# Semantische Segmentierung mit Deep Learning

Marvin Teichmann* Martin Thoma*

***Abstract*—Distinguishing medical instruments from background is a basic task which supplements many applications and further analysis. If this task works robust and fast, one can use it for instance segmentation of medical instruments, operation phase detection, pose estimation of the instruments, recognition of the organs and ill tissue. This paper introduces basic ideas of semantic segmentation with deep learning in the context of minimal-invasive operations. A fully convolutional neural network (FCN) is applied to the EndoVis dataset.**

## I. Introduction

Operations like the removal of a tumor or the gallbladder (a cholecystectomy) require the body of the patient to be opened. Traditionally, the surgical incisions were as big as necessary for a surgeon to operate. However, minimal-invasive operations got more and more attentions since 1987 [Wic87]. In this kind of operation, the surgeon tries to make as little and as small cuts in the patients body as possible. The advantage is that the patients skin can heal faster and thus the patient can recover faster from the damage which was done by the operation. The disadvantage of minimal-invasive operations is that the operation itself gets harder for the surgeon. Special medical equipment has to be used: Endoscopes so that the surgeons can see what their doing as well as specialized scissors and tweezers. Machines can not only provide the possibility to make more fine-grained movements (e.g. with the *da Vinci* Surgical System (Intuitive Surgical, Mountain View, Calif)), but also improve vision. For example, the limited visibility due to cauterization-induced smoke can be fought by highlighting the medical instruments, the instruments themselves can be recognized and the operation phase can be detected. If the camera images had a pixel-wise segmentation of medical instruments and background, those tasks would be simpler.

## II. Related Work

Pixel-wise segmentation was successfully applied in other domains. For example, [BKTT15] introduces a system which does pixel-wise segmentation for autonomous cars of the classes *street* and *no street*. A more detailed introduction to semantic segmentation can be found in [Tho16].

An early example of semantic segmentation in medicine is [WAH97]. The authors suggested to place a marker on the instrument which can easily be distinguished from background just by color. The problem with this approach is that existing medical instruments can not easily be colored due to costly approval procedures. Hence a semantic segmentation algorithm which works with existing medical instruments is necessary.

For example, in [AOT+13] the authors detected and localized surgical instruments in laparoscopic images to estimate the pose of the instrument in 3D. They used Random Forests.

## III. Models

In this practical, we were examining deep neural network models for semantic segmentation. The most basic neural network model is a so called *multilayer Perceptron* (*MLP*). MLPs consist of an input layer, multiple hidden layers and an output layer. Each layer consists of nodes. The number of nodes of the input layer is determined by the features and the number of output nodes is determined by the classes. The number of nodes in the hidden layers can be arbitrary large, but is typically between 0.25 and 3 times the number of the layer before. The parameters which get adjusted during the training of a model are called *weights*. One weight is between two neurons of neighboring layers.

Each node has an activation function. Two functions which are used very often are the *sigmoid function* and the *rectified linear function*.

The sigmoid activation function is

$$\varphi : \mathbb{R} \to (0, 1) \qquad \varphi(x) = \frac{1}{1 + e^{-x}}$$

while rectified linear units (commonly abbreviated with ReLU) have the activation function

$$\varphi : \mathbb{R} \to [0, \infty) \qquad \varphi(x) = \max(0, x)$$

A more detailed introduction to MLPs can be found in [Mit97].

### A. Convolutional Neural Networks

One of the most important extensions of MLPs are so called *Convolutional Neural Networks* (CNNs). Those introduce two new layer types besides the fully connected layers: convolutional layers and pooling layers. Each convolutional layer has $k \in \mathbb{N}_{\geq 1}$ filters of size $p \times p \times c$, where $p \in \mathbb{N}_{\geq 1}$ is a design choice and $c \in \mathbb{N}_{\geq 1}$ is the number of channels of the layer before. The input layer has typically three channels:

* These authors contributed equally to this work.

Red, green, blue (RGB). When $k$ filters get applied in layer $A$, then the output of layer $A$ has $k$ channels. The parameters which get adjusted during training in convolutional layers are the weights of the filters. A layer with $k$ filters of size $p \times p \times c$ has $k \cdot p^2 \cdot c$ weights which get adjusted.

Convolutions produce an output which is smaller than their input, as one can see in Figure 1.

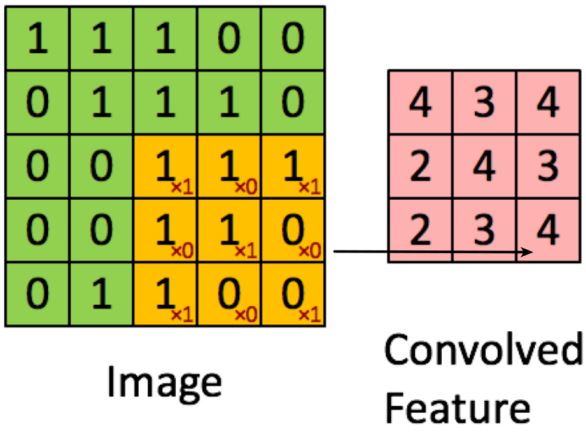


Figure 1: A $5 \times 5$ image convolved with a $3 \times 3$ filter becomes a $3 \times 3$ image as the filter size minus one gets removed from the input image size. Image source: [Cyf]

Sometimes it is desired to get an output of the same size as the input. In this case the image is padded with $\frac{\text{filter size} - 1}{2}$ in each direction. Commonly, the padding is done with zeroes.

The pooling layers do not learn any parameter. They reduce the data being processed by the network by grouping it. Pooling layers have four hyperparameters: the pooling function, the pooling size, the strides and the border mode. The pooling size gives the size of the region to which the pooling function gets applied. Typically, $3 \times 3$ pooling is chosen. For those four pixels the maximum function is chosen. Another option is average pooling. The stride is the parameter which reduces the amount of processed data. A stride of $(2, 2)$ is used which has the effect of reducing the data to $1/4$th of the original amount.

