CS482/682 Final Project Report Group 10

Exploiting 3D CNNs and Contrastive Learning for Robotic Tool Segmentation

Wencheng Zhong/wzhong10, John Cast/jcast1, Aayush Mishra/amishr24, Kexin Wang/kwang101

1 Introduction

Background Surgery robots not only help doctors perform complex surgeries precisely and minimize the cut, but also allow remote operations, advocating equal access to the medical resource. Pixel-wise video segmentation on the fly is fundamental to this demanding task with limited training set, where the utmost importance is always the accuracy. Vision-based deep learning networks like U-Net are very popular in this field, but vulnerable to some common interference like blood and tissue occlusion. In this work, we demonstrate how their precision and robustness can be improved by introduction of unsupervised signals and intelligent data augmentation.

Related Work Robotic tool segmentation is a well-researched topic. U-Net [1] is the go-to tool for segmentation tasks because of the proven superiority of using skip connections over other contemporary methods. Recent developments like [2, 3, 4, 5] are based on U-Nets. Unsupervised learning methods like Contrastive Predictive Coding (CPC) [6], extract representations from high-dimensional data in a task-agnostic manner preserving meaningful information which can be used elsewhere.

2 Methods

Dataset We use a public dataset, generated using the da Vinci Research Kit (dVRK) [5], consisting of 14 videos (8 training, 2 validation, 4 test) of 300 frames each, and kinematic data together with labels. A state-of-the-art was also established in [5].

Overall Architecture We use a 3D CNN based U-Net instead of a 2D CNN in prior works. We argue that 3D CNNs better capture temporal information between frames and hence choose 10-frame sequences as our model inputs and outputs. In addition, we pass unsupervised signals to our U-Net decoder by pre-training two CPC [6] networks (on video and kinematics data respectively). Note that our CPC models are unconventional because they are trained on sequences of sequences of images, demonstrated in App. B and C. We argue that this combined information improves the robustness of our decoder. See Figure 1 for a diagram of the approach and App. A for more details. The supervised signal z_e is obtained from the U-Net encoder, while the unsupervised video signals z_f and kinematic signal z_k are obtained via CPC embeddings.

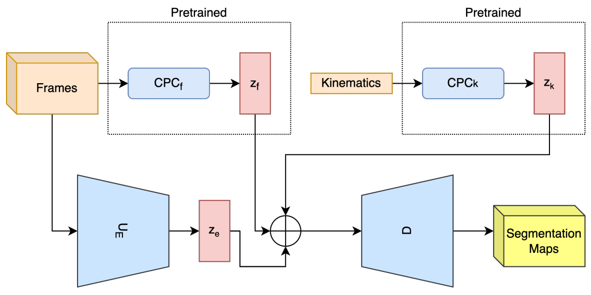


Figure 1: Methodology.

Unsupervised pretrained CPC encoders We pretrain each CPC network individually for corresponding frames and kinematics data. These embeddings are later concatenated to the original bottleneck of the 3D U-Net and mixed via a convolutional layer for the decoder before fine-tuning.

Data Augmentation In addition to conventional brightness, hue, and contrast augmentations, we develop a novel on-line data augmentation technique for simulating tissue occlusion. This helps increase the robustness of our model as evident from Fig. 2.
Loss and Evaluation Dice loss and BCE loss are used for U-Net and CPC training respectively. Like prior work, we use IoU Score as the evaluation metric.

