticky=tk.W)
111
112         # Ymin (galvano)
113         self.Ymin_label = ttk.Label(self.master,
114                                     style='my.TLabel',
115                                     text='Ymin[V]',
116                                     padding=[0,label_pad,0,label_pad])
117         self.Ymin_label.grid(row=4, column=0, sticky=tk.E)
118         self.Ymin_str = tk.StringVar()
119         self.Ymin_entry = ttk.Entry(self.master,
120                                     textvariable=self.Ymin_str,
121                                     width=10)
122         self.Ymin_entry.grid(row=4, column=1, sticky=tk.W)
123
124         # AccNum
125         self.AccNum_label = ttk.Label(self.master,
126                                       style='my.TLabel',
127                                       text='AccNum')
128         self.AccNum_label.grid(row=1, column=2, sticky=tk.E)
129         self.AccNum_str = tk.StringVar()
130         self.AccNum_entry = ttk.Entry(self.master,
131                                       textvariable=self.AccNum_str,
```

```
132                                     width=10)
133         self.AccNum_entry.grid(row=1, column=3, sticky=tk.W)
134 
135         # repeatNum
136         self.repeatNum_label = ttk.Label(self.master,
137                                          style='my.TLabel',
138                                          text='repeatNum')
139         self.repeatNum_label.grid(row=2, column=2, sticky=tk.E)
140         self.repeatNum_str = tk.StringVar()
141         self.repeatNum_entry = ttk.Entry(self.master,
142                                          textvariable=self.repeatNum_str,
143                                          width=10)
144         self.repeatNum_entry.grid(row=2, column=3, sticky=tk.W)
145 
146         # pixelNum
147         self.pixelNum_label = ttk.Label(self.master,
148                                         style='my.TLabel',
149                                         text='pixelNum')
150         self.pixelNum_label.grid(row=3, column=2, sticky=tk.E)
151         self.pixelNum_str = tk.StringVar()
152         self.pixelNum_entry = ttk.Entry(self.master,
153                                         textvariable=self.pixelNum_str,
154                                         width=10)
155         self.pixelNum_entry.grid(row=3, column=3, sticky=tk.W)
156 
157         # ScanFreq (when using 2 chans; max200, when using 1 chan; max400)
158         self.ScanFreq_label = ttk.Label(self.master,
159                                         style='my.TLabel',
160                                         text='ScanFreq')
161         self.ScanFreq_label.grid(row=4, column=2, sticky=tk.E)
162         self.ScanFreq_str = tk.StringVar()
163         self.ScanFreq_entry = ttk.Entry(self.master,
164                                         textvariable=self.ScanFreq_str,
165                                         width=10)
166         self.ScanFreq_entry.grid(row=4, column=3, sticky=tk.W)
167 
168         # checkbutton selecting channel and whether using segmentation or not
169         self.CH0_is = tk.BooleanVar()
170         self.CH0_is.set(True)
171         self.CH0_checkbox = ttk.Checkbutton(self.master,
172                                             text='CH0',
173                                             onvalue=True,
174                                             offvalue=False,
175                                             variable=self.CH0_is,
```

```
176                                         style='my.TCheckbutton',
177                                         )
178         self.CH0_checkbox.grid(row=1, column=5, sticky=tk.W)
179         self.CH1_is = tk.BooleanVar()
180         self.CH1_is.set(True)
181         self.CH1_checkbox = ttk.Checkbutton(self.master,
182                                         text='CH1',
183                                         onvalue=True,
184                                         offvalue=False,
185                                         variable=self.CH1_is,
186                                         style='my.TCheckbutton',
187                                         )
188         self.CH1_checkbox.grid(row=2, column=5, sticky=tk.W)
189         self.seg_is = tk.BooleanVar()
190         self.seg_is.set(True)
191         self.seg_checkbox = ttk.Checkbutton(self.master,
192                                         text='seg_is',
193                                         onvalue=True,
194                                         offvalue=False,
195                                         variable=self.seg_is,
196                                         style='my.TCheckbutton',
197                                         )
198         self.seg_checkbox.grid(row=3, column=5, sticky=tk.W)
199
200         # weight dir
201 #         self.weight_str = tk.StringVar()
202 #         self.weight_entry = ttk.Entry(self.master, textvariable=self.weight_str, width=30)
203 #         self.weight_entry.grid(row=1, column=7, columnspan=3, sticky=tk.W)
204         # model type
205         self.segModel_str = tk.StringVar()
206         self.segModel_combo = ttk.Combobox(self.master,
207                                         textvariable=self.segModel_str,
208                                         style='my.TCombobox',
209                                         width=70)
210         self.segModel_combo['values'] = \
211         ['20221126_SensSpecLoss(smooth1e-8)_UNetBN_fluo_newCARS_randAV_randFrame_fixhalf']
212         self.segModel_combo.current(0)
213         self.segModel_combo.place(x=500, y=20)
214
215
216         # Save dir
217         self.SaveOneImage_str = tk.StringVar()
218         self.SaveOneImage_entry = ttk.Entry(self.master,
219                                         textvariable=self.SaveOneImage_str,
```

```
220                                         width=30)
221         self.SaveOneImage_entry.grid(row=7, column=4, columnspan=2, sticky=tk.E)
222         self.SaveMovieImage_str = tk.StringVar()
223         self.SaveMovieImage_entry = ttk.Entry(self.master,
224                                               textvariable=self.SaveMovieImage_str,
225                                               width=30)
226         self.SaveMovieImage_entry.grid(row=8, column=4, columnspan=2, sticky=tk.W)
227 
228         # Frame settings
229         self.CanvasFrame_list = []
230         for i in range(3):
231             self.CanvasFrame_list.append(tk.Frame(self.master, bd=4, relief=tk.GROOVE))
232         self.frame_1 = tk.Frame(self.master, bd=4, relief=tk.RAISED)
233         self.frame_2 = tk.Frame(self.master, bd=4, relief=tk.RAISED)
234         self.frame_3 = tk.Frame(self.master, bd=4, relief=tk.RAISED)
235         self.frame_4 = tk.Frame(self.master, bd=4, relief=tk.RAISED)
236         self.frame_5 = tk.Frame(self.master, bd=4, relief=tk.RAISED)
237         self.frame_6 = tk.Frame(self.master, bd=4, relief=tk.RAISED)
238 #         self.frame_7 = tk.Frame(self.master, bd=4, relief=tk.RAISED)
239 
240         # widget settings
241         self.title_list = ['CH0:Trans', 'CH1:CARS', 'CH2:Segmentation']
242         self.canvas_list = []
243         self.fig_list = []
244         self.ax_list = []
245         self.pcolor_list = []
246         for i, title in enumerate(self.title_list):
247             fig, ax, pcolor = self.init_plot(self.img_list[i], title, False, None, None)
248             self.fig_list.append(fig)
249             self.ax_list.append(ax)
250             self.pcolor_list.append(pcolor)
251             self.canvas_list.append(FigureCanvasTkAgg(self.fig_list[-1],
252                                                       self.CanvasFrame_list[i]))
253         self.btn1 = ttk.Button(self.frame_1,
254                                style='my.TButton',
255                                text='SignalAcquire',
256                                command=self.Start)
257         self.btn2 = ttk.Button(self.frame_2,
258                                style='my.TButton',
259                                text='Abort',
260                                command=self.Abort)
261         self.btn3 = ttk.Button(self.frame_3,
262                                style='my.TButton',
263                                text='CloseWindow',
```