There are a couple of well-known image classification networks for which both, the architecture and the weights are available. One of those networks is called VGG-16 (for *Visual Geometry Group*) [KS], [SZ14].

### B. Fully Convolutional Neural Networks

Fully Convolutional Networks (FCNs) were introduced by [LSD15]. They were shown to improve performance in semantic segmentation tasks over standard convolutional neural networks (CNNs).

FCNs combine four ideas:

1) Train an image classification network for $k$ classes,
2) interpret this network as $k$ filters of a single, non-linear convolution,
3) apply those filters to images of arbitrary size to receive a coarse heat map for the $k$ classes.
4) Finally, train an upsampling network to get a fine-grained semantic segmentation.

Upsampling is realized by padding and convolution as visualized in Figure 2. It is worth emphasizing that this upsampling is learned.

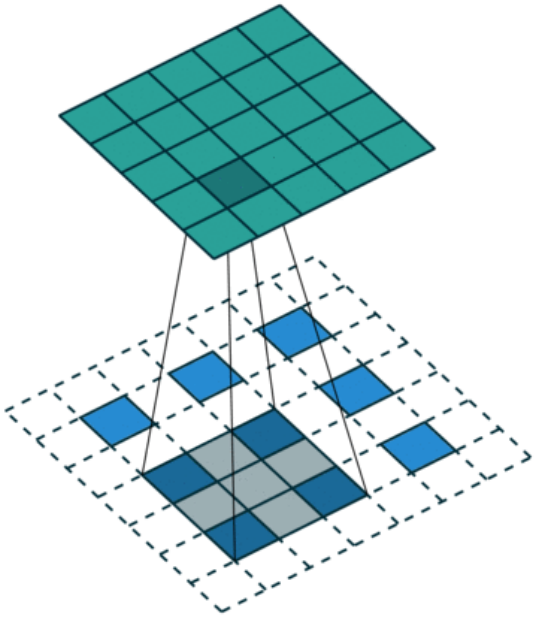

Figure 2: Upsampling: The blue input gets padded by the white cells. Then a convolution is applied — visualized in gray — to produce the green output.
Image source: [vdu16]

## IV. Experiments

A desktop computer with a Titan Black and an Intel Core i7-4930K was used for the training of the model and its evaluation.

### A. The data

The experiments were done on the *Instrument segmentation and tracking* dataset of the Endoscopic Vision

Challenge "EndoVis"[1]. It contains photos of minimal-invasive operations. The dataset already contains a training-test-split. The training data consists of four operations. Each operation has 40 RGB images in a resolution of $640\,\text{px} \times 480\,\text{px}$. The test set has 10 more images of those 4 operations as well as 50 images for 2 more operations. This means, in total the training data consist of $4 \cdot 40 = 160$ photos and the testing contains $4 \cdot 10 + 2 \cdot 50 = 140$ photos. As each pixel has to be classified as *medical instrument* or *background*, there are $140 \cdot 640 \cdot 480 = 43\,008\,000$ classifications to be done for testing.

The training data consist of $90.5\,\%$ background ($90.8\,\%$ in the testing data); the remaining pixels are medical instruments. This means an accuracy of $90.8\,\%$ can be reached without looking at the test image.

Analyzing the data, one can also see that some positions in the image are more likely than others to contain a medical instrument. This is visualized in Figure 3.

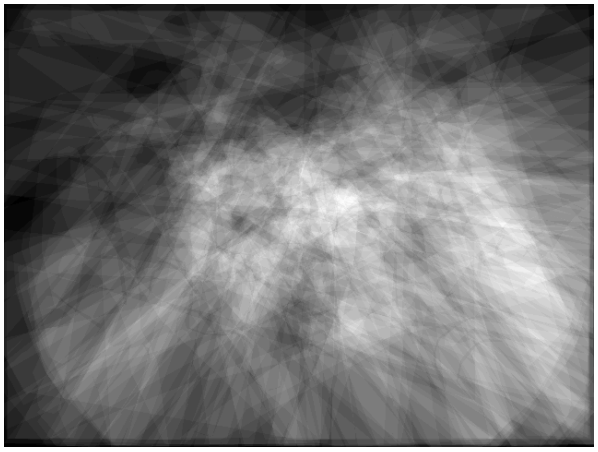

Figure 3: Position of medical instruments in the training data. The brighter the color, the more often was a medical instrument at this position. One can clearly see that the borders contain medical instruments less often. Also, in the center they are much more often.

### B. Base line experiments

The most basic information for pixel-wise semantic segmentation is the color of the pixel. Typically, images are in RGB format. This means the image has three channels (Red, Green, Blue). Each channel has 8 bit and thus $2^8 = 256$ possible values, ranging from 0 to 255. This gives $(2^8)^3 = 16\,777\,216$ possible colors for each pixel. Obviously, only the color can not give a perfect result in all circumstances as the measured color changes due to smoke, shadows, specular highlights and insufficient illumination. But it gives an impression how important local features are for the specific problem.

A model with 64 sigmoid nodes in a first hidden layer with $50\,\%$ dropout [SHK+14], 64 ReLu nodes in a second hidden layer with $50\,\%$ dropout and one sigmoid output unit was used as a baseline.

The architecture of the baseline model is visualized in Figure 4. Neither preprocessing nor data augmentation were applied.

The baseline model achieved a pixel-wise accuracy of $92.88\,\%$,[2] a precision of $76.13\,\%$ and a recall of $32.94\,\%$. The confusion matrix is given in Table II.

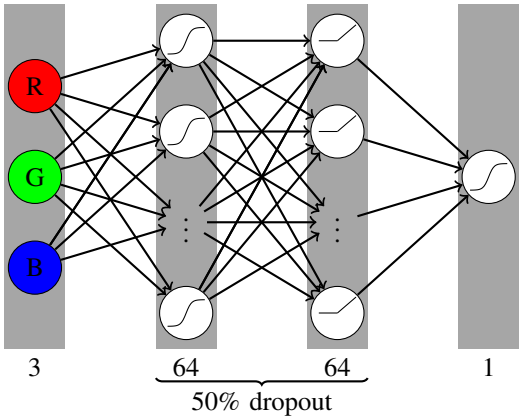


Figure 4: Architecture of the baseline model.

