ENDOSCOPY ARTEFACT DETECTION AND SEGMENTATION USING DEEP CONVOLUTIONAL NEURAL NETWORK

Haijian Chen, Chenyu Lian, Liansheng Wang

Department of Computer Science, School of Informatics, Xiamen University, China

ABSTRACT

Endoscopy Artefact Detection and Segmentation (EAD2020) includes 3 sub-tasks: Multi-class artefact detection, Semantic segmentation and Out-of-sample generalisation. This manuscript summarizes our solution. The challenge can be considered as two independent problems: object detection and semantic segmentation. For the detection problem, we use Cascade R-CNN with FPN and Hyper Task Cascade. For the segmentation problem, we use DeepLab v3+ model with bce+dice loss.

1. INTRODUCTION

Endoscopy is a widely used clinical procedure for the early detection of numerous cancers. However, a major drawback of these video frames is that they are heavily corrupted with multiple artifacts. Thus, accurate detection and even segmentation of artifacts are very helpful to improve the endoscopy tools. This task aims to localise bounding boxes, predict class labels and pixel-wise segmentation of 8 different artifact classes for given frames and clinical endoscopy video clips.

2. DATASETS

The details of Endoscopy Artifact Detection and Segmentation Dataset are described well in the original papers [1, 2, 3]. The following part gives a brief analysis of EAD2020 data.

2.1. Object detection

We combine the two phases of the dataset together. As shown in Table 1, the distribution of different classes is very imbalanced. The counts of ‘blur’, ‘instrument’, and ‘blood’ are significantly smaller than others, which could become hard examples when training models. The counts of ‘specularity’ and ‘artifact’ are very big and the objects of them are very small in size. Based on the condition, we pay attention to the balance of each class when we divide 20% of data as validation set.

Class	Count	Ratio	Class	Count	Ratio
specularity	9791	36.2%	contrast	1641	6.1%
saturation	1277	4.7%	bubbles	4670	17.3%
artifact	8012	29.6%	instrument	470	1.7%
blur	684	2.5%	blood	491	1.8%

Table 1. Class distribution of detection dataset

Class	fg	bg	fg/$\sum$fg	fg/(fg+bg)
Instrument	15997225	371567134	36.39%	4.13%
Specularity	4700063	382864296	10.69%	1.21%
Artefact	4100248	383464111	9.33%	1.06%
Bubbles	8967902	378596457	20.40%	2.31%
Saturation	10190545	377373814	23.18%	2.63%

Table 2. Pixel distribution of segmentation dataset (fg: foreground, bg : background)

Size	Count	Ratio	Size	Count	Ratio
512×512	138	25.36%	Smaller	129	23.71%
1349×1079	118	21.69%	Bigger	159	29.23%
Total	544	100%			

Table 3. Image sizes of segmentation dataset (Smaller : height <800, width <700, Bigger is the contrary)

2.2. Semantic segmentation

Many ground-truth pixel values are between 0 and 255 in the dataset. After dividing all ground truth pixel values by 255 and using a threshold of 0.5 to classify foreground and background pixels, the results are shown in Table 2. Foreground pixels are significantly fewer than background pixels. The foreground pixels of different classes in ground-truth images are imbalanced as well. As shown in Table 3, the most common size of images is 512×512 and 1349×1079. The others contain different sizes. We shuffle the dataset randomly and use 20% of the data as the validation set.

3. METHODS

3.1. Object Detection

3.1.1. Model Overview

We use Cascade R-CNN [4] with ResNetXt-101 [5] backbone and FPN [6] as the neck of the model. We also train a Hyper Task Cascade model [7] with the same backbone and neck.

3.1.2. Loss

We use Cross Entropy Loss for classification. Smooth L1 Loss is utilized to improve the precision of detection.

3.1.3. Augmentation

In training data, we perform random flip, normalization and resizing. The images are resized to 512×512.

3.1.4. Implementation Details

We extract candidate bounding box with RPN (region proposal network) , and use non-maximum suppression (NMS) to filter the useful bounding-box. Observing that some small objects are ignored, NMS threshold is increased from 0.7 to 0.8. It slightly improves the recall rate and mAP. Soft-NMS [8] is applied to avoid mistakenly discard the bounding-box directly.

We use SGD optimizer with a momentum of 0.9 and a weight decay of 0.0001. In order to get better results when convergence, we add a warm-up period to make the training rate linearly increase to 0.0025 in the first 500 iterations. The network is trained for 13 epochs totally.