```
264                                 command=self.CloseWindow)
265         self.btn4 = ttk.Button(self.frame_4,
266                                 style='my.TButton',
267                                 text='SaveOneImage',
268                                 command=self.SaveOneImage)
269         self.btn5 = ttk.Button(self.frame_5,
270                                 style='my.TButton',
271                                 text='SaveMovieImage',
272                                 command=self.SaveMovieImage)
273 
274         # model number
275         self.ModelNum_label = ttk.Label(self.master,
276                                         style='my.TLabel',
277                                         text='ModelNum')
278         self.ModelNum_label.grid(row=4, column=5, sticky=tk.E)
279         self.ModelNum_val = tk.StringVar()
280         self.ModelNum_val.set('1')
281         self.ModelNum_Sbox = ttk.Spinbox(self.frame_6,
282                                 state='readonly',
283                                 textvariable=self.ModelNum_val,
284                                 from_=1,
285                                 to=60,
286                                 increment=1)
287 
288         # put widget
289         self.CanvasFrame_list[0].grid(row=5, column=0, columnspan=4)
290         self.CanvasFrame_list[1].grid(row=5, column=4, columnspan=4)
291         self.CanvasFrame_list[2].grid(row=6, column=0, columnspan=4)
292 
293         self.frame_1.grid(row=7, column=0, columnspan=4)
294         self.frame_2.grid(row=8, column=0, columnspan=4)
295         self.frame_3.grid(row=9, column=0, columnspan=4)
296         self.frame_4.grid(row=7, column=7, columnspan=2, sticky=tk.W)
297         self.frame_5.grid(row=8, column=7, columnspan=2, sticky=tk.W)
298         self.frame_6.grid(row=4, column=6, columnspan=3, sticky=tk.W)
299 #         self.frame_7.grid(row=2, column=7, sticky=tk.W)
300 
301         for i in range(3):
302             self.canvas_list[i].get_tk_widget().config(width=self.canvas_width,
303                                                        height=self.canvas_height)
304         self.canvas_list[0].get_tk_widget().grid(row=6, column=0, columnspan=8, rowspan=4)
305         self.canvas_list[1].get_tk_widget().grid(row=6, column=1, columnspan=8, rowspan=4)
306         self.canvas_list[2].get_tk_widget().grid(row=7, column=0, columnspan=8, rowspan=4)
307         self.btn1.grid(row=0, column=0)
```

```
308         self.btn2.grid(row=0, column=0)
309         self.btn3.grid(row=0, column=0)
310         self.btn4.grid(row=0, column=0)
311         self.btn5.grid(row=0, column=0)
312 #         self.btn7.grid(row=0, column=0)
313         self.ModelNum_Sbox.grid(row=0, column=0)
314 
315         # color bar
316         self.fix_is = tk.BooleanVar()
317         self.fix_is.set(False)
318         self.fix_checkbox = ttk.Checkbutton(self.master,
319                                             text='color scale fix_is',
320                                             onvalue=True,
321                                             offvalue=False,
322                                             variable=self.fix_is,
323                                             style='my.TCheckbutton',
324                                             )
325         self.fix_checkbox.place(x=830, y=200)
326 
327         # color max
328         self.colormax_label = ttk.Label(self.master,
329                                         style='my.TLabel',
330                                         text='max',
331                                         padding=[0,label_pad,0,label_pad])
332         self.colormax_label.place(x=830, y=220)
333         self.colormax_str = tk.StringVar()
334         self.colormax_entry = ttk.Entry(self.master,
335                                         textvariable=self.colormax_str,
336                                         width=5)
337         self.colormax_entry.place(x=870, y=230)
338 
339         # color min
340         self.colormin_label = ttk.Label(self.master,
341                                         style='my.TLabel',
342                                         text='min',
343                                         padding=[0,label_pad,0,label_pad])
344         self.colormin_label.place(x=830, y=250)
345         self.colormin_str = tk.StringVar()
346         self.colormin_entry = ttk.Entry(self.master,
347                                         textvariable=self.colormin_str,
348                                         width=5)
349         self.colormin_entry.place(x=870, y=260)
350 
351         # color map
```

```
352         self.cmap_label = ttk.Label(self.master, style='my.TLabel', text='color map:')
353         self.cmap_label.place(x=830, y=290)
354         self.cmap_str = tk.StringVar()
355         self.cmap_str.set('Greys')
356         self.cmap_combo = ttk.Combobox(self.master,
357                                        textvariable=self.cmap_str,
358                                        style='my.TCombobox',
359                                        width=10)
360         self.cmap_combo['values'] = ['Greys', 'inferno', 'rainbow', 'hot', 'cool']
361         self.cmap_combo.current(0)
362         self.cmap_combo.place(x=830, y=320)
363 
364         # reverse color map
365         self.reverse_is = tk.BooleanVar()
366         self.reverse_is.set(True)
367         self.reverse_checkbox = ttk.Checkbutton(self.master,
368                                                 text='reverseColor',
369                                                 onvalue=True,
370                                                 offvalue=False,
371                                                 variable=self.reverse_is,
372                                                 style='my.TCheckbutton',
373                                               )
374         self.reverse_checkbox.place(x=830, y=350)
375 
376         # set center position
377         self.setCenterPosition_btn = ttk.Button(self.master,
378                                                 style='my.TButton',
379                                                 text='SetCenterPosition',
380                                                 command=self.SetCenterPosition)
381         self.setCenterPosition_btn.place(x=850, y=50)
382 
383         # devNum
384         self.devNum_label = ttk.Label(self.master, style='my.TLabel', text='deviceNum:')
385         self.devNum_label.place(x=850, y=100)
386         self.devNum_str = tk.StringVar()
387         self.devNum_str.set('dev1')
388         self.devNum_combo = ttk.Combobox(self.master,
389                                          textvariable=self.devNum_str,
390                                          style='my.TCombobox',
391                                          width=10)
392         self.devNum_combo['values'] = ['dev1', 'dev2']
393         self.devNum_combo.current(0)
394         self.devNum_combo.place(x=850, y=120)
395 
```

```
396         # window status
397         self.status_label = ttk.Label(self.master,
398                                       font=('RictyDiminished', '14', 'bold'),
399                                       text='status: stop')
400         self.status_label.place(x=850, y=850)
401 
402         self.LoadConfig(pathlib.Path(''))
403 
404     def init_plot(self, img_array, title, fix_is, colormax, colormin):
405         '''initialize image windows'''
406         # make Figure instance
407         fig = plt.Figure()
408         plt.gray()
409 
410         # make Axes ` and set scales at up, down, left, and right
411         ax1 = fig.add_subplot(111)
412         fig.subplots_adjust(left=0, right=1, bottom=0.1, top=0.9)
413         ax1.yaxis.set_ticks_position('both')
414         ax1.xaxis.set_ticks_position('both')
415 
416         # plot data
417         ax1.imshow(img_array)
418 
419         # set labels and ticks
420         ax1.set_title(title)
421         nx = 256
422         no_labels = 4 # how many labels to see on axis x
423         step_x = int(nx / (no_labels - 1)) # step between consecutive labels
424         x_positions = np.arange(0, nx, step_x) # pixel count at label position
425         y_positions = x_positions[::-1]
426         x_labels = [1,2,3,4] # labels you want to see on x
427         y_labels = [1,2,3,4] # labels you want to see on y
428         ax1.set_xticks(x_positions)
429         ax1.set_yticks(y_positions)
430         ax1.set_xticklabels(x_labels)
431         ax1.set_yticklabels(y_labels)
432         ax1.set_xlabel('x [V]')
433         ax1.set_ylabel('y [V]')
434 
435         if fix_is:
436             mappable1 = ax1.pcolormesh(range(img_array.shape[0]),
437                                        range(img_array.shape[1]),
438                                        img_array,
439                                        norm=Normalize(vmin=colormin,
```