### C. Local Models

A slightly better model than the baseline used the coordinate of the pixel in the image as two additional features ($x, y \in \mathbb{R}$) as well as $3\,\text{px}$ erosion and dilation. This is known as a morphological opening. This model achieved an higher accuracy and a higher precision, but the recall improved most. As one can see in the results in Table I, most of the improvement is due to the coordinate features. The confusion matrix is given in Table IV.

An example for the segmentation can be seen in Figure 5. The model fits some of the edges of the medical instruments very well, but has severe problems with specular highlights. It is also not able to deal with the red reflection on the medical instruments head and very dark, but not black areas with little image information.

[1] http://endovis.grand-challenge.org

[2] This is the same as the DICE coefficient.

| Model | Accuracy | Precision | Recall |
|---|---|---|---|
| Baseline (BL) | 92.88 % | 76.13 % | 32.94 % |
| BL + CFs | 93.42 % | 76.98 % | 40.65 % |
| BL + Opening | 91.07 % | 74.76 % | 4.50 % |
| BL + CFs + Opening | 93.51 % | 76.76 % | 42.30 % |

Table I: Results of the local model experiments. The coordinate features (CFs) improved the quality.

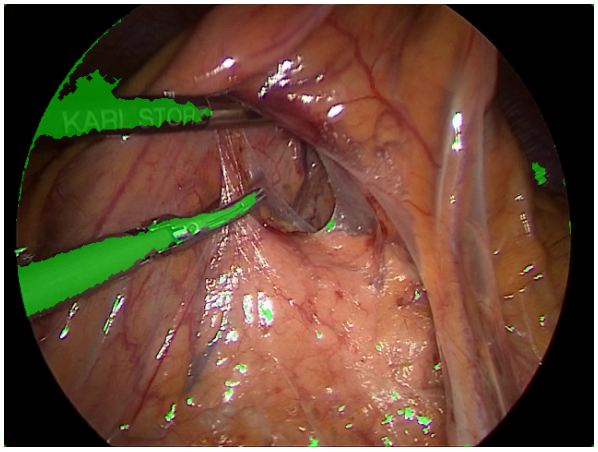

Figure 5: Example segmentation by the local model with opening.

Astonishingly, the accuracy of models which made use of a tiny patch around the pixel which was to classify decreased to about 92.11 %, the precision is 42.30 % and the recall is 76.74 %. The classification of one image took about 1.54 s. In theory, the model could simply set the weights to 0 and thus ignoring the other features. This means such a model should never be worse than a model without the surrounding pixel information. Besides programming errors and numerical problems, the problem might be the increased number of parameters. It is possible that the optimization was stuck in a local minimum.

### D. Fully Convolutional Networks

*1) Architecture:* In the following, the mean value of the image net dataset was subtracted from each image channel (mean blue: 103.939, mean green: 116.779, mean red: 123.68).

For the following experiment, the VGG-16 architecture as introduced in [SZ14] was used. Not only was the architecture used, but the network was also pre-trained on image net and used as described in [Fos16]. The VGG-16 architecture consists of 20 layers:

1) `conv1_1`, `conv1_2`, `pool1`: Two convolutional layers with 64 filters of size $3 \times 3$ each, followed by max pooling with a stride of $s = (2, 2)$ and a pooling region of $p = (2, 2)$.
2) `conv2_1`, `conv2_2`, `pool2`: Two convolutional layers with 128 filters of size $3 \times 3$ each, followed by max pooling with a stride of $s = (2, 2)$ and a pooling region of $p = (2, 2)$.
3) `conv3_1`, `conv3_2`, `conv3_3`, `pool3`: Three convolutional layers with 256 filters of size $3 \times 3$ each, followed by max pooling with a stride of $s = (2, 2)$ and a pooling region of $p = (2, 2)$.
4) `conv4_1`, `conv4_2`, `conv4_3`, `pool4`: Three convolutional layers with 512 filters of size $3 \times 3$ each, followed by max pooling with a stride of $s = (2, 2)$ and a pooling region of $p = (2, 2)$.
5) `conv5_1`, `conv5_2`, `conv5_3`, `pool5`: Three convolutional layers with 512 filters of size $3 \times 3$ each, followed by max pooling with a stride of $s = (2, 2)$ and a pooling region of $p = (2, 2)$.
6) `fc6`: A fully connected layer with 4096 nodes.
7) `fc7`: A fully connected layer with 1000 nodes.

The network was trained on image net data. The idea is to use the general image recognition features learned in the first layers. Thus the network is a very sophisticated preprocessing algorithm.

After `fc7`, a convolutional layer `score_fr` with $k$ filters of the shape $1 \times 1 \times 1000$ is added where $k$ is the number of classes. Then an upscore layer `up` is added to scale the coarse output of `score_fr` to the same size as the input image.

The $k = 2$ upsampling filters of size $64 \times 64$ were initialized as a bilinear upsampling. Those filters make the result so consistent. As they mainly interpolate between neighboring pixels, they will never give unrealistically small segments.

The FCN was trained end-to-end with the mean softmax cross entropy between logits and labels as training objective:

$$-\frac{1}{|D|} \sum_{(\mathbf{x},\mathbf{y}) \in D} \sum_{y_i \in \mathbf{y}} [y_i \ln a_i + (1 - y_i) \ln(1 - a_i)]$$

where $D$ is the set of training data, $x$ is a feature vector, $y$ is a one-hot encoded target and $a_i$ is the prediction for the $i$-th class of the classifier.

