International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 10, Number 2, pages 259–276, July 2020

CNN Architecture for Surgical Image Segmentation with Recursive Structure and Flip-Based Upsampling

Taito Manabe, Koki Tomonaga, Koki Fujita, and Yuichiro Shibata
Graduate School of Engineering, Nagasaki University
1-14 Bunkyo-machi, Nagasaki, 852-8521, Japan
E-mail: {manabe, tomonaga, kfujita, shibata}@pca.cis.nagasaki-u.ac.jp

Taiichiro Kosaka and Tomohiko Adachi
Graduate School of Biomedical Sciences, Nagasaki University
1-7-1 Sakamoto, Nagasaki, 852-8520, Japan

Received: February 14, 2020
Revised: May 4, 2020
Accepted: June 1, 2020
Communicated by Kouzou Ohara

Abstract

Laparoscopic surgery, a less invasive camera-aided surgery, is now performed commonly. However, it requires a camera assistant who holds and maneuvers a laparoscope. By controlling the laparoscope automatically using a robot, a surgeon can perform the operation without a camera assistant, which would be beneficial in areas suffering from lack of surgeons. In this paper, a prototype image segmentation architecture based on a convolutional neural network (CNN) is proposed to realize an automated laparoscope control for cholecystectomy. Since a training dataset is annotated manually by a few surgeons, its scale is limited compared to common CNN-based systems. Therefore, we built a recursive network structure, with some sub-networks which are used multiple times, to mitigate overfitting. In addition, instead of the common transposed convolution, the flip-based subpixel reconstruction is introduced into upsampling layers. Furthermore, we applied stochastic depth regularization to the recursive structure for better accuracy. Evaluation results revealed that these improvements bring better classification accuracy without increasing the number of parameters. The system shows a throughput sufficient for real-time laparoscope robot control with a single NVIDIA GeForce GTX 1080 GPU.

Keywords: laparoscopic surgery, semantic segmentation, CNN, recursive structure

1 Introduction

Laparoscopic surgery, or laparoscopy, is an operation performed through small incisions using a fiber optic camera system called a laparoscope. It is less invasive than the common open surgery and also known as minimally invasive surgery. Since it brings various advantages such as reduced pain and shorter recovery time to the patient, it has become more and more common these days.

In the laparoscopic surgery, a camera assistant who holds and maneuvers a laparoscope is essential as well as a surgeon. If a laparoscope can be controlled automatically using a robot, a surgeon can perform the operation without a camera assistant, which would be helpful in areas where there are not sufficient surgeons, such as small remote islands. To automate the camera control, a system to decide an appropriate camera angle depending on an input image is required, in addition to the robot itself. There are various potential

algorithms, one of which is the use of a semantic segmentation system to recognize position and distance of each organ in the image.

In this paper, a prototype image segmentation system architecture for laparoscopic cholecystectomy, the most common type of laparoscopic surgery, is proposed. The system is based on a convolutional neural network (CNN) trained with a training dataset annotated manually by a few surgeons. At present, however, scale of the dataset is limited because a surgical image segmentation is a highly-specialized task which requires a close cooperation with experts. This causes an overfitting problem, which means that the network fits the training dataset excessively and does not work properly in an actual environment. Though the dataset has been improved compared to our previous work [22], it is still smaller than commonly expected. For example, the VOC2012 dataset [14] contains 11530 images, though the average image size is somehow smaller than our dataset. Therefore, along with improving the dataset, we devised a network structure to mitigate the problem. The main contributions of this paper include:

- Recursive network structure and flip-based subpixel reconstruction improve classification accuracy without increasing the number of parameters. The former means that some sub-networks are reused multiple times, and the latter can be used as a substitution for the common transposed convolution.
- In addition to our previous work [22], stochastic depth regularization [7] is applied to the recursive network structure and shows improvement in classification accuracy. The addition of PCA color augmentation [2] to basic online data augmentation is also tested, which lowers the peak accuracy for the current dataset but results in more consistent increase in accuracy.
- The system operating with a single NVIDIA GeForce GTX 1080 GPU shows a throughput of 17 fps, which is sufficient for real-time laparoscope robot control.

The rest of the paper is organized as follows. First, we show some related works in Section 2. Then a detailed network architecture of the system is introduced in Section 3. Section 4 explains a procedure to train the network and Section 5 shows evaluation results on classification accuracy and inference performance. Finally, some conclusions and future work are described in Section 6.

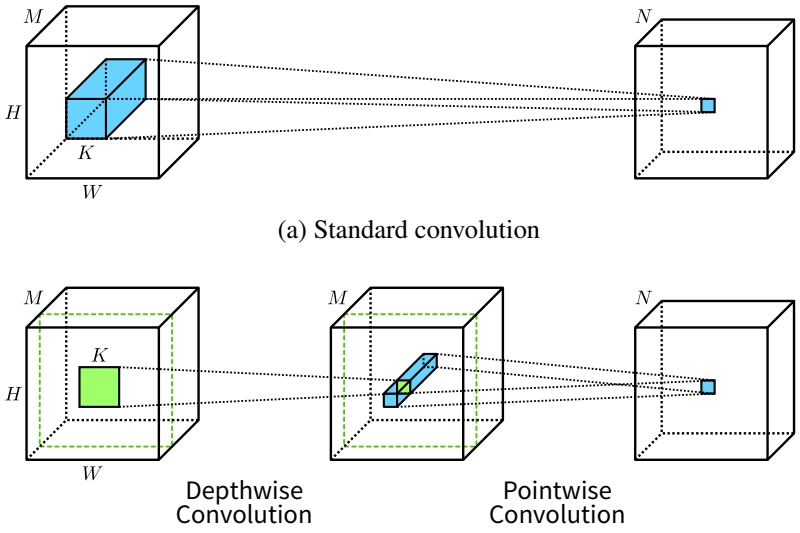
2 Related Work

Various architectures for semantic segmentation based on a convolutional neural network have been proposed to date. Fully Convolutional Network (FCN) [11], proposed by J. Long, et al., is characterized by its network structure without any fully connected layers. FCNs use multiple pooling layers to obtain global features, while maintaining local detailed features using skip connections. To improve memory efficiency of FCN, V. Badrinarayanan, et al. proposed SegNet [24], which uses upsampling based on pooling indices to avoid holding the whole of intermediate feature maps.

O. Ronneberger, et al. presented U-Net [17], a CNN for biomedical image segmentation, with an encoder-decoder structure including a contracting path and a symmetric expanding path for considering both global context and local information. H. Zhao, et al. introduced a pyramid pooling module for context aggregation [9]. The module applies convolution operations with multiple pooling sizes simultaneously to feature maps which are previously extracted using ResNet [12] with dilated convolutions [6]. Results are upsampled to the original size, concatenated and then converted into the final feature map. In this way, global context and local detail can be aggregated. The whole system is called Pyramid Scene Parsing Network (PSPNet).

As well as improvements in network structure, some attempts to reduce computational complexity and a parameter size, both of which are major obstacles to wide utilization of neural networks, have been made. For example, Xception [5] and MobileNet [1] adopted depthwise separable convolutions to build light-weight neural networks. In this strategy, a traditional convolution layer is divided into depthwise and pointwise (1×1) convolutions for efficient feature extraction. Since pointwise convolutions tend to be a bottleneck of separable convolutions, further improvements including ShuffleNet [28] and MobileNetV2 [15] have been proposed. In these, a network size is reduced by approximating a pointwise convolution using multiple smaller pointwise convolutions based on a technique such as grouped convolutions.

Many of the existing segmentation systems including those referred above depend on a large training dataset. However, such a dataset requires a lot of cost and effort to prepare and thus is not necessarily



(a) Standard convolution

(b) Separable convolution

Figure 1: Comparison between standard and separable convolutions

available. An insufficient dataset causes a network to overfit it and compromises utility of the system. In this paper, we devise a network architecture to lessen a degree of overfitting and evaluate its performance.