```
                                                vmax=colormax))
        else:
            mappable1 = ax1.pcolormesh(range(img_array.shape[0]),
                                       range(img_array.shape[1]),
                                       img_array,
                                       norm=Normalize(vmin=img_array.min(),
                                                      vmax=img_array.max()))

        divider1 = mpl_toolkits.axes_grid1.make_axes_locatable(ax1)
        cax1 = divider1.append_axes('right', '5%', pad='3%')
        pp1 = fig.colorbar(mappable1, cax=cax1, orientation='vertical')

        return fig, ax1, pp1

    def PlotImg(self, fig, ax, pp1, img_array, title, fix_is, colormax, colormin):
        '''update acquired images'''
        # clear before plot
        ax.clear()
        # clear color bar
        try:
            pp1.remove()
        except KeyError:
            print('not exist pp1')

        # plot data
        ax.imshow(img_array)

        # set labels
        ax.set_title(title)
        ax.set_xlabel('x [V]')
        ax.set_ylabel('y [V]')

        # set ticks
        nx = self.pixelNum
        no_labels = 4 # how many labels to see on axis x
        step_x = int(nx / (no_labels - 1)) # step between consecutive labels
        x_positions = np.arange(0, nx, step_x) # pixel count at label position
        y_positions = x_positions[::-1]
        delta_galvano = round(self.GS_Xmax - self.GS_Xmin, 2)
        step_galvano = delta_galvano / (no_labels - 1)
        x_labels = [round(self.GS_Xmin+i*step_galvano,2) for i in range(4)]
        y_labels = [round(self.GS_Ymin+i*step_galvano,2) for i in range(4)]
        ax.set_xticks(x_positions)
        ax.set_yticks(y_positions)
```

```
484         ax.set_xticklabels(x_labels)
485         ax.set_yticklabels(y_labels)
486 
487         if self.reverse_is.get():
488             colorMap = self.cmap_str.get() + '_r'
489         else:
490             colorMap = self.cmap_str.get()
491 
492         if fix_is:
493             colormax = float(self.colormax_str.get())
494             colormin = float(self.colormin_str.get())
495             mappable1 = ax.pcolormesh(range(img_array.shape[0]),
496                                       range(img_array.shape[1]),
497                                       img_array,
498                                       cmap=colorMap,
499                                       norm=Normalize(vmin=colormin,
500                                                      vmax=colormax))
501         else:
502             mappable1 = ax.pcolormesh(range(img_array.shape[0]),
503                                       range(img_array.shape[1]),
504                                       img_array,
505                                       cmap=colorMap,
506                                       norm=Normalize(vmin=img_array.min(),
507                                                      vmax=img_array.max()))
508 
509         divider1 = mpl_toolkits.axes_grid1.make_axes_locatable(ax)
510         cax1 = divider1.append_axes('right', '5%', pad='3%')
511         pp1 = fig.colorbar(mappable1, cax=cax1, orientation='vertical')
512         return pp1
513 
514     def update(self, roop_is=True):
515         '''update acquired images every specified time'''
516         if self.update_flag == 1:
517             for i in range(3):
518                 self.fig_list[i].clear()
519 
520                 # make Axes、and set scales at up, down, left, and right
521                 ax = self.fig_list[i].add_subplot(111)
522                 self.fig_list[i].subplots_adjust(left=0,
523                                                  right=1,
524                                                  bottom=0.1,
525                                                  top=0.9)
526                 ax.yaxis.set_ticks_position('both')
527                 ax.xaxis.set_ticks_position('both')
```

```
528
529                 # plot data
530                 ax.imshow(self.img_list[i])
531
532                 # set labels
533                 ax.set_title(self.title_list[i])
534                 ax.set_xlabel('x [V]')
535                 ax.set_ylabel('y [V]')
536
537                 # set ticks
538                 nx = self.pixelNum
539                 no_labels = 4 # how many labels to see on axis x
540                 step_x = int(nx / (no_labels - 1)) # step between consecutive labels
541                 x_positions = np.arange(0, nx, step_x) # pixel count at label position
542                 y_positions = x_positions[::-1]
543                 delta_galvano = round(self.GS_Xmax - self.GS_Xmin, 2)
544                 step_galvano = delta_galvano / (no_labels - 1)
545                 x_labels = [round(self.GS_Xmin+i*step_galvano,2) for i in range(4)]
546                 y_labels = [round(self.GS_Ymin+i*step_galvano,2) for i in range(4)]
547                 ax.set_xticks(x_positions)
548                 ax.set_yticks(y_positions)
549                 ax.set_xticklabels(x_labels)
550                 ax.set_yticklabels(y_labels)
551
552                 # set colorMap
553                 if self.reverse_is.get():
554                     colorMap = self.cmap_str.get() + '_r'
555                 else:
556                     colorMap = self.cmap_str.get()
557
558                 if self.fix_is.get() and i == 1:
559                     colormax = float(self.colormax_str.get())
560                     colormin = float(self.colormin_str.get())
561                     mappable1 = ax.pcolormesh(range(self.img_list[i].shape[0]),
562                                               range(self.img_list[i].shape[1]),
563                                               self.img_list[i],
564                                               cmap=colorMap,
565                                               norm=Normalize(vmin=colormin,
566                                                              vmax=colormax))
567                 elif i == 2:
568                     colormax = 1.0
569                     colormin = 0.0
570                     mappable1 = ax.pcolormesh(range(self.img_list[i].shape[0]),
571                                               range(self.img_list[i].shape[1]),
```