Each weight was regularized with L2 loss. This means the original objective function $f_{\text{training data}}(\mathbf{w})$ which gets minimized was modified to

$$\tilde{f}_{\text{training data}}(\mathbf{w}) = f_{\text{training data}}(\mathbf{w}) + \sum_{w \in \mathbf{w} w^2}$$

Not only the given training data was used, but it was also heavily augmented by the following methods:

- Horizontal flipping
- Vertical flipping
- Horizontal and vertical flipping
- Random brightness adjustments
- Random crops were taken

*2) Results:* The FCN evaluation results are shown in Table V. Our fully convolutional network is noticeably better than the other semantic segmentation systems.

A training run with 5000 epochs took about 5.5 hours. Running the network on a single image takes in average 1.3 s.

*3) Analysis of data augmentation:* Data augmentation is commonly done to train the network to become invariant to changes in the data which should not matter. For example, in computer vision this is an image-wide change in brightness.

Data augmentation becomes especially useful if very little data is present to artificially increase the available training material. However, it is unclear how much data is enough data and which data augmentation methods have which effect on the quality of the semantic segmentation system.

We were interested in three properties: The quality of the segmentation system at the end of 10 000 epochs (11 hours) of training, the stability of the quality during training and the speed of convergence.

To evaluate the quality of the training, four metrics were used and evaluated on the test set after each 250 epochs:

- Accuracy: $\frac{TP+TN}{TP+TN+FP+FN} \in [0,1]$
- Precision: $\frac{TP}{TP+FP} \in [0,1]$
- Recall: $\frac{TP}{TP+FP} \in [0,1]$
- F1-Score: $2\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} \in [0,1]$

Other metrics besides the accuracy were used as over 90 % of the data was background. Precision and recall are typical choices to solve this problem. We thought about using the F1 score as a single number is always simpler to compare than two numbers. However, for our experiments with FCNs on this dataset the Person correlation coefficient of the accuracy and the F1 score is 0.96. The F1 score has lower values and a higher variance, but it shows the same characteristics for FCNs on this datasets.

Figure 13 suggests that data augmentation has little to no effect on the overall accuracy. Only random crops seem to have a negative effect on accuracy.

Figures 14 and 15 show that the precision and recall have a lot of variance for all data augmentation methods. Random crops seem to have a negative effect on precision and recall as well.

All training methods suffer from enormous outliers in the quality even during late stages in the training.

We did not notice any effect of any data augmentation method on the stability during the training.

## V. Discussion

Fully Convolutional Networks are also in this domain clearly superior to traditional approaches. However, examining results such as the one shown in Figure 6, one can see that there is room for improvement. While the overall position was correctly detected and the proposal is fairly consistent, the model fails to match borders exactly. Simple improvements such as combining the results of a FCN with the baseline might already fix this problem. More elaborate techniques such as Conditional Random Fields (CRFs) are also promising.

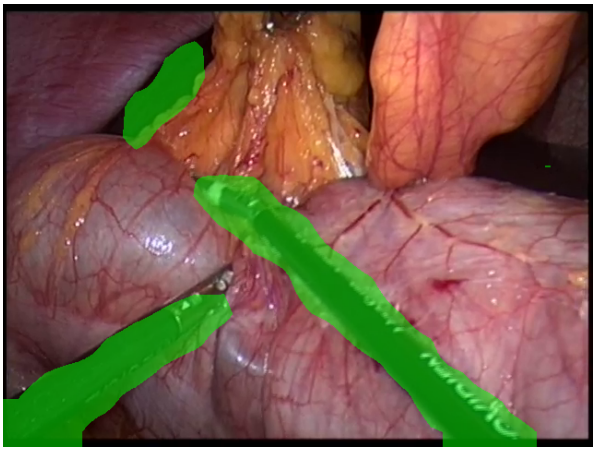

Figure 6: Semantic segmentation result of the FCN. While it

## VI. Acknowledgement

We would like to thank the “Begabtenstiftung Informatik Karlsruhe” for supporting our research. They supported the work on TensorVision (github.com/TensorVision/TensorVision), which was used for this practical.

## References

[AOT+13] M. Allan, S. Ourselin, S. Thompson, D. J. Hawkes, J. Kelly, and D. Stoyanov, “Toward detection and localization of instruments in minimally invasive surgery,” *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 4, pp. 1050–1058, 2013. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6359786

[BKTT15] S. Bittel, V. Kaiser, M. Teichmann, and M. Thoma, "Pixel-wise segmentation of street with neural networks," *arXiv preprint arXiv:1511.00513*, 2015. [Online]. Available: http://arxiv.org/abs/1511.00513

[Cyf] Cyfoo, "Convolution schematic." [Online]. Available: http://deeplearning.stanford.edu/wiki/index.php/File:Convolution_schematic.gif

[Fos16] D. Fossard, "VGG in Tensorflow," Jun. 2016. [Online]. Available: http://www.cs.toronto.edu/~frossard/post/vgg16/

[KS] A. Z. Karen Simonyan, "Very deep convolutional networks for large-scale visual recognition." [Online]. Available: http://www.robots.ox.ac.uk/~vgg/research/very_deep/

[LSD15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440. [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=7478072

[Mit97] T. M. Mitchell, *Machine learning*. McGraw-Hill, Mar. 1997.

[SHK+14] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[SZ14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. [Online]. Available: http://arxiv.org/pdf/1409.1556.pdf

[Tho16] M. Thoma, "A survey of semantic segmentation," *arXiv preprint arXiv:1602.06541*, Feb. 2016. [Online]. Available: http://arxiv.org/abs/1602.06541

[vdu16] vdumoulin, "vdumoulin/conv-arithmetic," May 2016. [Online]. Available: https://github.com/vdumoulin/conv_arithmetic

[WAH97] G.-Q. Wei, K. Arbter, and G. Hirzinger, "Automatic tracking of laparoscopic instruments by color coding," in *CVRMed-MRCAS'97*. Springer, 1997, pp. 357–366. [Online]. Available: http://link.springer.com/chapter/10.1007/BFb0029257#page-1

[Wic87] J. E. Wickham, "The new surgery." *British medical journal (Clinical research ed.)*, vol. 295, no. 6613, p. 1581, Dec. 1987.