3 Architecture

3.1 Depthwise Separable Convolution

Depthwise separable convolution (abbreviated as SepConv in this paper) is a technique to divide one convolution layer into depthwise and pointwise convolution layers as shown in Figure 1, on the assumption that a correlation in space and one in channels (depth) are independent and can be isolated. As shown in [5] and [1], introducing separable convolutions reduces computational complexity and a parameter size of CNNs greatly, without causing serious deterioration in inference accuracy.

Given that a filter kernel size is $K \times K$ and numbers of input and output channels are M and N, respectively, a common convolution layer (Figure 1 (a)) contains K^2MN filter coefficients. With SepConv (Figure 1 (b)), the number decreases to $K^2M + MN = M(K^2 + N)$. K^2M is for the depthwise and MN is for the pointwise. Therefore, if N is far larger than K, which is common for most large-scale CNNs, the pointwise convolution would be a dominant part of the whole parameters. [28] and [15] show that decomposing the pointwise convolution is effective in this case. For example, a grouped convolution technique combined with channel shuffle is used in [28]. If the pointwise convolution layer is divided into G groups, the number of parameters can be decreased from MN into MN/G. Note that, because of the channel shuffle, the computational cost does not decrease in the same way. In our implementation at this time, however, N is relatively small ($K = 3,\ N = 20$) due to a small-scale training dataset. So we deem that a plain SepConv is sufficient.

After the pointwise convolution, Leaky ReLU [3] is applied for activation in this implementation. Leaky ReLU is a non-linear function which is defined as follows:

$$f(x) = \max(ax,\ x) \quad (0 < a < 1) \tag{1}$$

Compared to common ReLU [25][27][29] $f(x) = \max(0,\ x)$, Leaky ReLU has a non-zero slope in its negative domain, which mitigates the gradient vanishing problem in a deep neural network. The slope a is currently set to $2^{-2} = 0.25$. No activation function exists between the depthwise and pointwise convolutions.

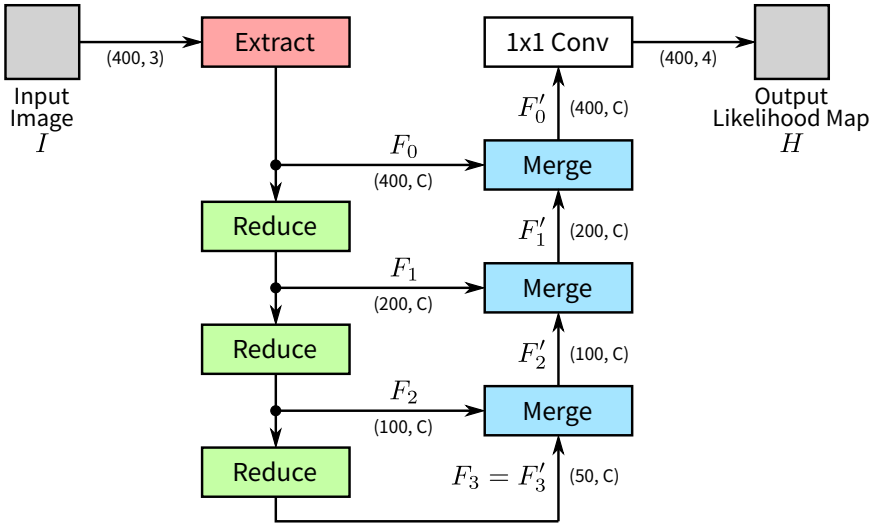


Figure 2: Network overview where the recursion level $L = 3$. Each of the two sub-networks, *Reduce* and *Merge*, is used 3 times

3.2 Network Structure

As the basis of the system, we adopted the encoder-decoder model as seen in [24] and [17]. The network is composed of 3 sub-networks which are named *Extract*, *Reduce*, and *Merge* in addition to the output 1×1 convolution layer, as shown in Figure 2. A notation (S, C) in the figure indicates that a feature map size is $S \times S$ and the number of channels is C. In order to mitigate overfitting problem caused by a small dataset, a recursive structure is introduced into the network. That is, two of the sub-networks, *Reduce* and *Merge*, are reused multiple times. We use the term "recursion level" L (= 1, 2, ...) to mean how many times each of the two is used. L can be specified as a parameter.

Firstly, an input RGB image passes through *Extract* and is converted into a C-channel feature map. Though a size of the input image is 400×400 in Figure 2, which is a size used for training as explained later in Section 4, the system can accept any size S such that $S = 2^L k$ ($k \in \mathbb{N}$) since it does not contain any fully-connected layers. *Reduce* has the function of reducing the feature map of size $2S \times 2S$ into $S \times S$. *Reduce* is applied L times, helping to take account of the global context of the input image. Let F_n ($0 \leq n \leq L$) be the feature map output from the n-th *Reduce*, with the proviso that F_0 is the output from *Extract*. If we name the input image I, this can be formulated as follows:

$$F_0 = \text{Extract}(I) \tag{2}$$

$$F_{i+1} = \text{Reduce}(F_i) \quad (0 \leq i \leq L-1) \tag{3}$$

Next, *Merge* takes F_L and F_{L-1}. The former is firstly upsampled inside and concatenated with the latter. Then the feature map of $2C$ channels are merged into C channels through convolution layers. The output is then merged again with F_{L-2}. By applying *Merge* L times in this way, all of the *Extract* and *Reduce* outputs (F_0, F_1, ... F_L) are merged into one feature map. Given that the output of the n-th *Merge* is F'_{n-1} with the proviso that $F'_L = F_L$, this procedure can be summarized as follows:

$$F'_L = F_L \tag{4}$$

$$F'_i = \text{Merge}(F_i,\ F'_{i+1}) \quad (0 \leq i \leq L-1) \tag{5}$$

The merged feature map F'_0 is converted into an X-channel likelihood map H using the 1×1 (pointwise) convolution layer at the end of the network. X corresponds to the number of all classes for segmentation. As explained later in Section 4.1, X is set to 4 in this implementation. Finally a class of pixel at (x, y) in the input image is predicted as follows:

$$\text{class}(I(x,\ y)) = \text{argmax}(H(x,\ y)) \tag{6}$$

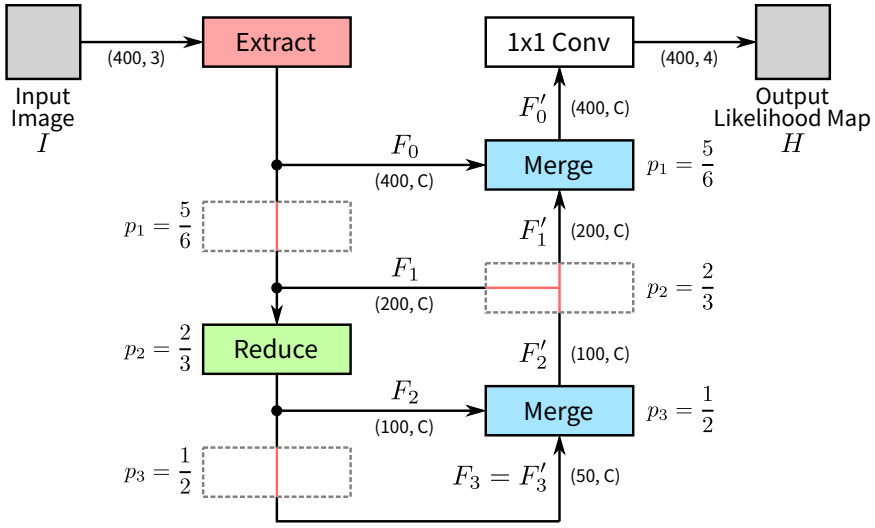


Figure 3: Random sub-network bypass with stochastic depth in the training process

Because of the recursive network structure explained above, *Reduce* and *Merge* sub-networks will naturally learn how to handle feature maps in multiple sizes without additional parameters. Combined with online data augmentation such as random rotation and resize shown in Section 4.2, this enables the network to be more versatile with restrained risk of overfitting. The relationship between the recursion level and classification accuracy is evaluated later in Section 5.1.