```
572                                             self.img_list[i],
573                                             cmap=colorMap,
574                                             norm=Normalize(vmin=colormin,
575                                                            vmax=colormax))
576                 else:
577                     mappable1 = ax.pcolormesh(range(self.img_list[i].shape[0]),
578                                             range(self.img_list[i].shape[1]),
579                                             self.img_list[i],
580                                             cmap=colorMap,
581                                             norm=Normalize(vmin=self.img_list[i].min(),
582                                                            vmax=self.img_list[i].max()))
583
584                 divider1 = mpl_toolkits.axes_grid1.make_axes_locatable(ax)
585                 cax1 = divider1.append_axes('right', '5%', pad='3%')
586                 pp1 = self.fig_list[i].colorbar(mappable1, cax=cax1, orientation='vertical')
587
588                 # redraw canvas
589                 self.canvas_list[i].get_tk_widget().config(width=self.canvas_width,
590                                                            height=self.canvas_height)
591                 self.canvas_list[i].draw()
592
593         if roop_is:
594             self.after_id = self.master.after(self.after_time, self.update)
595 #         print('{}: plot update'.format(time.time()))
596 #         print('after_id in update func:{}'.format(self.after_id))
597         self.update_flag = 0
598
599     def Start(self):
600         '''start all threads'''
601         self.status_label['text'] = 'status: imaging'
602         self.GS_Xmax = float(self.Xmax_str.get())
603         self.GS_Xmin = float(self.Xmin_str.get())
604         self.GS_Ymax = float(self.Ymax_str.get())
605         self.GS_Ymin = float(self.Ymin_str.get())
606         if self.GS_Xmax>5 or self.GS_Xmin <-5 or self.GS_Ymax>5 or self.GS_Ymin <-5:
607             raise ValueError('this voltage is not supported')
608         self.AccNum = int(self.AccNum_str.get())
609         self.repeatNum = int(self.repeatNum_str.get())
610         self.pixelNum = int(self.pixelNum_str.get())
611         self.ScanFreq = int(self.ScanFreq_str.get())
612         self.CH0_get_is = self.CH0_is.get()
613         self.CH1_get_is = self.CH1_is.get()
614         self.seg_get_is = self.seg_is.get()
615         self.devNum = self.devNum_str.get()
```

```
616         print(self.devNum)
617 
618         if self.CH0_get_is and self.CH1_get_is:
619             self.ai_channel = '{}/ai0:1'.format(self.devNum)
620         elif self.CH0_get_is:
621             self.ai_channel = '{}/ai0'.format(self.devNum)
622         elif self.CH1_get_is:
623             self.ai_channel = '{}/ai1'.format(self.devNum)
624 
625         # initialize img
626         self.img_list = []
627         for i in range(3):
628             temp = np.random.rand(self.pixelNum**2).reshape([self.pixelNum, self.pixelNum])
629             self.img_list.append(temp)
630         self.movie_list0 = []
631         self.movie_list1 = []
632         self.movie_list2 = []
633 
634         self.SaveConfig(pathlib.Path(''))
635         self.MakeDAQarray()
636 
637         # initialize Seg models
638         if self.seg_get_is:
639             print('initialize seg models')
640             self.init_SegModels()
641 
642         self.update_flag = 0
643         self.after_id = self.master.after(self.after_time, self.update)
644         # print('after_id in start func:{}'.format(self.after_id))
645 
646         # staring SubThread
647         self.threadFlag = True
648         self.thread_acquire = threading.Thread(target=self.SubThreadSignalAcquire)
649         self.thread_acquire.start()
650 
651         # print('mainThread has started')
652 
653     def Abort(self):
654         '''Abort all subthread and updating images'''
655         print('Abort button')
656         # stop updating plot
657         self.master.after_cancel(self.after_id)
658 
659         # stop SubThread
```

```
660         self.threadFlag = False
661         self.thread_acquire.join()
662         #print('Abort thread_acquire')
663 
664         # update image
665         self.update(roop_is=False)
666         self.status_label['text'] = 'status: stop'
667         # print('Abort func has finished')
668 
669     def CloseWindow(self):
670         '''close GUI panel'''
671         self.master.quit()
672         self.master.destroy()
673 
674     def SubThreadSignalAcquire(self):
675         '''subthread to acquire images'''
676         print('SubThreadSignalAcquire starts')
677         self.ExposureTime_list = []
678         self.ModelProcessingTime_list = []
679         i = 0
680         while self.threadFlag and i < self.repeatNum:
681             scan_start = time.time()
682             self.FastScan()
683             scan_end = time.time()
684             print('{0}: data scan{1}, process time:{2:.2f}'\
685                   .format(time.time(), i+1, scan_end-scan_start))
686             self.ExposureTime_list.append(scan_end-scan_start)
687 
688             '''deep learning process (if seg is true)'''
689             if self.seg_get_is:
690                 seg_start = time.time()
691                 self.toCUDA()
692                 self.segmentation()
693                 seg_end = time.time()
694                 self.ModelProcessingTime_list.append(seg_end-seg_start)
695 
696             self.addMovieList()
697             self.update_flag = 1
698             i += 1
699 
700         self.SaveTimeList(pathlib.Path(''))
701         if i == self.repeatNum:
702             print('SignalAcquire is done')
703             self.status_label['text'] = 'status: stop'
```

```
704         else:
705             print('SignalAcquire is aborted')
706 
707     def FastScan(self):
708         '''scan galvano, acquire images, transfer data to CUDA'''
709 
710         self.ScanFlag = 0
711 
712         img_list0 = []
713         img_list1 = []
714         for i in range(self.AccNum):
715             with nidaqmx.Task() as write_task, nidaqmx.Task() as read_task, \
716             nidaqmx.Task() as sample_clk_task:
717                 sample_clk_task.co_channels.add_co_pulse_chan_freq('{}/ctr0'.format(self.devNum),
718                                                                    freq=self.sample_rate)
719                 sample_clk_task.timing.cfg_implicit_timing(samps_per_chan=self.X.size)
720                 samp_clk_terminal = '/{}/Ctr0InternalOutput'.format(self.devNum)
721 
722                 write_task.ao_channels.add_ao_voltage_chan('{}/ao0:1'.format(self.devNum),
723                                                            max_val=10,
724                                                            min_val=-10)
725                 write_task.timing.cfg_samp_clk_timing(self.sample_rate,
726                                                       source=samp_clk_terminal,
727                                                       active_edge=Edge.RISING,
728                                                       samps_per_chan=self.X.size)
729                 read_task.ai_channels.add_ai_voltage_chan(self.ai_channel,
730                                                           max_val=10,
731                                                           min_val=-10)
732                 read_task.timing.cfg_samp_clk_timing(self.sample_rate,
733                                                       source=samp_clk_terminal,
734                                                       active_edge=Edge.FALLING,
735                                                       samps_per_chan=self.X.size)
736 
737                 writer = AnalogMultiChannelWriter(write_task.out_stream)
738                 if self.ai_channel == '{}/ai0:1'.format(self.devNum):
739                     reader = AnalogMultiChannelReader(read_task.in_stream)
740                 else:
741                     reader = AnalogSingleChannelReader(read_task.in_stream)
742                 t1 = time.time()
743                 if self.ScanFlag==0:
744                     writer.write_many_sample(self.outputData1)
745                     self.ScanFlag = 1
746                 else:
747                     writer.write_many_sample(self.outputData2)
```