# APPENDIX A
# TABLES

| | | Predicted class | |
|---|---|---|---|
| | | **0** | **1** |
| **Actual class** | **0** | 38 640 407 | 2 654 875 |
| | **1** | 408 816 | 1 303 902 |

Table II: Confusion matrix of a model trained soley on pixel colors. Class 0 is background, class 1 is medical instruments.

| | | Predicted class | |
|---|---|---|---|
| | | **0** | **1** |
| **Actual class** | **0** | 38 989 042 | 3 780 484 |
| | **1** | 60 181 | 178 293 |

Table III: Confusion matrix of a model trained soley on pixel colors. Additionally, a morphological closing operation with $3\,\mathrm{px}$ was applied. Class 0 is background, class 1 is medical instruments.

| | | Predicted class | |
|---|---|---|---|
| | | **0** | **1** |
| **Actual class** | **0** | 38 541 570 | 2 284 099 |
| | **1** | 507 653 | 1 674 678 |

Table IV: Confusion matrix of a model trained on pixel colors and the pixels coordinates. Additionally, a morphological closing operation was applied. Class 0 is background, class 1 is medical instruments.

| **Model** | **Accuracy** | **Precision** | **Recall** |
|---|---|---|---|
| Baseline | 92.88 % | 76.13 % | 32.94 % |
| | (+8 %) | (+4 %) | (+11 %) |
| Local model + opening | 93.42 % | 76.98 % | 40.65 % |
| | (+62 %) | (+41 %) | (+77 %) |
| FCN | **97.32** % | **85.87** % | 84.90 % |
| KIT | 93 % | 69 % | 46 % |
| Lausanne | 96.5 % | 76 % | **93** % |

Table V: Results of the Endoscopic Vision Challenge "EndoVis"