Furthermore, we applied stochastic depth regularization, proposed by G. Huang, et al. in [7], to our recursive network for further mitigation of overfitting. This is a training procedure which bypasses each of layers with certain probability depending on its depth. In this implementation, we regard each of the two sub-networks, *Reduce* and *Merge*, as a unit to be bypassed. A probability of the n-th *Reduce* or *Merge not* being skipped, p_n, is defined as follows:

$$p_n = 1 - \frac{n}{L}(1 - p_L) \quad (1 \leq n \leq L) \tag{7}$$

where p_L is a hyperparameter. We set p_L to 0.5 this time, so $p_n = 1 - n/(2L)$.

Figure 3 illustrates how stochastic depth works in the training process. The skipped sub-networks do nothing except for up/downsampling to adjust the size of feature maps. Detailed explanation will be given in Section 3.3. Stochastic depth works as a regularization mechanism similarly to Dropout [16], preventing the network from overfitting. In addition, stochastic depth speeds up training process since it reduces a substantial recursion level. During inference, on the other hand, scaling by p_n is applied instead of the stochastic bypass. Hence segmentation behavior stays deterministic. We will compare classification accuracy with or without stochastic depth in Section 5.1.

3.3 Sub-Networks

Figure 4 is a block diagram depicting the structure of each sub-network. *Extract* has 4 convolution layers. The first layer is a standard 3×3 convolution followed by Leaky ReLU, since the number of input channels (M in Figure 1) is no more than 3 (RGB) and thus it does not yield remarkable increase in the number of parameters. Let C be the number of output channels (N). Then the first layer has $3^2 \times 3 \times C = 27C$ parameters. The rest are 3×3 SepConv layers with $M = N = C$, each of which retains $C^2 + 9C$ parameters. Appropriate zero padding precedes every 3×3 convolution in the network so that input and output feature maps have the same size. This padding makes the concatenation in *Merge* easier.

Following all of the convolutions in each of the sub-networks including *Reduce* and *Merge*, Batch Renormalization (BRenorm) [20] is applied. It is an extended version of Batch Normalization [21] aiming at improved effectiveness for training mini-batches which are small or composed of dependent samples. This

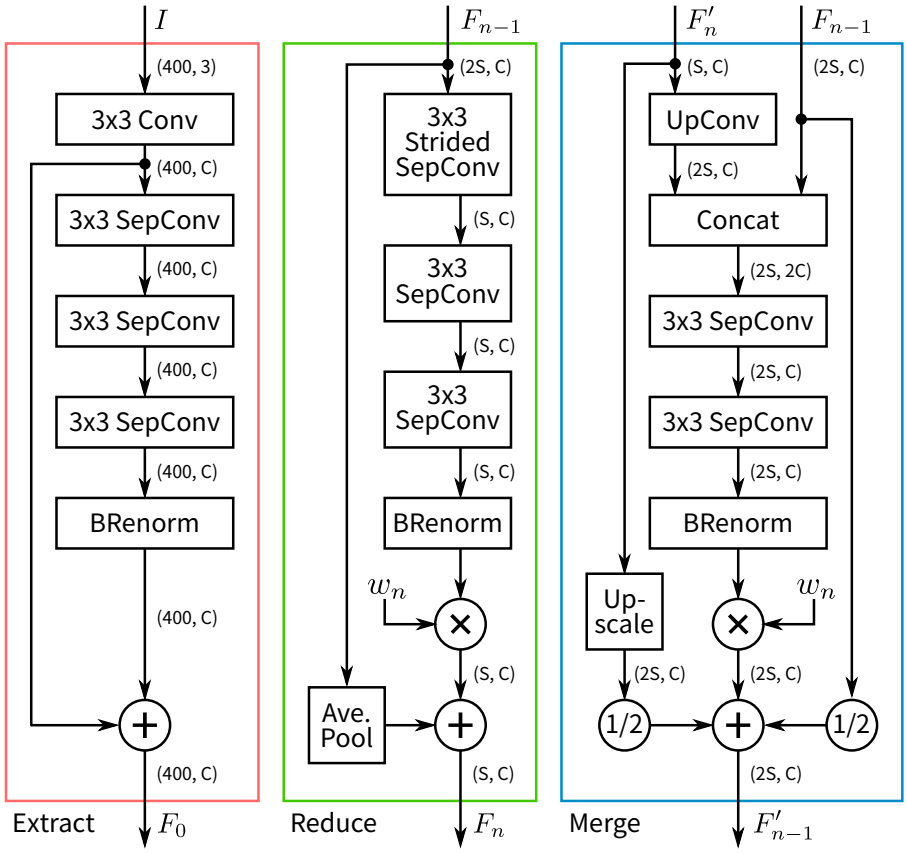


Figure 4: Structures of the sub-networks

characteristic is suitable for the dataset we use this time, whose details are shown in Section 4.1. Each sub-network also contains a shortcut connection [12], as shown in Figure 4, to let gradients travel more smoothly at the backpropagation process.

Reduce consists of 3 SepConv layers. In the first layer, depthwise convolutions are performed with stride of 2 (Strided SepConv) to downsample an input feature map F_{n-1}. The number of parameters for each of the three layers is $C^2 + 9C$ as well. The stride has no relation to this value. The shortcut connection in *Reduce* contains a 2×2 average pooling layer (Ave. Pool) for downsampling. Before the skipped input is added, the result after Batch Renormalization is multiplied by a weight value w_n. This is for implementing stochastic depth regularization, and $w_n = 1$ when it is not used. Otherwise, w_n is defined as follows. If x is a random value sampled from the standard uniform distribution $U(0,\ 1)$, w_n in the training process is:

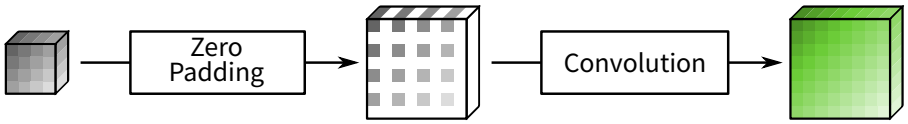
$$w_n = \begin{cases} 1 & (x \leq p_n) \\ 0 & (\text{otherwise}) \end{cases} \tag{8}$$

where p_n is the probability of *not* being skipped, shown in Eq. (7). With $w_n = 0$, the output is just a downsampled input F_{n-1} and all convolution and batch renormalization layers have no effect. In an actual implementation, they are simply disabled and no calculations are done in them. For inference, on the other hand, w_n is equal to p_n as explained in Section 3.2.

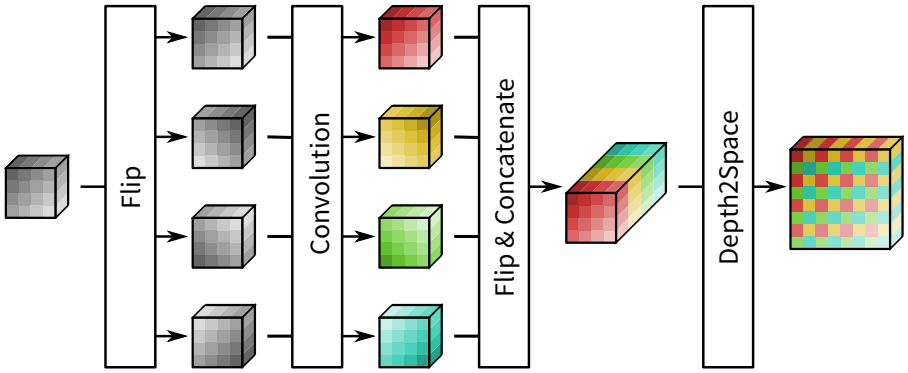
Unlike the other sub-networks, *Merge* has two inputs F'_n and F_{n-1} with different sizes. To the smaller one (F'_n), an upsampling convolution (UpConv) layer based on SepConv is applied so that it can be concatenated with the larger one (F_{n-1}). For the UpConv layer, we evaluate 2 upsampling methods described in Section 3.4. In either case, the number of parameters is the same: $C^2 + 9C$. The concatenated feature maps ($2C$ channels) are then merged into C channels with 2 SepConv layers, which have $2C^2 + 18C$ and $C^2 + 9C$ parameters, respectively. *Merge* also has skip connections, which adds an average of F_{n-1} and upscaled F'_n

Table 1: Number of parameters (filter coefficients) in the network

Sub-Network	# of parameters	
	With SepConv	W/O SepConv
Extract	$3C^2 + 54C$	$27C^2 + 27C$
Reduce	$3C^2 + 27C$	$27C^2$
Merge	$4C^2 + 36C$	$36C^2$
1×1 Conv	$4C$	$4C$
Total	$10C^2 + 121C$	$90C^2 + 31C$



(a) Transposed convolution



(b) Flip-based sub-pixel reconstruction

Figure 5: Comparison of two upsampling convolution methods

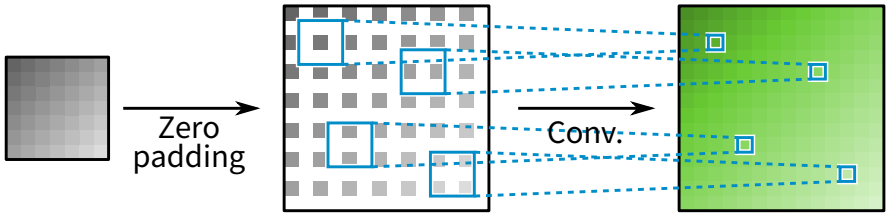
to the convolution result. The multiplication by w_n for stochastic depth is the same as that in *Reduce*.