```
748                     self.ScanFlag = 0
749
750                 read_task.start()
751                 write_task.start()
752                 sample_clk_task.start()
753                 reader.read_many_sample(self.inputData,
754                                         number_of_samples_per_channel=self.X.size,
755                                         timeout=2)
756                 t2 = time.time()
757 #                 print('exp:{:.2f} s'.format(t2-t1))
758
759                 np_img0 = np.zeros((self.pixelNum, self.pixelNum+self.edge_pixel))
760                 np_img1 = np.zeros((self.pixelNum, self.pixelNum+self.edge_pixel))
761                 if self.ai_channel == '{}/ai0:1'.format(self.devNum):
762                     for j in range(self.pixelNum):
763                         start = self.pixelNum*j + self.edge_pixel*j + \
764                         int(self.pixelNum*self.return_ratio)*j
765                         end = start + self.pixelNum + self.edge_pixel
766                         np_img0[j,:] = self.inputData[0,start:end]
767                         np_img1[j,:] = self.inputData[1,start:end]
768                     img_list0.append(np_img0[:,-self.pixelNum:].copy())
769                     img_list1.append(np_img1[:,-self.pixelNum:].copy())
770                 else:
771                     for j in range(self.pixelNum):
772                         start = self.pixelNum*j + self.edge_pixel*j + \
773                         int(self.pixelNum*self.return_ratio)*j
774                         end = start + self.pixelNum + self.edge_pixel
775                         np_img0[j,:] = self.inputData[start:end]
776                     img_list0.append(np_img0[:,-self.pixelNum:].copy())
777
778         if self.ai_channel == '{}/ai0'.format(self.devNum):
779             self.img_list[0] = np.mean(np.array(img_list0), axis=0)
780         elif self.ai_channel == '{}/ai1'.format(self.devNum):
781             self.img_list[1] = np.mean(np.array(img_list0), axis=0)
782         elif self.ai_channel == '{}/ai0:1'.format(self.devNum):
783             self.img_list[0] = np.mean(np.array(img_list0), axis=0)
784             self.img_list[1] = np.mean(np.array(img_list1), axis=0)
785
786     def MakeDAQarray(self):
787         '''make voltage array for scanning galvano and data array for recieving signals'''
788
789         edge_ratio = 0.2
790         self.edge_pixel = int(self.pixelNum*edge_ratio)*2
791         self.return_ratio = 0.4
```

```
792         pixelY = self.pixelNum
793         pixelX = self.pixelNum + self.edge_pixel+int(self.pixelNum*self.return_ratio)
794         Xmax_edge = self.GS_Xmax + (self.GS_Xmax-self.GS_Xmin)*edge_ratio
795         Xmin_edge = self.GS_Xmin - (self.GS_Xmax-self.GS_Xmin)*edge_ratio
796 
797         # when ScanFlag=0
798         tempX1 = np.linspace(Xmax_edge, Xmin_edge, pixelY+self.edge_pixel)
799         returnX1 = np.linspace(Xmin_edge,
800                                Xmax_edge,
801                                int(self.pixelNum*self.return_ratio))
802         outputX1 = np.append(tempX1, returnX1)
803         tempY1 = np.linspace(self.GS_Ymax, self.GS_Ymin, pixelY)
804         outputY1 = np.ones(pixelX)*tempY1[0]
805         for i in range(pixelY-1):
806             outputX1 = np.append(outputX1, tempX1)
807             outputX1 = np.append(outputX1, returnX1)
808             outputY1 = np.append(outputY1, np.ones(pixelX)*tempY1[i+1])
809         self.outputData1 = np.array([outputX1, outputY1])
810         print(outputX1.shape, outputY1.shape)
811         self.outputData1 = self.outputData1.reshape((2,len(outputX1)))
812 
813         # when ScanFlag=1
814         tempX2 = np.linspace(Xmax_edge, Xmin_edge, pixelY+self.edge_pixel)
815         returnX2 = np.linspace(Xmin_edge,
816                                Xmax_edge,
817                                int(self.pixelNum*self.return_ratio))
818         outputX2 = np.append(tempX2, returnX2)
819         tempY2 = np.linspace(self.GS_Ymax, self.GS_Ymin, pixelY)
820         outputY2 = np.ones(pixelX)*tempY2[0]
821         for i in range(pixelY-1):
822             outputX2 = np.append(outputX2, tempX2)
823             outputX2 = np.append(outputX2, returnX2)
824             outputY2 = np.append(outputY2, np.ones(pixelX)*tempY2[i+1])
825         self.outputData2 = np.array([outputX2, outputY2])
826         self.outputData2 = self.outputData2.reshape((2,len(outputX2)))
827 
828         if self.ai_channel == '{}/ai0:1'.format(self.devNum):
829             self.inputData = np.zeros(self.outputData1.shape[1]*2)
830             self.inputData = self.inputData.reshape((2,self.outputData1.shape[1]))
831         else:
832             self.inputData = np.zeros(self.outputData1.shape[1])
833 
834         self.X = np.zeros(self.outputData1.shape[1])
835         self.sample_rate = self.ScanFreq * (self.pixelNum+self.edge_pixel/2) * 2
```

```
836
837         print('sample_rate:{}'.format(self.sample_rate))
838
839     def SaveTimeList(self, dir_path):
840         result = []
841         result.append(['exposure time [s] per one scan '])
842         result.append(self.ExposureTime_list)
843         save_path = dir_path.joinpath('ExposureTime.csv')
844         with open(save_path, 'w') as f:
845             writer = csv.writer(f, lineterminator='\n')
846             writer.writerows(result)
847
848         if self.seg_get_is:
849             result = []
850             result.append(['model processing time [s] per one image'])
851             result.append(self.ModelProcessingTime_list)
852             save_path = dir_path.joinpath('ModelProcessingTime.csv')
853             with open(save_path, 'w') as f:
854                 writer = csv.writer(f, lineterminator='\n')
855                 writer.writerows(result)
856
857     def SaveConfig(self, dir_path):
858         '''save scanning configuration to config.csv'''
859
860         GS_Xmax = float(self.Xmax_str.get())
861         GS_Xmin = float(self.Xmin_str.get())
862         GS_Ymax = float(self.Ymax_str.get())
863         GS_Ymin = float(self.Ymin_str.get())
864         GS_list = [GS_Xmax, GS_Xmin, GS_Ymax, GS_Ymin]
865         AccNum = int(self.AccNum_str.get())
866         repeatNum = int(self.repeatNum_str.get())
867         pixelNum = int(self.pixelNum_str.get())
868         ScanFreq = int(self.ScanFreq_str.get())
869         modelNum = int(self.ModelNum_val.get())
870
871         save_list = []
872         save_list.append(['GS_Xmax', GS_Xmax])
873         save_list.append(['GS_Xmin', GS_Xmin])
874         save_list.append(['GS_Ymax', GS_Ymax])
875         save_list.append(['GS_Ymin', GS_Ymin])
876         save_list.append(['AccNum', AccNum])
877         save_list.append(['repeatNum', repeatNum])
878         save_list.append(['pixelNum', pixelNum])
879         save_list.append(['ScanFreq', ScanFreq])
```