## Appendix B
## Graphs

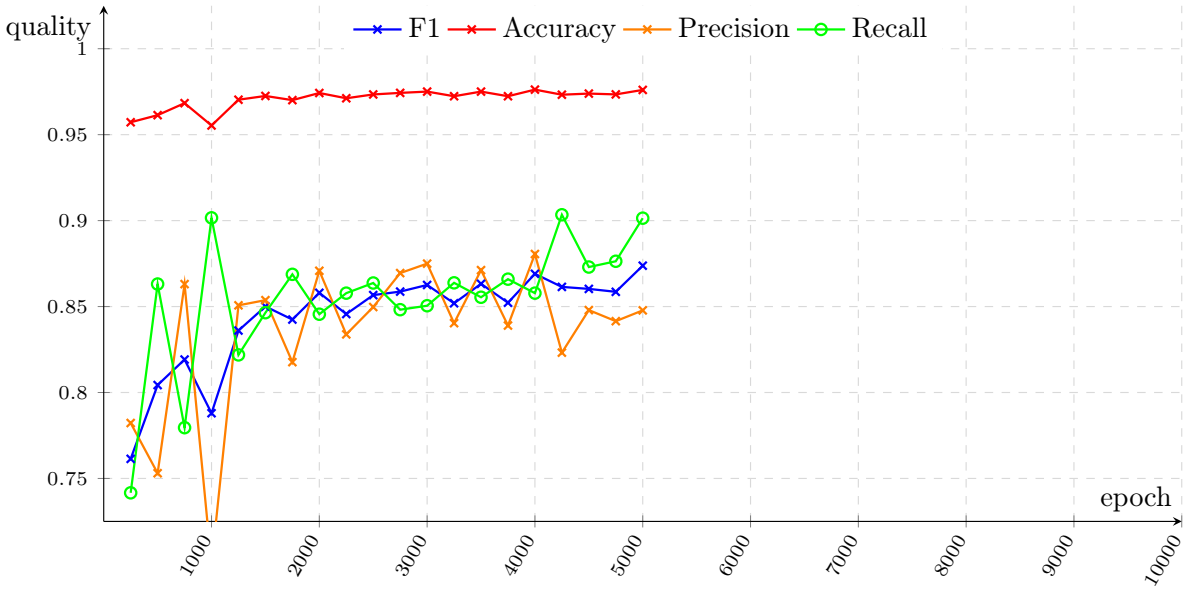


Figure 7: Training without data augmentation

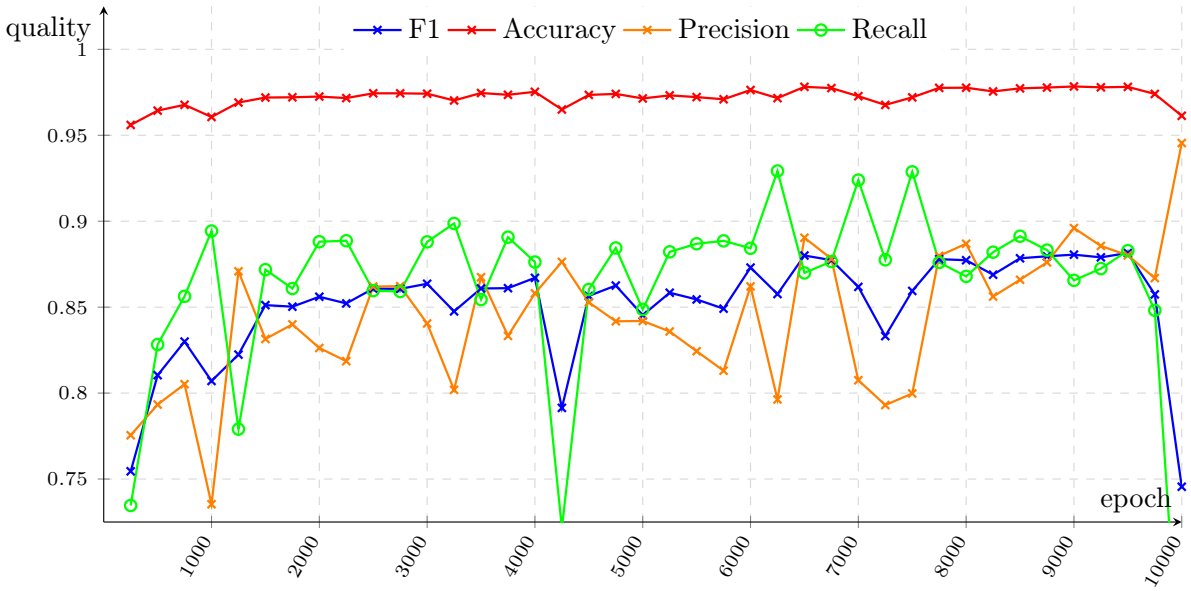


Figure 8: Training without data augmentation

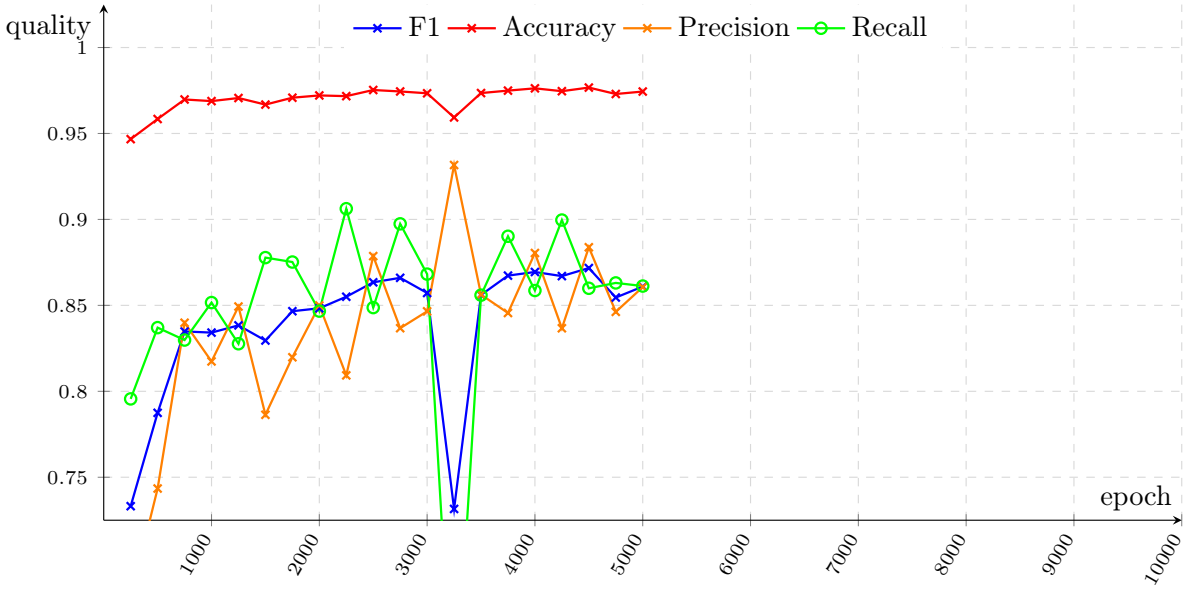


Figure 9: Training with data augmentation (left-right flip)

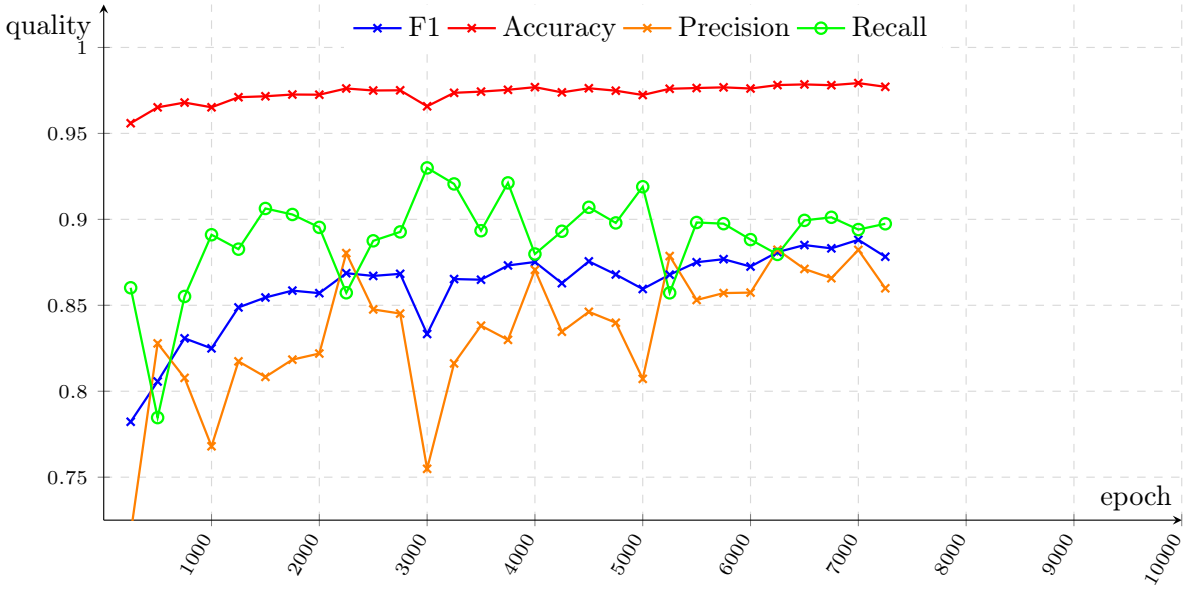


Figure 10: Training with data augmentation (up-down flip)