Total number of parameters in the whole network is $10C^2 + 121C$, as summarized in Table 1. It should be noted that the number is not affected by the recursion level L or the use of stochastic depth. Without SepConv, the number will increase to $90C^2 + 31C$. If $C = 20$, the systems with and without SepConv have 6420 and 36620 parameters, respectively. This means that the number of parameters as well as theoretical computational complexity is reduced by about 82.5%.

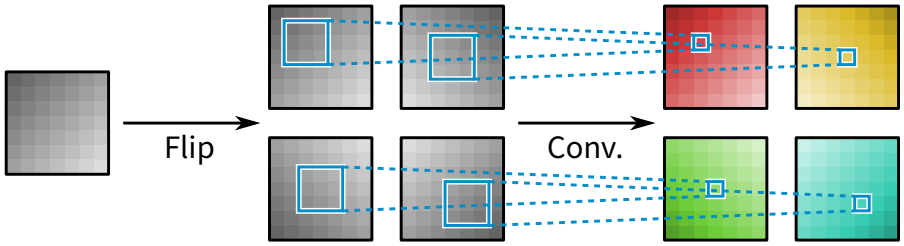
3.4 Flip-Based Subpixel Reconstruction

To perform convolution with upsampling (UpConv) in *Merge*, we consider two methods illustrated in Figure 5: transposed convolution and convolution with flip-based sub-pixel reconstruction. The former is also known as deconvolution or fractionally strided convolution, and is widely used for upsampling layers of CNNs. In brief, this is just a standard convolution preceded by image extension using zero padding or some other algorithm as shown in Figure 5 (a).

The latter, on the other hand, is based on the super-resolution algorithm proposed in [23]. Given that an input feature map has a shape of $(C,\ H,\ W)$, where C, H, and W are the number of channels, height, and width, respectively. At first, the input feature map is flipped in 3 ways: horizontally, vertically, and along both axes. Then each of the original and the flipped feature maps passes through the identical convolution layer. Every result is flipped again in the same way as its origin, and all the flipped results are concatenated along a channel axis. Finally, the concatenated feature map F_{in} of shape $(4C,\ H,\ W)$ is reshaped into F_{out} $(C,\ 2H,\ 2W)$ using



(a) Transposed convolution



(b) Flip-based sub-pixel reconstruction

Figure 6: Number of pixels to be convolved with each upsampling method

the following *depth to space* transformation:

$$F_{\text{out}}(c,\, v,\, h) = F_{\text{in}}\left(c + dC,\, \left\lfloor \frac{v}{2} \right\rfloor,\, \left\lfloor \frac{h}{2} \right\rfloor\right) \tag{9}$$

$$d = (v \bmod 2) \times 2 + (h \bmod 2) \tag{10}$$

where $F(c,\, v,\, h)$ means a pixel value at the coordinates $(v,\, h)$ of channel c. These procedures are graphically shown in Figure 5 (b).

This transformation technique is also known as pixel shuffler, proposed as a part of the CNN-based super-resolution model called ESPCN [26]. However, in [26], a CNN has $4C$ output channels and converts the input feature map of shape $(C,\, H,\, W)$ into the output of shape $(4C,\, H,\, W)$ directly. Our flip-based method, on the other hand, reduces the number of output channels from $4C$ to C by utilizing the symmetricalness within the 4 channels, as explained in [23]. This means that the number of parameters can be reduced.

For both upsampling methods, the same 3×3 SepConv is used. Therefore, the number of parameters does not vary. However, with transposed convolution, the number of pixels to be convoluted is limited to 1, 2, or 4 pixels depending on coordinates, instead of 9 pixels expected from the filter size, as shown in Figure 6 (a). This is caused by the zero padding prior to the convolution. In contrast, flip-based sub-pixel reconstruction can make use of all the 9 pixels as Figure 6 (b) shows. Though the convolution operations are performed for 4 feature maps with sub-pixel reconstruction, total computational complexity is almost the same since each input feature map has quarter pixels compared to the transposed convolution. In Section 5.1, we show comparison results between the two methods.

4 Training

4.1 Dataset

Our segmentation system is intended to support cholecystectomy using laparoscope by classifying each pixel in an input image into 4 classes: background/tools (black), gallbladder (blue), cystic duct (green), and common bile duct (red). Each of the classes is visualized using the color shown in the parentheses. The training dataset to train the network described in Section 3 and the validation dataset to validate classification performance are composed of 138 and 45 image pairs, respectively. Each pair contains an input image and a

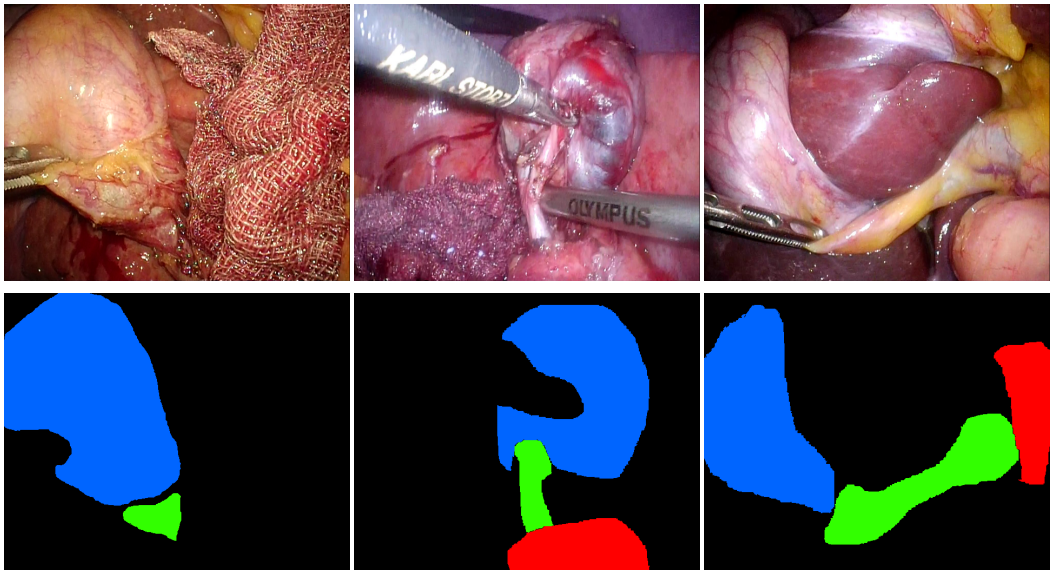


Figure 7: Examples of images in a dataset. Input images and label images are shown in the first and second rows, respectively

Table 2: Distribution of pixels corresponding to each class

Class	Color	Training	Validation	Total
Background	Black	65.46%	64.97%	65.34%
Gallbladder	Blue	21.37%	22.35%	21.61%
Cystic duct	Green	4.96%	5.49%	5.09%
CBD	Red	8.21%	7.18%	7.96%
# of image pairs		138	45	183

supervisory label image (ground truth) as shown in Figure 7. An image size of all the images is 640×512. To have the *Merge* sub-networks work properly, both width and height must be the multiple of 2^L.