3.2. Semantic Segmentation

3.2.1. Model Overview

We use DeepLab V3+ network [9] with ResNet101 [5] backbone for semantic segmentation. DeepLab V3+ is an encoder-decoder network with dilation convolution. ASPP modules and decoder is implemented as the original paper does.

The output of the network is activated by sigmoid function to get the probability map, since there may be overlap among different channels of the mask. The segmentation problem is considered as multiple binary segmentation tasks.

3.2.2. Loss

We evaluated different losses, including Binary Cross Entropy, Dice Loss, Lovsz-Hinge Loss [10], and their combination. Based on the testing results discussed in 4.2, we choose bce+dice as the loss of our model at last, which simply means

$$L = L_{bce} + L_{dice} = -\, y_{gt}\log y_{pred} - (1 - y_{gt})\log y_{pred} \\ + 1 - \frac{(2\sum y_{gt} \cdot y_{pred} + \epsilon}{\sum y_{gt} + \sum y_{pred} + \epsilon})$$

($\epsilon = 10^{-7}$, y_{gt} and y_{pred} are flattened tensors)

3.2.3. Augmentation

We apply random brightness and contrast changes, random horizontal and vertical flip, random shift scale rotation, Gaussian blurring, resizing and normalization to images of the training set. All random transformations are applied by a probability of 0.5 with the default parameters of Albumentations library [11]. We apply image normalization in the validation set.

The images are resized to 512×512 and 1024×1024 during the training phase, see 4.2.

3.2.4. Implementation Details

We load the weights pre-trained on the ImageNet for the backbone network. The network is trained using SGD with a momentum of 0.9 and a weight decay of 0.0001. We train the model using mini-batches of size 4. The learning rate is increased linearly over the warm-up period of 5 epochs, to the maximum value of 0.01, then adjusted by cosine annealing with warm restarts [12] by a period of 40 epochs. The images are resized to 512×512 to train 200 epochs and then resized to 1024×1024 to train another 100 epochs.

4. RESULTS

4.1. Object Detection

Table 4 shows mAPs of different classes in the validation set, which are evaluated by COCO metrics. And Table 5 shows more details of evaluation results. Metrics of both models are pretty close to each other. In Figure 1, we find the HTC model is good at detecting large objects while doing poorly in some small objects, though its AP_small metric is slightly better than the other.

Class	Cascade R-CNN	HTC	Faster R-CNN
instrument	0.64791	**0.64965**	0.56197
artifact	**0.22540**	0.22511	0.21733
blood	0.10594	**0.12520**	0.10998
blur	**0.26506**	0.26097	0.19428
bubbles	**0.11302**	0.10491	0.10600
contrast	**0.40275**	0.39182	0.38044
saturation	**0.27912**	0.24990	0.26373
specularity	0.09281	**0.09485**	0.08561

Table 4. mAPs of different classes in validation set

Metric	Cascade R-CNN	HTC	Faster R-CNN
mAP	**0.267**	0.263	0.240
AP50	0.501	**0.505**	0.498
AP75	0.246	**0.249**	0.209
AP_small	0.082	**0.091**	0.086
AP_medium	0.162	**0.166**	**0.166**
AP_large	0.337	0.337	0.299

Table 5. AP metrics of evaluation results in validation set

Model	mAPd	IoUd	mAPg	mAPsq
Cascade R-CNN	0.2238	**0.1707**	0.2405	0.3038
HTC network	**0.2393**	0.0674	**0.2621**	**0.3214**

Table 6. Detection scores in the first phase of test data

Model	Score_d	dstd	gmAP	gdev
Cascade R-CNN	**0.2193**	**0.0871**	0.2485	0.0552
HTC network	0.2021	0.0901	**0.2744**	**0.0556**

Table 7. Detection scores in the final test

Size	Score_d	dstd	gmAP	gdev
512×512	**0.2193**	**0.0871**	0.2485	0.0552
1024×1024	0.2156	0.0991	**0.2659**	**0.0764**

Table 8. Detection scores in the final test with Cascade models trained with different sizes

The results of Table 6,7,8 are provided by the official leaderboard. Table 6 shows the detection scores in the first phase of test data. The Hybrid Task Cascade network performs better in mAP, while getting a lower score in IoU.

Table 7 shows the scores in the final test. We get a higher detection score with the Cascade R-CNN network.

As shown in Table 8, resizing the image to 1024×1024 instead of 512×512 doesn't give a better score but contributes to generalization performance.