```
880         save_list.append(['CH0_is', self.CH0_is.get()])
881         save_list.append(['CH1_is', self.CH1_is.get()])
882         save_list.append(['seg_is', self.seg_is.get()])
883         save_list.append(['modelNum', modelNum])
884         save_list.append(['devNum', self.devNum])
885 
886         save_path = dir_path.joinpath('config.csv')
887         with open(save_path, 'w') as f:
888             writer = csv.writer(f, lineterminator='\n')
889             writer.writerows(save_list)
890         print('SaveConfig')
891 
892     def LoadConfig(self, dir_path):
893         '''load scanning configuration from config.csv'''
894 
895         load_path = dir_path.joinpath('config.csv')
896         if load_path.exists():
897             with open(load_path) as f:
898                 reader = csv.reader(f)
899                 config_list = [row[1] for row in reader]
900             self.Xmax_str.set(config_list[0])
901             self.Xmin_str.set(config_list[1])
902             self.Ymax_str.set(config_list[2])
903             self.Ymin_str.set(config_list[3])
904             self.AccNum_str.set(config_list[4])
905             self.repeatNum_str.set(config_list[5])
906             self.pixelNum_str.set(config_list[6])
907             self.ScanFreq_str.set(config_list[7])
908             self.CH0_is.set(strtobool(config_list[8]))
909             self.CH1_is.set(strtobool(config_list[9]))
910             self.seg_is.set(strtobool(config_list[10]))
911             self.ModelNum_val.set(config_list[11])
912             temp_devnum = config_list[12]
913             devnum_index = self.devNum_combo['values'].index(temp_devnum)
914             self.devNum_combo.current(devnum_index)
915             print('loading config csv')
916         else:
917             print('config csv does not exist')
918 
919     def addMovieList(self):
920         '''add acquired image to movie_list'''
921         if self.CH0_get_is:
922             self.movie_list0.append(self.img_list[0])
923         if self.CH1_get_is:
```

```
924             self.movie_list1.append(self.img_list[1])
925         if self.seg_get_is:
926             self.movie_list2.append(self.img_list[2])
927 
928     def SaveOneImage(self):
929         '''save one image in each checked window to specified directory'''
930 
931         print('SaveOneImage')
932         print(self.SaveOneImage_str.get())
933         save_dir = pathlib.Path(self.SaveOneImage_str.get())
934         os.makedirs(save_dir, exist_ok=True)
935         self.SaveConfig(save_dir)
936         self.SaveTimeList(save_dir)
937         if self.CH0_get_is:
938             save_path = save_dir.joinpath('CH0-trans.npy')
939             np.save(save_path, self.img_list[0])
940             save_path = save_dir.joinpath('CH0-trans.png')
941             cv2.imwrite(str(save_path),
942                         self.img_list[0]/self.img_list[0].max()*256)
943 #             cv2.imwrite(str(save_path), self.img_list[0]*self.img_save_factor)
944         if self.CH1_get_is:
945             save_path = save_dir.joinpath('CH1-CARS.npy')
946             np.save(save_path, self.img_list[1])
947             save_path = save_dir.joinpath('CH1-CARS.png')
948 #             cv2.imwrite(str(save_path), self.img_list[1]*self.img_save_factor)
949             cv2.imwrite(str(save_path),
950                         self.img_list[1]/self.img_list[1].max()*256)
951         if self.seg_get_is:
952             save_path = save_dir.joinpath('segmentation.npy')
953             np.save(save_path, self.img_list[2])
954             save_path = save_dir.joinpath('segmentation.png')
955             cv2.imwrite(str(save_path), self.img_list[2])
956 
957     def SaveMovieImage(self):
958         '''save movie image in each checked window to specified directory'''
959 
960         print('SaveMovieImage')
961         save_dir = pathlib.Path(self.SaveMovieImage_str.get())
962         os.makedirs(save_dir, exist_ok=True)
963         self.SaveConfig(save_dir)
964         self.SaveTimeList(save_dir)
965         if self.CH0_get_is:
966             save_path = save_dir.joinpath('CH0-trans.npy')
967             np.save(save_path, self.movie_list0)
```

```
968         if self.CH1_get_is:
969             save_path = save_dir.joinpath('CH1-CARS.npy')
970             np.save(save_path, self.movie_list1)
971         if self.seg_get_is:
972             save_path = save_dir.joinpath('segmentation.npy')
973             np.save(save_path, self.movie_list2)
974 
975     def init_SegModels(self):
976         '''initialize segmentation model'''
977         time_start = time.time()
978         self.model_num = int(self.ModelNum_val.get())
979         if self.model_num == 1:
980             self.model = UNet(n_channels=1,
981                               n_classes=1,
982                               BN_is=True,
983                               dw_is=False,
984                               Sigmoid_is=True)
985         else:
986             #repeatNum=8, processing time: about 33ms
987             self.model = UNets(n_channels=1,
988                                n_classes=1,
989                                repeatNum=self.model_num,
990                                BN_is=True,
991                                dw_is=False,
992                                Sigmoid_is=True)
993         self.GetWeightPaths()
994         if self.model_num > len(self.weight_paths):
995             raise ValueError('model_num > len(weight paths)')
996         else:
997             print('OK weight')
998             self.load_weights(self.model, 'UNetBN')
999         self.model.eval()
1000        self.model.to('cuda')
1001        self.model.share_memory()
1002        time_end = time.time()
1003        print('init_seg_model, model_num:{0}, process:{1:.2f}s'\
1004              .format(self.model_num, time_end-time_start))
1005
1006    def GetWeightPaths(self):
1007        target_dir = pathlib.Path('UNet_models/{}'.format(self.segModel_str.get()))
1008        if self.segModel_str.get() == \
1009        '20210605_SensSpecLoss(smooth1e-8)_UNetdw_fluo_newCARS_randomaverage_alltuning':
1010            self.weight_paths = sorted(target_dir.joinpath('test5/valid4').glob('*.pth'))
1011        elif self.segModel_str.get() == \
```

```
        '20210606_SensSpecLoss(smooth1e-8)_UNetdw_fluo_newCARS_randomaverage_fixhalf':
            self.weight_paths = sorted(target_dir.joinpath('test4/valid6').glob('*.pth'))
        elif self.segModel_str.get() == \
        '20210608_SensSpecLoss(smooth1e-8)_UNetdw_newCARS_randomaverage_fromScratch':
            self.weight_paths = sorted(target_dir.joinpath('test5/valid3').glob('*.pth'))
        elif self.segModel_str.get() == \
        '20221126_SensSpecLoss(smooth1e-8)_UNetBN_fluo_newCARS_randAV_randFrame_fixhalf':
            self.weight_paths = sorted(target_dir.joinpath('test1/valid2').glob('*.pth'))
        print('The path num is {}'.format(len(self.weight_paths)))
        for path in self.weight_paths:
            print(path)

    def toCUDA(self):
        '''transport image data to CUDA'''
        temp_array = np.zeros([1, self.pixelNum, self.pixelNum]).astype(np.float32)
        temp_array[0,:,:] = self.img_list[1]/self.img_list[1].max()
        temp_array = np.expand_dims(temp_array, axis=0)
        temp_TF = torch.from_numpy(temp_array.copy())
        self.CudaTensor = temp_TF.to('cuda')

    def segmentation(self):
        '''conduct segmentation'''
        seg_start = time.time()
        output = self.model(self.CudaTensor)
        seg_end = time.time()
        self.ModelProcessingTime_list.append(seg_end-seg_start)

        temp_array = np.zeros([self.pixelNum, self.pixelNum])
        for temp in output:
            temp = temp.to('cpu')
            if self.model_num == 1:
                temp = temp.detach().numpy()[0,:,:]
            else:
                temp = temp.detach().numpy()[0,0,:,:]
            temp[temp>=0.5] = 1
            temp[temp<0.5] = 0
            temp_array += temp.copy()
        temp_array[temp_array<self.model_num/2] = 0
        temp_array[temp_array>=self.model_num/2] = 255
        self.img_list[2] = temp_array.copy()
#         self.img_list[2] = np.random.rand(256 * 256).reshape([256, 256])
        print('processing time:{0:.4f} in self.segmentation'\
              .format(seg_end-seg_start))

```