All of the input images are captured from the moving images (videos) of actual laparoscopic cholecystectomy performed on 57 patients, and the corresponding label images are manually annotated by a few surgeons. Each pixel of the latter has a value which corresponds to one of the classes as stated above. Since the annotation process takes time and the images must be treated with great care because of privacy concern, a scale of the dataset is quite limited at the moment in comparison with common datasets for deep neural networks, where more than tens of thousands of pairs are commonly required. While the devices for the network explained in Section 3 are introduced to mitigate overfitting, we are also working on continuous improvement of the scale simultaneously.

As is often the case with real-world datasets, there is major class imbalance in the dataset. Distribution of the classes is as listed in Table 2, which shows that the background pixels cover approximately 65% of the all images while the cystic duct pixels cover less than 6%. In evaluating classification accuracy, this imbalance must be taken into account, as we will describe later in Section 5.1.

4.2 Tools and Procedures

To define and train the network, we use Chainer [19] 5.3.0, a Python-based framework for neural networks. Image processings for training and validation, such as rescaling and rotating images, are performed with OpenCV-Python [18] 4.0.1.24, a popular open-source library for computer vision.

Training procedures are as follows. At first, 2 image pairs are read from the training dataset in turns. Then 3 types of modification as shown below are applied to the images for data augmentation.

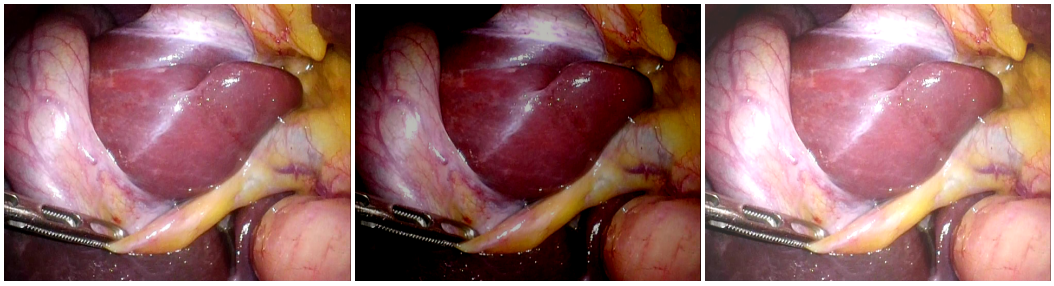


Figure 8: Examples of PCA color augmentation results

- Random resize: Images are rescaled using `cv2.resize()` by a random factor sampled from the uniform distribution $U(2/3, 4/3)$, with Lanczos4 [13] interpolation for input images and nearest neighbor interpolation for label images.

- Random rotation: Images are rotated using `cv2.warpAffine()` by a random angle sampled from $U(0, 2\pi)$, with linear interpolation for input images and nearest neighbor interpolation for label images. Pixel values outside the original image are set to 0.

- Random horizontal flip: Images are flipped horizontally with a probability of 0.5.

A combination of horizontal and vertical flip is equivalent to a rotation by π, so the random flip is performed only in a horizontal axis. Since these data augmentation procedures are online, in other words, they are applied not in advance but in parallel with the training process, data size of the dataset is not affected and thus the whole dataset can easily be kept on memory, leading to faster training. It also enables to use random floating-point scale factor and angle for more effective augmentation, though it gets impossible to determine how many images the dataset is equivalent to after the augmentation.

As additional data augmentation, we also tried combining PCA color augmentation [2] for some of the tested configurations shown in Table 3. This is a technique to apply random color shift to input images based on principal component analysis (PCA). In advance, we calculate the 3×3 covariance matrix of RGB pixel values of each input image, whose eigenvectors $\boldsymbol{p}_i$ and eigenvalues λ_i ($1 \leq i \leq 3$) are then calculated. Every time the image is used for training, the following color shift value, based on 3 random values α_i drawn from the normal distribution with $\mu = 0$ and $\sigma = 0.1$ ($N(0, 0.1^2)$), is added to it:

$$(\Delta_r,\ \Delta_g,\ \Delta_b)^T = (\boldsymbol{p}_1,\ \boldsymbol{p}_2,\ \boldsymbol{p}_3)(\alpha_1\lambda_1,\ \alpha_2\lambda_2,\ \alpha_3\lambda_3)^T \tag{11}$$

Since we can pre-compute $\boldsymbol{p}_i$ and λ_i for each image, online PCA color augmentation can be done relatively easily at the cost of increased memory usage. Some examples of PCA color augmentation are shown in Figure 8. Intensities of RGB channels are changed without hurting the original color balance. To realize autonomous laparoscope control, the system must handle various lighting conditions since a lamp is placed on the tip of a moving laparoscope. In such situations, it is possible that PCA color augmentation brings more stable segmentation results.

Following the data augmentation procedures above, small images of size 400×400 are cut out from the interpolated images. A cut-out position is decided randomly for each pair. In this way a mini-batch containing 2 pairs of input and label images is generated and given to the network. Finally, a softmax cross-entropy loss is backpropagated to update parameters. An optimization function is Adam [4] with the default parameters ($\alpha = 0.001,\ \beta_1 = 0.9,\ \beta_2 = 0.999,\ \epsilon = 10^{-8}$). Once all the images in the training dataset have been used, they are permutated before the next iteration begins.

Table 3: Evaluated system configurations

Name	UpConv	Not Recursive	C	L	Stochastic Depth	PCA Color Augmentation
Flip L1	Flip	-	20	1	-	-
Flip L2		-	20	2	-	-
Flip L2+SD		-	20	2	yes	-
Flip L4		-	20	4	-	-
Flip L4+SD		-	20	4	yes	-
Flip L4+CA		-	20	4	-	yes
Flip L4+SD+CA		-	20	4	yes	yes
Flip30 L4+SD		-	30	4	yes	-
FlipNR L4+SD		yes	20	4	yes	-
Trans L1	Trans	-	20	1	-	-
Trans L2+SD		-	20	2	yes	-
Trans L4+SD		-	20	4	yes	-

5 Evaluation

5.1 Accuracy

As an evaluation criterion, we use mean accuracy of classification calculated as follows:

$$\text{Mean accuracy} = \frac{1}{X}\sum_{i=0}^{X-1}\frac{n_i}{t_i} \tag{12}$$

where $X = 4$ is the number of classes, t_i is a total number of pixels which belong to class i in the ground truth, and n_i is the number of pixels in class i which are successfully predicted to be in class i. Unlike pixel accuracy $(\sum n_i)/(\sum t_i)$, which tends to be affected excessively by some classes which cover large areas, mean accuracy can reflect performance for minor classes. For example, 65.0% of pixels in the used validation dataset belong to the background class as shown in Table 2. Because of this, if the system predicts that the whole image is the background, pixel accuracy is evaluated to 65.0%, which is far higher than the expected result. In contrast, using mean accuracy in this case provides more reasonable results: 25.0%.

To evaluate effectiveness of devised architectures shown in Section 3 and PCA color augmentation shown in Section 4, 12 network configurations listed in Table 3 are evaluated. Note that "Flip" and "Trans" in the "UpConv" column mean flip-based subpixel reconstruction and transposed convolution in Section 3.4, respectively. C and L are the number of channels and the recursion level, respectively. Stochastic depth and PCA color augmentation are abbreviated as "SD" and "CA" in the configuration names. $C = 20$ is used for all the configurations excluding the Flip30 L4+SD ($C = 30$). In addition, for comparison between our network model and common non-recursive models, the FlipNR L4+SD configuration, where all the *Reduce* and *Merge* sub-networks are independent, is included in the tested configurations. During the training process described in Section 4, classification accuracy is evaluated using the validation dataset without cropping, on every completion (iteration) of 1000 mini-batches. For example, if we say that 5 iterations have finished, it means that 5000 mini-batches (i.e. 10000 image pairs of size 400×400) have been fed to the network to update the parameters.