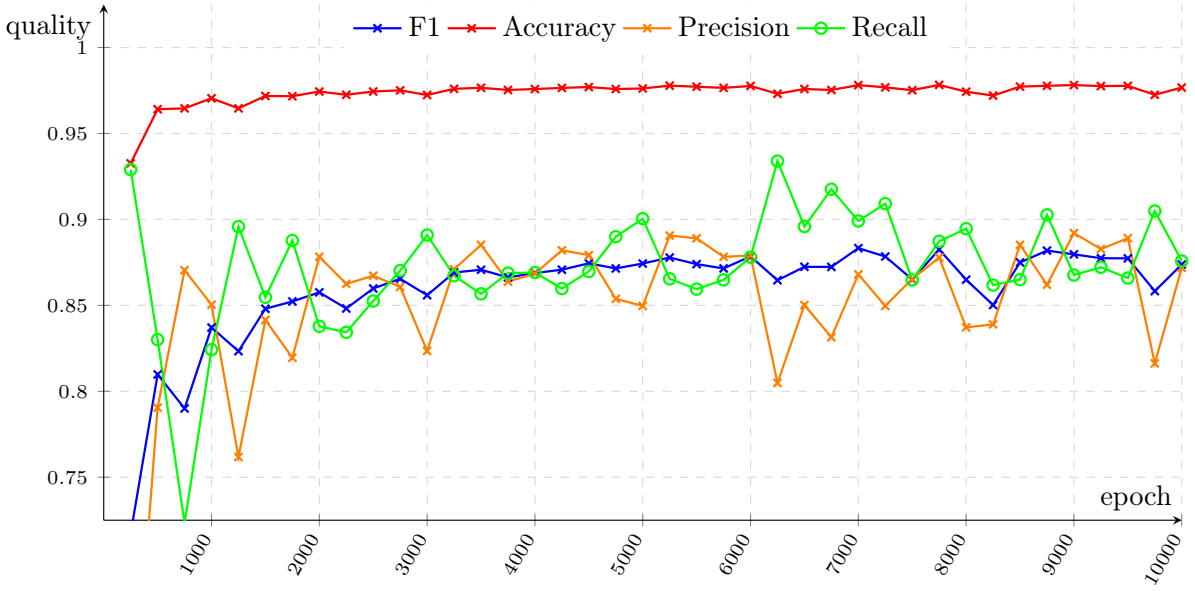


Figure 11: Training with data augmentation (random brightness)

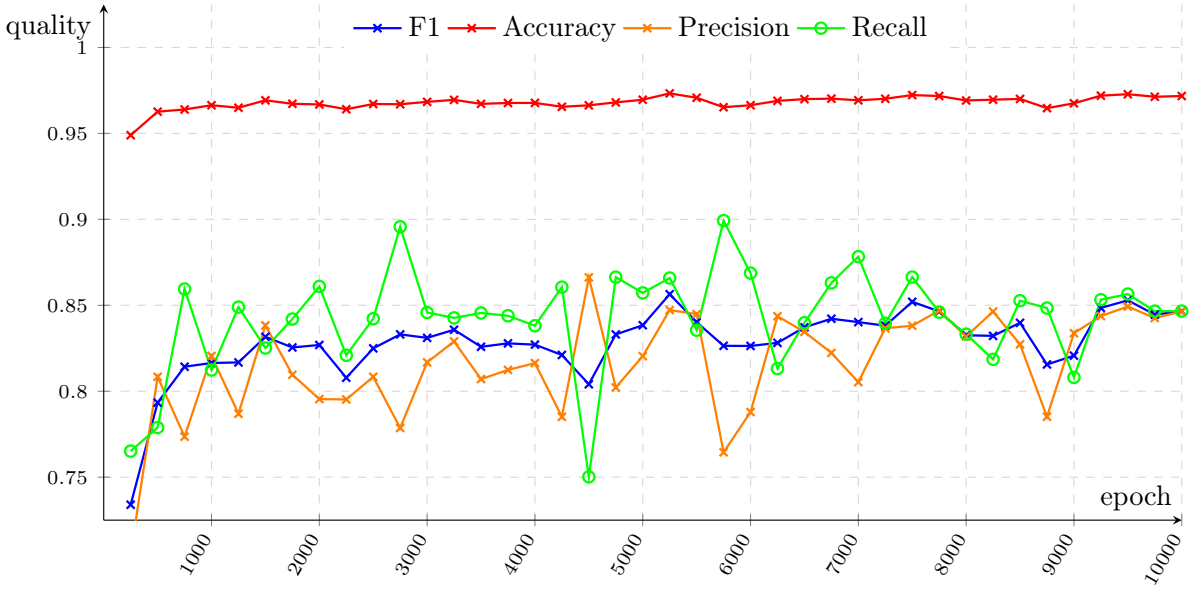


Figure 12: Training with data augmentation (random crops)

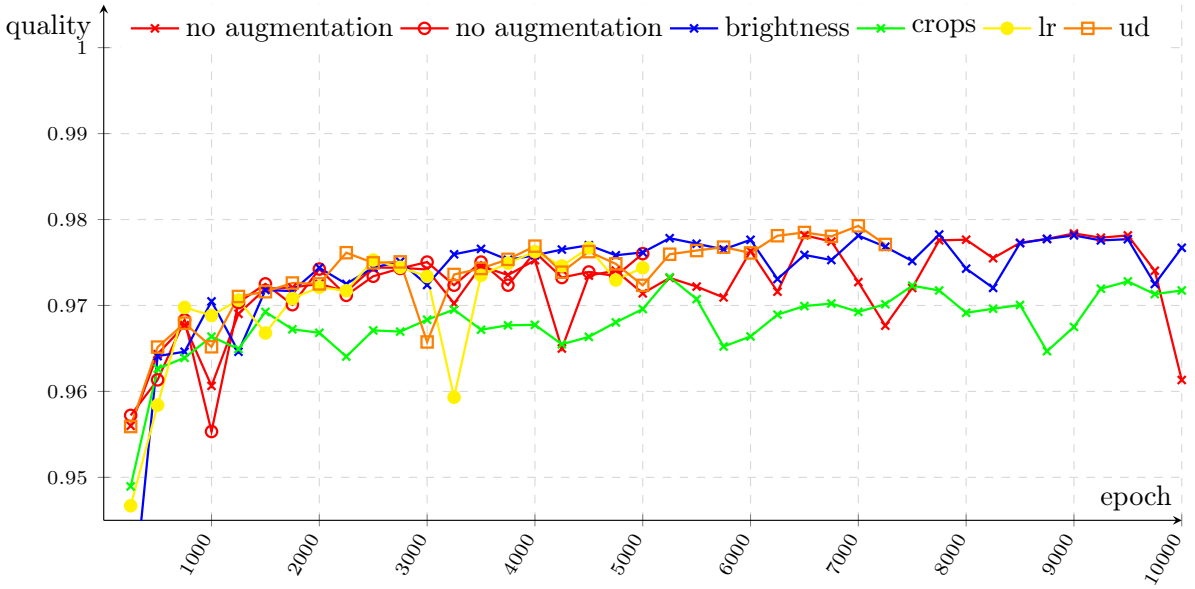


Figure 13: Accuracy over epochs of FCNs with different data augmentation setups. No augmentation was evaluated twice with different random initialization, random brightness and random crops as well as upside down flips and left-right flips.