```
1056     def load_weights(self, model, modelType):
1057         '''load segmentation weights'''
1058         if self.model_num == 1:
1059             print(self.weight_paths[0])
1060 #             model.load_state_dict(torch.load(self.weight_paths[0]))
1061 
1062     def SetCenterPosition(self):
1063         '''set point at center position in image window'''
1064         GS_Xmax = float(self.Xmax_str.get())
1065         GS_Xmin = float(self.Xmin_str.get())
1066         GS_Ymax = float(self.Ymax_str.get())
1067         GS_Ymin = float(self.Ymin_str.get())
1068         if GS_Xmax>5 or GS_Xmin <-5 or GS_Ymax>5 or GS_Ymin <-5:
1069             raise ValueError('this voltage is not supported')
1070         centerX = GS_Xmin + (GS_Xmax-GS_Xmin)/2
1071         centerY = GS_Ymin + (GS_Ymax-GS_Ymin)/2
1072         self.devNum = self.devNum_str.get()
1073 
1074         print('SetCenterPosition, centerX:{}, centerY:{}'.format(centerX, centerY))
1075 
1076         # X axis
1077         WriteConstVoltage(self.devNum, 'ao0', centerX)
1078         # Y axis
1079         WriteConstVoltage(self.devNum, 'ao1', centerY)
1080 
1081 def main():
1082     root = tk.Tk()
1083     app = Application(master=root)
1084     app.mainloop()
1085 
1086 if __name__ == "__main__":
1087     main()
```

List of publications

Original papers

1. **Naoki Yamato**, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto "Improvement of nerve imaging speed with coherent anti-Stokes Raman scattering rigid endoscope using deep-learning noise reduction," Scientific reports **10**(1), 15212 (2020).

2. **Naoki Yamato**, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto "Nerve Segmentation with Deep Learning from Label-Free Endoscopic Images Obtained Using Coherent Anti-Stokes Raman Scattering," Biomolecules **10**(7), 1012 (2020).

Proceedings

1. **Naoki Yamato**, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto, "Near real-time nerve visualization using coherent Raman scattering rigid endoscope and deep learning-based image processing for nerve-sparing surgery," Proc. SPIE, Biomedical Vibrational Spectroscopy 2022: Advances in Research and Industry, 11957, 119570B, (2022).

Review articles

1. 新岡宏彦，**大和尚記**，三宅淳，橋本守, ”AI によるバイオメディカル画像解析と光イメージング装置開発”, O plus E, **42**(4), 517–522 (2020).

2. **大和尚記**，新岡宏彦，三宅淳，橋本守, ” 深層学習による非線形ラマン硬性鏡観察の高速化”, 光学, **50**(6), 259 (2021).

3. **大和尚記**，新岡宏彦，三宅淳，橋本守, ” 深層学習を用いた非線形ラマン散乱硬性内視鏡による神経イメージングの高速化と神経抽出”, 光学, **51**(5), 231–237 (2022).

Invited talks

1. **大和尚記**, 松谷真奈, 工藤信樹, 新岡宏彦, 三宅淳, 橋本守, ” 非線形ラマン散乱硬性内視鏡と深層学習による神経イメージング”, 第 32 回日本内視鏡外科学会総会, パシフィコ横浜会議センター，横浜市，神奈川県, Dec. 05–07 (2019/12/07).

2. **大和尚記**, 松谷真奈, 新岡宏彦, 三宅淳, 橋本守, ” 非線形ラマン散乱硬性内視鏡と深層学習による無染色な末梢神経イメージングの検討”, 日本蛍光ガイド手術研究会第 3 回学術集会, On Line, Oct. 16–17 (2020/10/16).

3. **大和尚記**，新岡宏彦，橋本守, ” 非線形ラマン散乱と深層学習による末梢神経を無染色に可視化する内視鏡外科支援装置の開発ー手術中の神経温存を目指してー”, 日本光学会生体光イメージング産学連携専門委員会キックオフシンポジウム, 佐鳴会館会議室, 静岡大学，浜松市，静岡, Jul. 2 (2022/07/02).

International conferences & symposiums

1. **Naoki Yamato**, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto, "Label-free nerve imaging with anti-Stokes Raman scattering rigid endoscope by improving imaging speed with deep learning," Focus on Microscopy 2020, Due to the COVID-19 (coronavirus) pandemic, the FOM2020 conference was canceled. However, the abstract was reviewed and accepted., Osaka, Japan, Apr. 5-8 (2020) (2020/4/06).

2. **Naoki Yamato**, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto, "Nerve visualization using coherent Raman scattering rigid endoscope with deep learning-based image processing," 2021 International Workshop on New Frontiers in Convergence Science and Technology, Hokkaido University (On-line), Japan, Nov. 5 (2021).

3. Yoshitada Kashimura and Riku Matsuda and **Naoki Yamato**, and Mamoru Hashimoto, "Second-harmonic generation arthroscope with integrated femtosecond Yb fiber laser," Conference on Lasers and Electro-Optics PacificRim2022 (CLEO-PR 2022), Sapporo Convention Center, Sapporo, Japan, July 31 - Aug. 5 (2022) (2022/08/01).