At first, we compare the Flip configurations using multiple recursion levels with or without stochastic depth regularization (Flip L1, L2, L2+SD, L4, L4+SD). In Figure 9, a transition of accuracy through the training process with each configuration is shown. For better visibility, we use approximate curves. From the figure, it can be said that both higher recursion level L and the use of stochastic depth lead to better peak accuracy. Receptive field sizes corresponding to L are summarized in Table 4. Since the network has multiple paths from an output to an input, the longest path is chosen for calculation so that the size is maximized. As shown in Table 4, receptive field size is almost doubled every time L is incremented by 1. Though L does not affect the number of parameters, higher L brings a larger receptive field size, which enables the system to consider global context and perform better classification. Applying stochastic depth has no impact on the

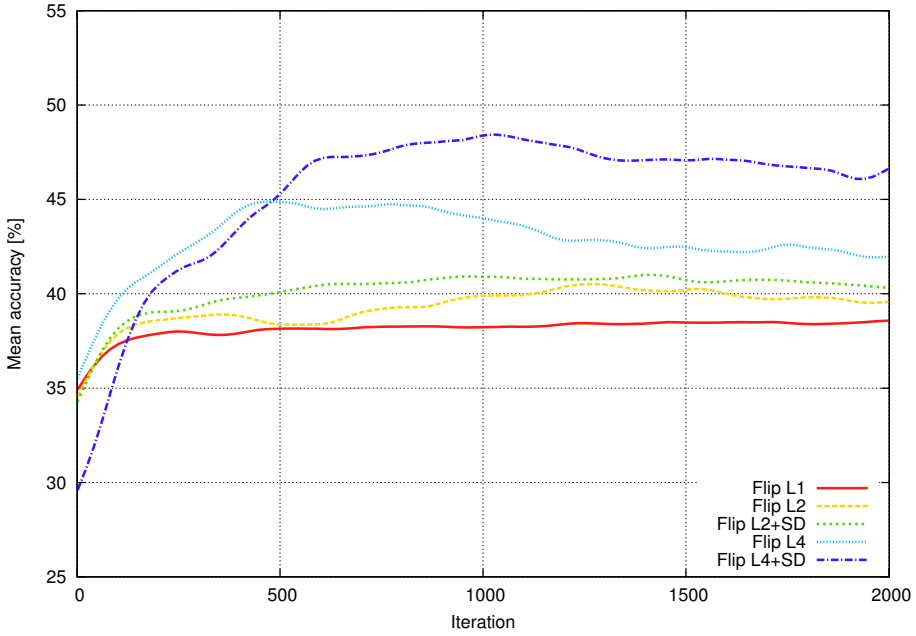


Figure 9: Mean accuracy transition of Flips with multiple recursion levels

Table 4: Receptive field size of each L

L	1	2	3	4
Flip	27	63	135	271
Trans	25	55	119	239

size since sub-network bypass does not occur in the inference process.

With higher recursion level L, accuracy tends to reach its peak earlier and then start decreasing. Figure 9 shows this decrease is conspicuous especially with the $L = 4$ configurations. Even after the peak, an increase in training loss was not observed, so this behavior indicates the possibility of overfitting. When $L = 1$, on the other hand, there seems to be no sign of overfitting. The possible reason is because the receptive field size is too small for the network to overfit the training dataset. Nevertheless, accuracy of $L = 1$ is the worst and thus there is little reason to adopt this recursion level.

Next comes the evaluation on how PCA color augmentation works. In Figure 10, comparison of accuracy transition for Flip L4 and L4+SD with or without PCA color augmentation is shown. From the figure, it can be said that the augmentation lowers accuracy of both L4 and L4+SD configurations, even though accuracy decrease, indicating overfitting, is not seen with it. This indicates that the network somewhat depends on particular color to recognize the corresponding organ, which is possible because the dataset is derived from videos of actual laparoscopic surgeries under appropriate distance and lighting control, and PCA color augmentation makes the task more difficult by disturbing color. For automatic laparoscope control, however, the system must handle more complicated situations. For example, a color tone of organs varies from person to person and a lighting condition is not constant since a lamp is typically placed on a laparoscope which moves around. Under such situations, it is possible for PCA color augmentation to work well. To confirm this, we need larger-scale dataset with a diversity of images.

To evaluate how the recursive network structure as well as the number of channels C works, L4+SD configurations of Flip (6640 parameters), Flip30 (12630 parameters), and FlipNR (18600 parameters) are compared in Figure 11, which shows that Flip L4+SD configuration outperforms the others even though it contains the least parameters of the three. Flip30 and FlipNR reach their peaks earlier and then start decreasing. Though FlipNR is more stable than Flip30, its accuracy is still inferior to Flip. This quality reversal indicates overfitting. For confirmation, comparison of their accuracy transition using the training

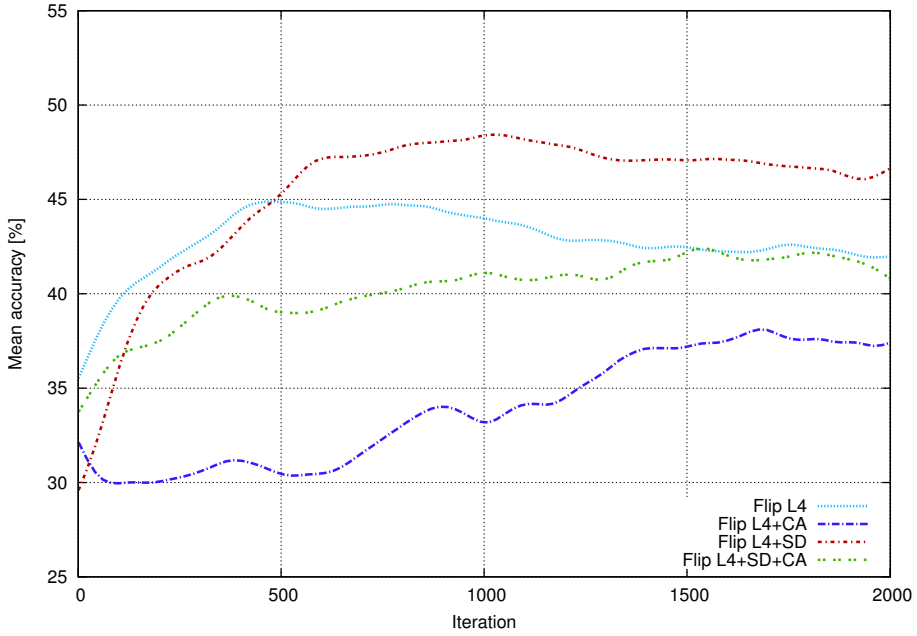


Figure 10: Mean accuracy transition of Flips with or without PCA color augmentation

and validation dataset is shown in Figure 12. According to this, accuracy for the training dataset gets higher as the number of parameters increases (Figure 12 (a)). Accuracy of FlipNR in particular reaches over 60%. However, focusing on the validation dataset result (Figure 12 (b)), FlipNR loses its accuracy more greatly than Flip. This means that FlipNR is adapted to the training dataset excessively.

Finally, we compare the two upsampling methods using Flip and Trans of L1, L2+SD, L4+SD configurations. Accuracy transition of them is shown in Figure 13, in which the Flip configurations are superior to the Trans. Difference between them is significant especially with $L = 4$, though for both Flip and Trans accuracy improves as L gets higher. Since the 3×3 depthwise transposed convolutions in each *Merge* sub-network is performed after image extension using zero padding, as described in Section 3.4, they can convolve only 1, 2, or 4 pixels for each channel (Figure 6 (a)). Compared to the flip-based subpixel reconstruction, which can use the whole 9 pixels (Figure 6 (b)), a transposed convolution is less expressive.

In conclusion, introducing the recursive structure combined with stochastic depth regularization and the flip-based subpixel reconstruction brings better classification accuracy without increasing the number of parameters. On the other hand, applying PCA color augmentation is not effective for the task based on the current dataset. However, there is a possibility of it working well under actual environments, which must be confirmed by improving the dataset further. The peak mean accuracy values of all the configurations throughout the training process up to 2000 iterations are listed in Table 5. Note that, since the accuracy transition graphs are approximated, the values might seem to be higher than expected. The values show almost the same tendency as we have described so far, and the Flip L4+SD configuration marks the highest mean accuracy: 55.1%. Some samples of segmentation results using the parameters corresponding to the table are shown in Figure 14. With $L = 1$, pixels of each class look scattered, whereas $L = 4$ produces better organized results. Especially with the Flip L4+SD, the arrangement of the three organs is roughly satisfactory considering that the purpose of the system is an automatic laparoscope control, even though there is a room for further improvement.