3 Results

Results in Max IoU(%) score and IQR(%) of IoU scores

Methods	**IoU (%)**	**IQR(%)**
TernausNet [4](baseline)	76.67	13.63
UCL [5]	88.49	11.22
3D UNet	90.42	7.49
3D UNet (Occlusion trained)	*90.05*	7.95
3D UNet w/ CPC (Image)	91.86	9.86
3D UNet w/ CPC (Both)	**93.21**	8.25

4 Discussion

Our results are summarized in the table above. The performance of the base 3D U-Net alone beats SOTA without much hyperparameter tuning. This supports our argument of taking advantage of both spatial and temporal information in the 3D input.
3D U-Net + CPC We find that the 3D U-Net with CPC embeddings added as input on the bottleneck increases the average median IoU score by $\approx 0.76\%$. Our best model (combining U-Net with both CPC embeddings) scores $\approx 2.8\%$ above the top U-Net-only model in terms of median IoU. This strongly supports our claim of unsupervised signals being helpful.
Bottleneck Mixing We experimented using two different methods of mixing the CPC embeddings at the U-Net bottleneck. First, a naive averaging method and second, concatenation of embeddings at the UNet Bottleneck with an additional convolution layer. We find that mixing using convolution performs with higher average median IoU than the naive averaging.
Kinematic Data [5] use kinematic data to generate simulated images to use in addition to real. We argue that using this data directly in an unsupervised fashion using CPC would equate if not improve over their approach. Our results table conveys similarly.

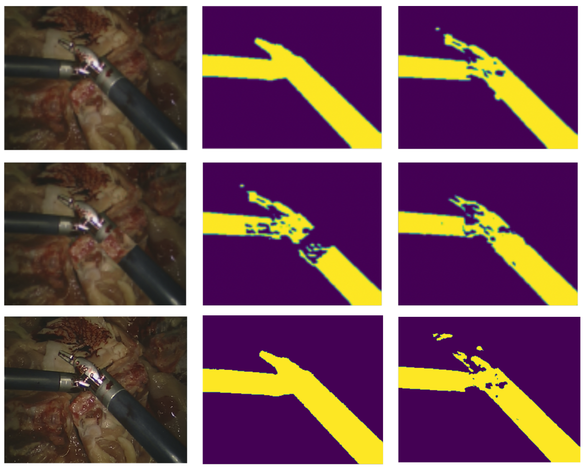


Figure 2: Row 1: UNet. Row 2: UNet, occlusion. Row 3: U-Net + both CPC.

Augmentation Our experiment with bottleneck space noise augmentation like in [7] didn't help. However, our novel occlusion based augmentation improved the robustness against such issues. Median IoU Scores for our base UNet dropped to *88.78* from *90.42* if test frames had this occlusion, while a model trained with these occlusions stayed at a healthy *89.74* from *90.05*. However, we could only compare with our base model. Details in Appendix.
Future Work In future, we propose doing more ablation studies with the combination of CPC embeddings and the 3D U-Net bottleneck. Specifically, fine-tunings using the combination of CPC embeddings with the 3D U-Net when using our tissue occlusion augmentation would make an interesting point of comparison with our current findings. Additionally, we would like to explore other methods of mixing the CPC embeddings at the bottleneck of the 3D U-Net. Furthermore, the use of cross-consistency training regime could prove useful in making the final model more robust [7].

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[2] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[3] Alexey A Shvets, Alexander Rakhlin, Alexandr A Kalinin, and Vladimir I Iglovikov. Automatic instrument segmentation in robot-assisted surgery using deep learning. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 624–628. IEEE, 2018.

[4] Vladimir Iglovikov and Alexey Shvets. Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. *arXiv preprint arXiv:1801.05746*, 2018.

[5] Emanuele Colleoni, Philip Edwards, and Danail Stoyanov. Synthetic and real inputs for tool segmentation in robotic surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 700–710. Springer, 2020.

[6] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[7] Yassine Ouali, Céline Hudelot, and Myriam Tami. Semi-supervised semantic segmentation with cross-consistency training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12674–12684, 2020.

Appendices

A Overall Architecture

The best model in our paper is the 3D U-Net with the unsupervised pretrained CPC image and kinematic encoder, overall architecture is explained in the main part and diagrammed in Fig. 3 here.