4. **Naoki Yamato**, Hirohiko Niioka, Jun Miyake, and Mamoru Hashimoto, "Fast peripheral nerve imaging with coherent Raman scattering rigid endoscope by noise reduction utilizing deep learning," 25th Congress of the International Commission for Optics (ICO-25) & 16th International Conference on Optics Within Life Science (OWLS-16), Technische Universitat Dresden, Dresden, Germany, Sep. 5-9 (2022) (2022/09/09).

Awards

1. 第 57 回日本生体医光学大会北海道支部大会研究奨励賞
   大和尚記, 新岡宏彦, 橋本守, ” 深層学習を用いた非線形ラマン散乱イメージングの高速化”, 第 57 回日本生体医工学会北海道支部大会, 情報科学研究科北海道大学, 札幌市, 北海道, Oct. 20 (2018/10/20).

2. 令和元年度研究奨励賞・阿部賞
   大和尚記，新岡宏彦，橋本守, ” 非線形ラマン散乱硬性鏡による神経イメージングの転移学習を用いた高速化”, 第 58 回日本生体医工学会大会, 沖縄コンベンションセンター, 宜野湾市, 沖縄, Jun. 06–08 (2019/06/08).

3. 第 55 回応用物理学会北海道支部/第 16 回日本光学会北海道支部合同学術講演会・日本光学会北海道支部学術講演会発表奨励賞
   大和尚記, 新岡宏彦, 三宅淳, 橋本守, ” 非線形ラマン散乱硬性内視鏡のイメージング高速化における深層学習モデルの比較”, 第 55 回応用物理学会北海道支部/第 16 回日本光学会北海道支部合同学術講演会, 北海道大学札幌キャンパス，札幌市，北海道, Jun. 11–12 (2020/01/11).

4. 第 45 会レーザー学会奨励賞
   大和尚記, 松谷真奈, 新岡宏彦, 三宅淳, 橋本守, ” 非線形ラマン散乱硬性内視鏡と深層学習による神経イメージング装置の開発”, レーザー学会第 547 回研究会, On Line, Oct. 9 (2020/10/9).

5. 2022 年度 C&C 若手優秀論文賞
   大和尚記, ”Near real-time nerve visualization using coherent Raman scattering rigid endoscope and deep learning-based image processing for nerve-sparing surgery”, 公益財団法人 NEC C&C 財団, On Line, Jan. 24 (2023/1/24).

Domestic conferences

1. **大和尚記**, 新岡宏彦, 橋本守, ” 畳み込みオートエンコーダを用いたコヒーレントアンチストークスラマン散乱イメージの高画質化”, 第 79 回応用

物理学会秋季学術講演会, 名古屋国際会議場, 名古屋市, 愛知, Sep. 18–21 (2018/09/21).

2. **大和尚記**, 新岡宏彦, 橋本守, ” 深層学習を用いた非線形ラマン散乱イメージングの高速化”, 第 57 回日本生体医工学会北海道支部大会, 情報科学研究科北海道大学, 札幌市, 北海道, Oct. 20 (2018/10/20).

3. **大和尚記**, 新岡宏彦, 橋本守, ” 深層学習を用いた非線形ラマン散乱イメージングの高速化”, 第 16 回医用分光学研究会, フロンティア応用化学研究棟北海道大学, 札幌市，北海道, Nov. 21–22 (2018/11/21).

4. 橋本守, **大和尚記**, 新岡宏彦, ” コヒーレントアンチストークスラマン散乱硬性鏡の開発と神経イメージングへの応用”, 第 16 回医用分光学研究会, フロンティア応用化学研究棟北海道大学, 札幌市，北海道, Nov. 21–22 (2018/11/22).

5. **大和尚記**, 新岡宏彦, 橋本守, ” 深層学習を用いた非線形ラマン散乱イメージングの高速化”, 第 1 回日本メディカル AI 学会学術集会, 国立がん研究センター研究所・新研究棟, 中央区, 東京, Jun. 25–26 (2019/01/26).

6. 新岡宏彦, **大和尚記**, 三宅淳, 橋本守, ” 深層学習を用いた非線形ラマン散乱イメージングの高速化”, レーザ顕微鏡研究会第 44 回講演会・シンポジウム, 銀杏会館, 大阪大学吹田キャンパス, 吹田市, 大阪府, Jul. 04–05 (2019/07/05).

7. 松谷真奈, **大和尚記**, 新岡宏彦, 橋本守, ” 転移学習を用いた深層学習による蛍光画像からの神経抽出”, 日本分子イメージング学会第 14 回学会総会・学術集会, かでる２．７（北海道立道民活動センター）, 札幌, 北海道, May 23–24 (2019/05/24).

8. **大和尚記**，新岡宏彦，橋本守, ” 非線形ラマン散乱硬性鏡による神経イメージングの転移学習を用いた高速化”, 第 58 回日本生体医工学会大会, 沖縄コンベンションセンター, 宜野湾市, 沖縄, Jun. 06–08 (2019/06/08).

9. **大和尚記**，新岡宏彦, 三宅淳, 橋本守, ” 深層学習による CARS 硬性鏡イメージング高速化の評価 - 実時間術中神経イメージングを目指して -”, 第 80 回応用物理学会秋季学術講演会, 北海道大学札幌キャンパス，札幌市，北海道, Sep. 18–21 (2019/09/21).

10. 松谷真奈, **大和尚記**, 新岡宏彦, 工藤信樹, 三宅淳, 橋本守, ” 転移学習を用いた深層学習による非線形ラマン像からの神経セグメンテーション”, 第 80 回応用物理学会秋季学術講演会, 北海道大学札幌キャンパス，札幌市，北海道, Sep. 18–21 (2019/09/21).

11. 松谷真奈, **大和尚記**, 新岡宏彦, 工藤信樹, 三宅淳, 橋本守, ” ２段階転移学習を用いた深層学習による非線形ラマン像からの神経領域抽出”, 第 58 回日本生体医工学会北海道支部大会, 情報科学研究院棟 A21 講義室北海道大学，札幌市，北海道, Oct. 26 (2019/10/26).

12. **大和尚記**，新岡宏彦, 橋本守, ” 非線形ラマン散乱硬性内視鏡と深層学習による末梢神経イメージング”, 第 5 回北海道大学部局横断シンポジウム, 学友会館フラテホール北海道大学，札幌市，北海道, Nov. 06 (2019/11/06).

13. **大和尚記**, 新岡宏彦, 三宅淳, 橋本守, ” 非線形ラマン散乱硬性内視鏡のイメージング高速化における深層学習モデルの比較”, 第 55 回応用物理学会北海道支部/第 16 回日本光学会北海道支部合同学術講演会, 北海道大学札幌キャンパス，札幌市，北海道, Jun. 11–12 (2020/01/11).

14. **大和尚記**, 新岡宏彦, 橋本守, ” ノイズ除去のアンサンブル学習による非線形ラマン硬性内視鏡神経イメージングの高速化”, 第 59 回日本生体医工学会大会, On Line, May 25–27 (2020/05/27).

15. **大和尚記**, 松谷真奈, 新岡宏彦, 三宅淳, 橋本守, ” 非線形ラマン散乱硬性内視鏡と深層学習による神経イメージング装置の開発”, レーザー学会第 547 回研究会, On Line, Oct. 9 (2020/10/9).

16. 松田陸, **大和尚記**, 橋本守, ” 神経を可視化するコヒーレントアンチスト

ークスラマン散乱硬性内視鏡の可搬化のためのファイバーレーザー光源の開発”, 第 59 回日本生体医工学会北海道支部大会, On Line, Oct. 24 (2020/10/24).

17. **大和尚記**，松谷真奈，新岡宏彦，三宅淳，橋本守, ”CARS 硬性内視鏡の神経イメージングと深層学習による神経抽出ー実時間イメージングに向けた短時間露光神経画像の画像解析ー”, 第 68 回応用物理学会春季学術講演会, On Line, Mar. 16–19 (2021/03/19).

18. **大和尚記**，松谷真奈，新岡宏彦，三宅淳，橋本守, ” 非線形ラマン散乱硬性内視鏡と深層学習による実時間神経抽出”, 第 60 回日本生体医工学会大会・第 36 回日本生体磁気学会大会, On Line, Jun. 15–17 (2021/06/17).

19. **大和尚記**，新岡宏彦，三宅淳，橋本守, ” 術中の末梢神経ナビゲーションシステムを目指した非線形ラマン散乱硬性内視鏡と深層学習による末梢神経イメージング”, 第 7 回北海道大学部局横断ジンポジウム, On Line, Oct. 1 (2021/10/01).

20. **大和尚記**，新岡宏彦，三宅淳，橋本守, ” 非線形ラマン散乱硬性内視鏡と深層学習による神経イメージング装置の開発”, 一般社団法人レーザー学会学術講演会第 42 回年次大会, On Line, Jun. 12–14 (2022/01/13).

21. **大和尚記**, 松田陸, 橋本守, ” 無染色にコラーゲンを可視化する第二高調波発生関節鏡の開発”, 第 61 回日本生体医工学会北海道支部大会, On Line, Oct 29 (2022/10/29).