5.2 Inference Speed

Inference speed was evaluated using a desktop PC (Ubuntu 14.04, 64 GB of RAM, Intel Core i7-5930K CPU, NVIDIA GeForce GTX 1080 GPU). The adopted system configuration is the Flip L4+SD, which achieves the best accuracy based on the evaluation results shown in Section 5.1.

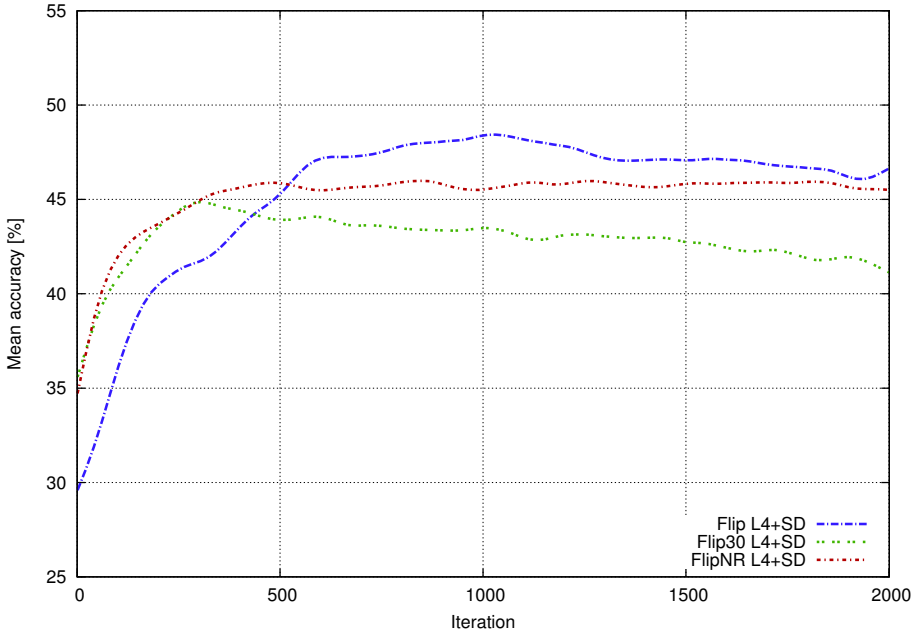


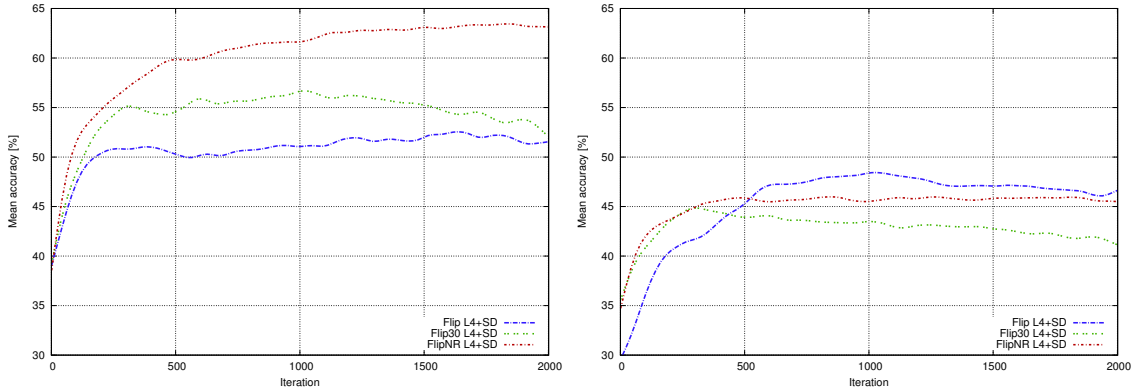
Figure 11: Mean accuracy transition of L4+SD configurations of Flip, Flip30, and FlipNR

As a result, the system could process moving images of size 640×480, the resolution of the camera used in our prototype laparoscope control system, at approximately 17 fps (frames per second). Since our preliminary network configuration, almost equivalent to Flip L4 except that it uses standard 3×3 convolutions, yielded $40 - 50$ fps, GPU implementation of separable convolutions in Chainer does not seem to be fully optimized. However, 17 fps is sufficient for the target application, since a laparoscope does not have to move so quickly. In fact, based on an experiment we have conducted using an actual working robot, a laparoscope could be controlled smoothly and there was little need for a higher framerate.

6 Conclusion and Future Work

We proposed and evaluated the prototype CNN architecture for surgical image segmentation systems to realize automatic laparoscope control. The architecture is based on the encoder-decoder structure using depthwise separable convolutions. It is characterized by its recursive structure that reuses the same sub-networks multiple times, and introduction of the flip-based subpixel reconstruction technique. Furthermore, we applied stochastic depth regularization to the recursive structure. These devices resulted in better peak classification accuracy without increasing the number of parameters, leading to 55.1% mean accuracy despite an insufficiency of the training dataset. The use of PCA color augmentation was also tested. With it, the peak accuracy was lowered but accuracy kept increasing constantly during the training process, which indicates its potential effectiveness in actual environments. The system could operate at approximately 17 fps with a single NVIDIA GeForce GTX 1080 GPU, which is sufficient for robot control.

For future work, more image pairs should be added into the datasets to improve classification accuracy further. In parallel, the network structure should be improved to be adequate to the larger-scale dataset. The recursive structure and the flip-based subpixel reconstruction can be combined with a wide variety of network architectures. For example, squeeze-and-excitation network (SENet) [10] and context encoding network (EncNet) [8] use techniques to consider a global context of an image without a large increase in parameters. Our system targets the specific type of laparoscopic surgery, and such techniques can be an effective mechanism to consider the restriction on an arrangement of the three organs. This is expected to prevent the system from performing unrealistic segmentation, yielding better accuracy as well as more visually natural results.



(a) Training dataset (b) Validation dataset

Figure 12: Comparison of mean accuracy transition for the training and validation datasets

Table 5: Maximum accuracy of each network configuration

Name	Mean Accuracy (%)	Iteration
Flip L1	40.7	1905
Flip L2	43.6	1336
Flip L2+SD	43.3	1436
Flip L4	52.3	910
Flip L4+SD	**55.1**	1059
Flip L4+CA	46.8	1906
Flip L4+SD+CA	49.3	1912
Flip30 L4+SD	53.9	266
FlipNR L4+SD	53.7	326
Trans L1	39.6	802
Trans L2+SD	42.5	1468
Trans L4+SD	45.5	1748

It can also be beneficial to introduce FPGA (Field-Programmable Gate Array) acceleration for power efficiency and compactness improvement, as well as for better performance. In [23], for example, a fully-pipelined CNN with standard convolutions, implemented on an FPGA, shows performance of 60 fps for moving images of size 1920×1080. The theoretical computational complexity of separable convolutions is far less than that of standard convolutions, therefore the dedicated accelerator can realize major performance boost. Leaky ReLU, an activation function we adopted, can reduce required dynamic range when using fixed-point number representation [23], which is an important characteristic to reduce resource usage.

As the first step, we tried implementing a small network ($C = 12$, $L = 4$). Since it does not use separable convolutions and recursive use of *Reduce* and *Merge* is emulated simply by instantiating the same module multiple times, resource utilization reaches about 400% of available resources on the target FPGA (Virtex UltraScale). However, it can operate at about 300 fps with a latency of 1 ms for VGA images when driven by a modest 100 MHz clock. Moreover, the resource utilization can be reduced by appropriate implementation of separable convolutions and the recursive structure. Considering the goal of the system, practical experiments for confirming safety and effectiveness of the system combined with a robot are essential. We have conducted an animal experiment several times using a working robot for maneuvering a laparoscope, and we will continue it to put the system into practical use.