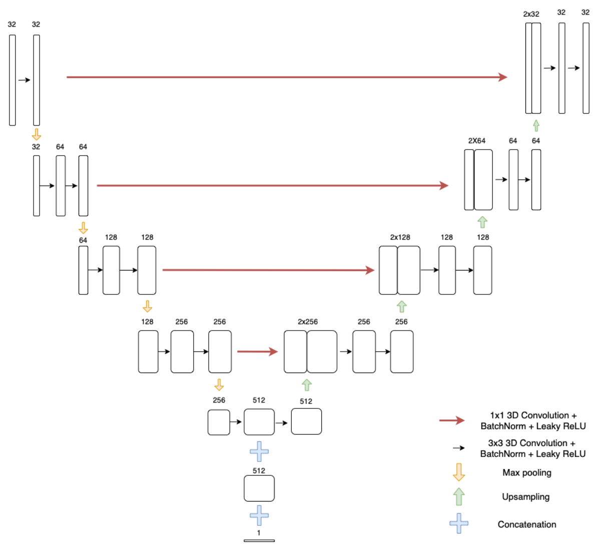


Figure 3: CPC for Sequence of Image Sequences

B CPC Image Network

The unsupervised learning of CPC Image Network works like this: a set of consecutive sequences of video frames are input as training terms; then another set of consecutive sequences of video frames are given as terms for the network to predict if they consecutively follow the training terms. It consists of an encoder to extract latent representatives, a GRU layer, which combines the advantage of both LSTM and RNN, and an auto regression layer. The workflow looks like Fig. 4.

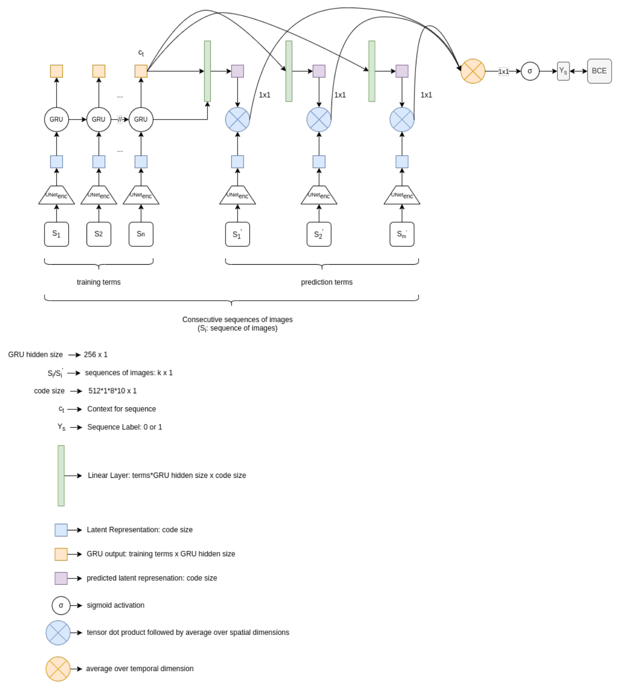

GRU
1x1
BCE
c_t
Y_s
σ
UNet enc
S_1
S_2
S_n
S_1'
S_2'
S_m'
training terms
prediction terms
Consecutive sequences of images
(S_i: sequence of images)
GRU hidden size → 256 x 1
S_i/S_i' → sequences of images: k x 1
code size → 512*1*8*10 x 1
c_t → Context for sequence
Y_s → Sequence Label: 0 or 1
Linear Layer: terms*GRU hidden size x code size
Latent Representation: code size
GRU output: training terms x GRU hidden size
predicted latent represenation: code size
sigmoid activation
tensor dot product followed by average over spatial dimensions
average over temporal dimension

Figure 4: CPC for Sequence of Image Sequences

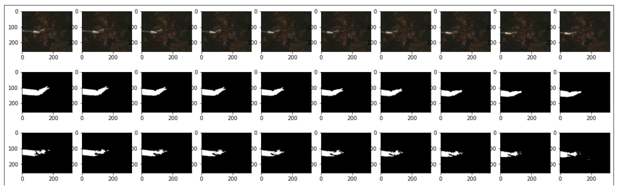


Figure 5: Sample Visualization After Fine-tuning U-Net + Both CPC networks for 30 epochs when using a CPC Image Sequence Encoder Trained on 15 Frames per Term