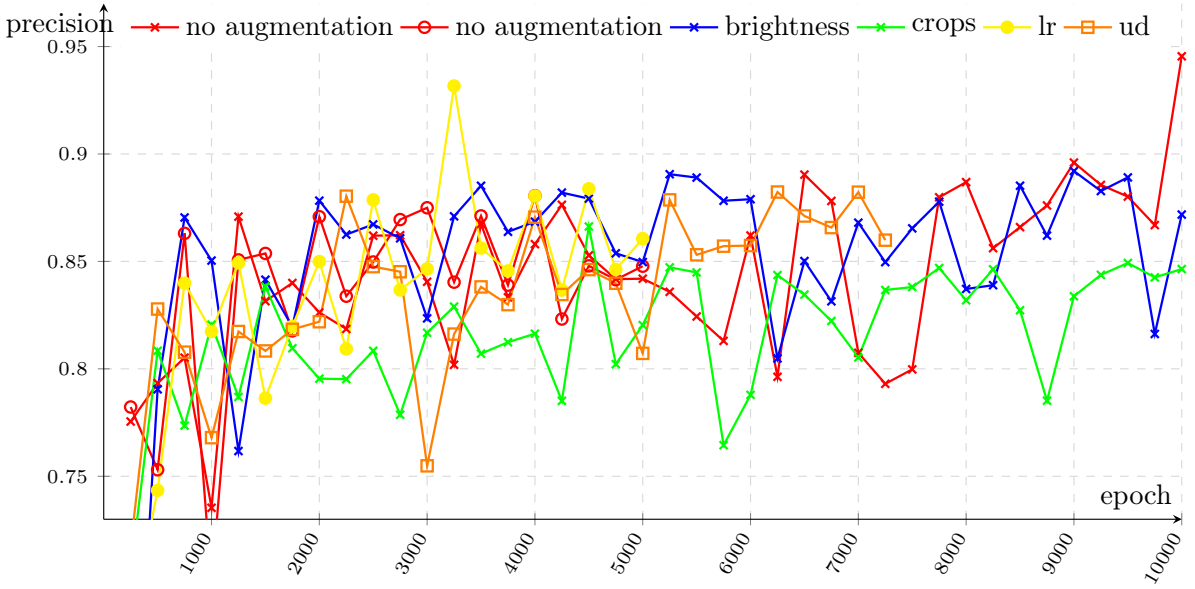


Figure 14: Precision over epochs of FCNs with different data augmentation setups. No augmentation was evaluated twice with different random initialization, random brightness and random crops as well as upside down flips and left-right flips.

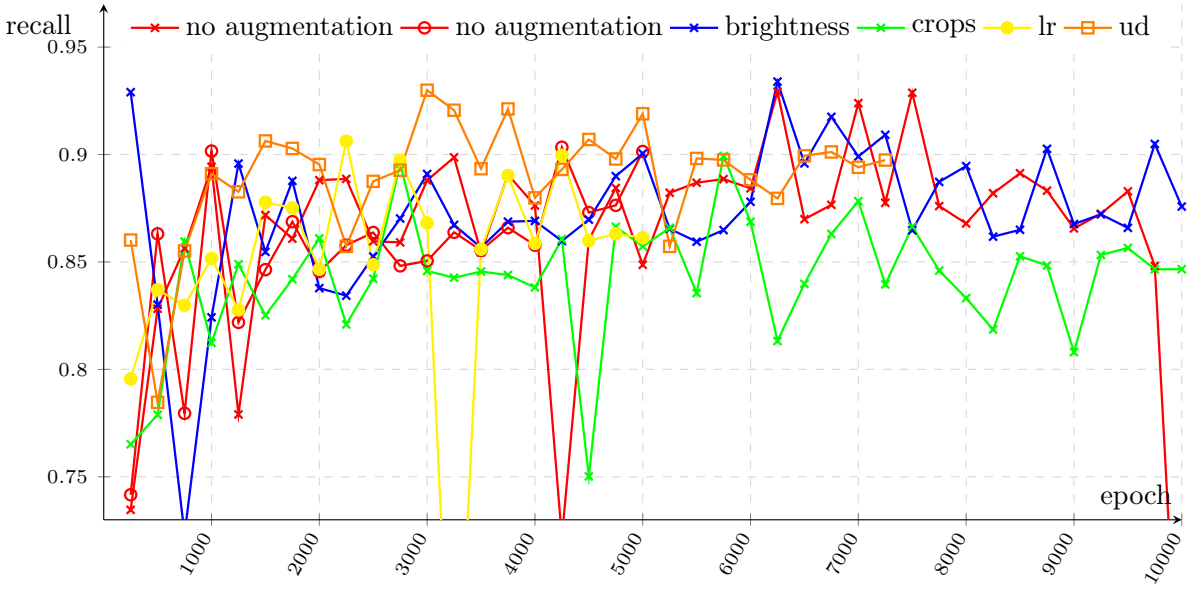


Figure 15: Recall over epochs of FCNs with different data augmentation setups. No augmentation was evaluated twice with different random initialization, random brightness and random crops as well as upside down flips and left-right flips.