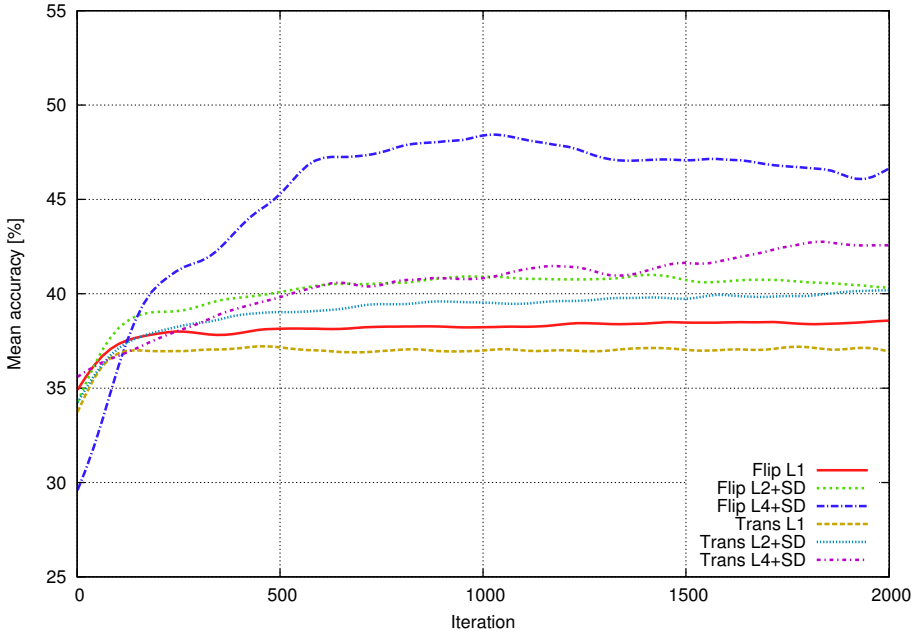


Figure 13: Mean accuracy transition of Flip and Trans configurations

References

[1] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, and W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*, 2017.

[2] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Proc. Neural Information Processing Systems (NeurIPS)*, pages 1106–1114, 2012.

[3] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *International Conference on Machine Learning (ICML) Workshop on Deep Learning for Audio, Speech, and Language Processing (WDLASL)*, 2013.

[4] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[5] F. Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[6] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *Proc. International Conference on Learning Representations (ICLR)*, 2016.

[7] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep Networks with Stochastic Depth. In *Proc. European Conference on Computer Vision (ECCV)*, pages 646–661, 2016.

[8] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context Encoding for Semantic Segmentation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 7151–7160, 2018.

[9] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 6230–6239, 2017.

[10] J. Hu, L. Shen, and G. Sun. Squeeze-and-Excitation Networks. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 7132–7141, 2018.

[11] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[13] K. Turkowski. Filters for Common Resampling Tasks. In A. S. Glassner, editor, *Graphic Gems*, pages 147–165. Academic Press, 1990.

[14] M. Everingham et al. The PASCAL Visual Object Classes Challenge 2012 (VOC2012). http://host.robots.ox.ac.uk/pascal/VOC/voc2012/.

[15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15:1929–1958, 2014.

[17] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 234–241, 2015.

[18] OpenCV Team. OpenCV. https://opencv.org/.

[19] Preferred Networks, inc. Chainer: A Flexible Framework of Neural Networks. https://chainer.org/.

[20] S. Ioffe. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. *Proc. Neural Information Processing Systems (NeurIPS)*, pages 1945–1953, 2017.

[21] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proc. International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[22] T. Manabe, K. Tomonaga, and Y. Shibata. CNN Architecture for Surgical Image Segmentation Systems with Recursive Network Structure to Mitigate Overfitting. In *Proc. International Symposium on Computing and Networking (CANDAR)*, pages 171–177, 2019.

[23] T. Manabe, Y. Shibata, and K. Oguri. FPGA Implementation of a Real-Time Super-Resolution System Using Flips and an RNS-Based CNN. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E101.A(12):2280–2289, 2018.

[24] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2481–2495, 2017.

[25] V. Nair and G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proc. International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[26] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

[27] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *Proc. Artificial Intelligence and Statistics (AISTATS)*, pages 315–323, 2011.

[28] X. Zhang, X. Zhou, M. Lin, and J. Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[29] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521:436–444, 2015.

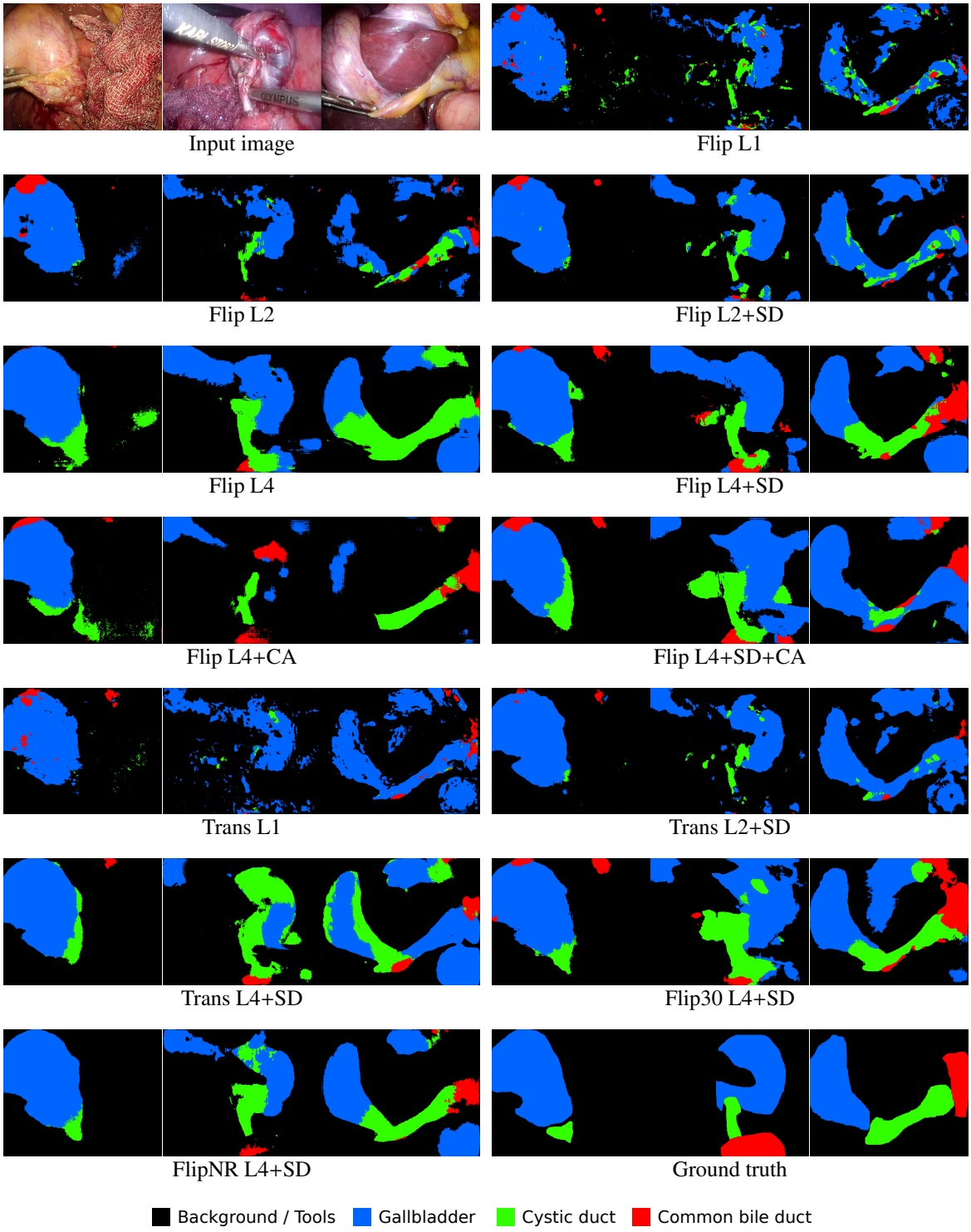


Figure 14: Examples of segmentation results