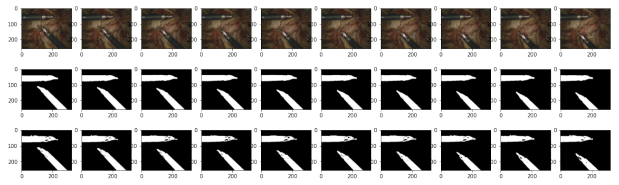


Figure 6: Sample Visualization After Fine-tuning U-Net + Both CPC networks for 30 epochs when using a CPC Image Sequence Encoder Trained on 10 Frames per Term

C CPC Kinematic Network

The CPC kinematic network follows nearly the same idea as that of the CPC image one. The only difference is that the input of each frame is 18 by 1 vector, which is viewed as 0D data of 18 channels. Therefore, the architecture of the encoder is totally different from the former, but it applies 1×1 convolution layer and two fully connected layers, connected with 1D batch normalization and activated by Leaky ReLU. The overall workflow is demonstrated in Fig. 7 and the parameters are summarized in Fig. 8.

For the hyper parameters in the CPC kinematic unsupervised training, the frame per term is set 10, input term is 3 and predict term is also 3. After 20 epochs, the loss is almost converged.

D Tissue Occlusion

Our random tissue occlusion (Fig. 2) in the video can be described in two steps: firstly, randomly select a start frame, flip vertically/horizontally/both, and translate the center square portion of the tissue only using the mask; secondly, replace the original tool content with a 90 degree rotated tissue if the result above lands on part of the tool. This way the new content only falls in the tool portion and doesn't introduce any artifact in the background, very similar to the real scenario in the surgery.

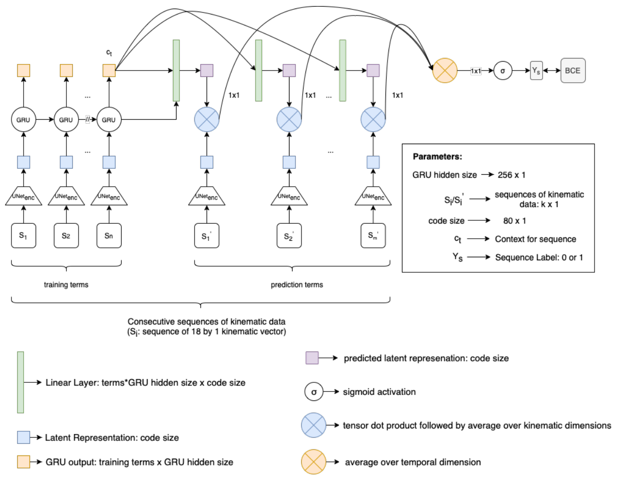
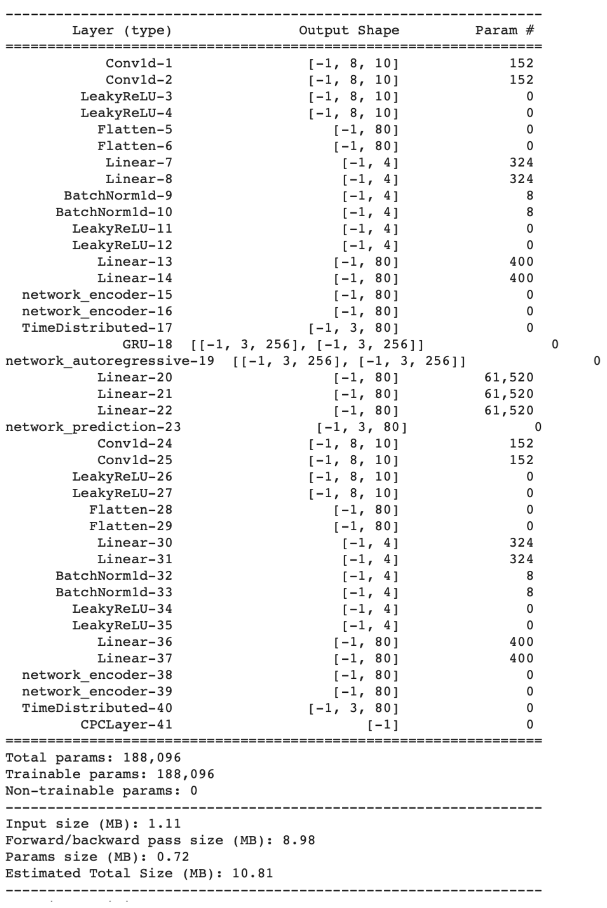

c_t
GRU
GRU
GRU
UNet_enc
S_1
S2
Sn
S_1'
S_2'
S_m'
1x1
1x1
1x1
1x1
σ
Y_s
BCE
Parameters:
GRU hidden size → 256 x 1
S_i/S_i' → sequences of kinematic data: k x 1
code size → 80 x 1
c_t → Context for sequence
Y_s → Sequence Label: 0 or 1
training terms
prediction terms
Consecutive sequences of kinematic data
(S_i: sequence of 18 by 1 kinematic vector)
Linear Layer: terms*GRU hidden size x code size
Latent Representation: code size
GRU output: training terms x GRU hidden size
predicted latent represenation: code size
sigmoid activation
tensor dot product followed by average over kinematic dimensions
average over temporal dimension

Figure 7: CPC for Sequence of Kinematic Sequences



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv1d-1                [-1, 8, 10]             152
            Conv1d-2                [-1, 8, 10]             152
         LeakyReLU-3                [-1, 8, 10]               0
         LeakyReLU-4                [-1, 8, 10]               0
           Flatten-5                   [-1, 80]               0
           Flatten-6                   [-1, 80]               0
            Linear-7                    [-1, 4]             324
            Linear-8                    [-1, 4]             324
       BatchNorm1d-9                    [-1, 4]               8
      BatchNorm1d-10                    [-1, 4]               8
       LeakyReLU-11                     [-1, 4]               0
       LeakyReLU-12                     [-1, 4]               0
          Linear-13                    [-1, 80]             400
          Linear-14                    [-1, 80]             400
 network_encoder-15                    [-1, 80]               0
 network_encoder-16                    [-1, 80]               0
 TimeDistributed-17                [-1, 3, 80]               0
             GRU-18  [[-1, 3, 256], [-1, 3, 256]]               0
network_autoregressive-19  [[-1, 3, 256], [-1, 3, 256]]               0
          Linear-20                    [-1, 80]          61,520
          Linear-21                    [-1, 80]          61,520
          Linear-22                    [-1, 80]          61,520
network_prediction-23                [-1, 3, 80]               0
          Conv1d-24                [-1, 8, 10]             152
          Conv1d-25                [-1, 8, 10]             152
       LeakyReLU-26                [-1, 8, 10]               0
       LeakyReLU-27                [-1, 8, 10]               0
         Flatten-28                   [-1, 80]               0
         Flatten-29                   [-1, 80]               0
          Linear-30                    [-1, 4]             324
          Linear-31                    [-1, 4]             324
     BatchNorm1d-32                    [-1, 4]               8
     BatchNorm1d-33                    [-1, 4]               8
       LeakyReLU-34                    [-1, 4]               0
       LeakyReLU-35                    [-1, 4]               0
          Linear-36                   [-1, 80]             400
          Linear-37                   [-1, 80]             400
 network_encoder-38                   [-1, 80]               0
 network_encoder-39                   [-1, 80]               0
 TimeDistributed-40                [-1, 3, 80]               0
        CPCLayer-41                       [-1]               0
================================================================
Total params: 188,096
Trainable params: 188,096
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.11
Forward/backward pass size (MB): 8.98
Params size (MB): 0.72
Estimated Total Size (MB): 10.81
----------------------------------------------------------------
```

Figure 8: Summary of CPC Kinematic Networks