4.2. Semantic Segmentation

4.2.1. Experiments of losses in validation set

To evaluate the results of different losses, we train a DeepLab V3+ model with ResNet101 backbone and a modified U-Net [13] model with ResNet-34 backbone for 160 epochs.

The threshold to predict foreground pixels is 0.5. Other configurations are the same as 3.2.4. In Table 9 and Table 10, 'bce is Binary Cross Entropy loss, 'dice is Dice Loss, 'bce+dice' is defined in 3.2.2. All 'p' and 'r' in the tables stand for precision and recall.

In Table 9, the experiment shows that bce+dice gets the best score in Dice, F2, and IoU score. The precision of bce+dice is pretty close to dice, while not losing much recall. In Table 10, we can see a significant improvement of UNet using bce+dice, showing the effectiveness of this loss.

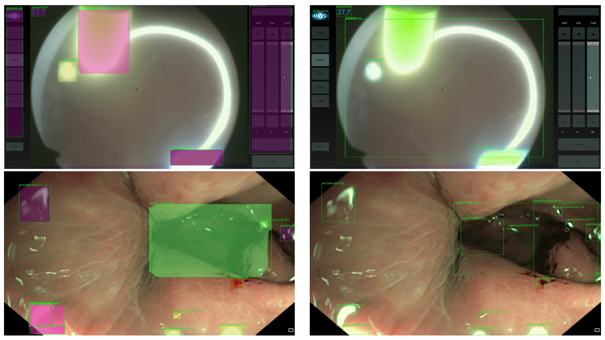
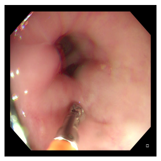
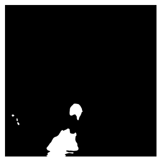


Fig. 1. Predictions of two images (left: Hybrid Task Cascade; right: Cascade R-CNN)





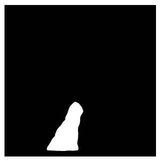


Fig. 2. Test image **Fig. 3**. Pred-512 **Fig. 4**. Pred-1024

After 300 epochs, the DeepLabV3+ model using bce+dice got 0.7927 in F1, 0.8386 in F2, 0.6857 in IoU, 0.7422 in precision and 0.887 in recall. The U-Net models don't get much better scores as they almost converge after 165 epochs.

We also tested Lovsz-Hinge loss. In our test, it is hard to converge if the model is trained from the ground up. Hence, we use Lovsz-Hinge loss to fine-tune the Deeplab model trained with bce+dice for 300 epochs. Table 11 shows the results of the first 20 epochs (Epochs means the training epochs with Lovsz). This model converges after 30 epochs but these results are worse than the model before fine-tuning, so we give up this method.

Metric	F1	F2	IoU	p	r
bce	0.585	0.6754	0.4447	0.5014	**0.8136**
dice	0.5846	0.5881	0.4874	**0.6755**	0.601
bce+dice	**0.6728**	**0.7042**	**0.5523**	0.6699	0.7585

Table 9. The metrics of the validation set with different losses in DeepLab V3+ model with ResNet101 backbone (160 epochs)

Metric	F1	F2	IoU	p	r
bce	0.4666	0.4346	0.3743	0.7201	0.4196
dice	0.561	0.5421	0.469	0.7188	0.5353
bce+dice	**0.6138**	**0.5837**	**0.5057**	**0.7415**	**0.5709**

Table 10. The metrics of the validation set with different losses in a modified U-Net model with ResNet-34 backbone (160 epochs)

We choose bce+dice to train the final model.

Epochs	F1	F2	IoU	p	r
5	0.5491	0.5038	0.4486	0.8490	0.4828
20	0.5115	0.4610	0.4173	0.8927	0.4367
40	0.5373	0.4846	0.4387	0.8872	0.4594

Table 11. Using Lovsz-Hinge loss to fine-tune a model trained with BCE + Dice loss

4.2.2. Experiments of backbones in validation set

Table 12 shows another experiment to compare different networks. We find that Xception-based DeepLabV3+ converges significantly slower than ResNet101-based model, and does not get better scores than the U-Net model.

Model	F1	F2	IoU	p	r
D-X	0.4189	0.4248	0.3167	0.486	0.4388
D-R101	**0.5823**	**0.5967**	**0.4717**	0.6288	**0.6313**
U-R34	0.5535	0.5209	0.4507	**0.7512**	0.5078

Table 12. The metrics of validation set with different network (85 epochs, D: DeeplabV3+, U:UNet, X:Xception, R:ResNet)

4.2.3. Submission results

	F1	F2	p	r	sscore	sd
1	0.4872	0.5027	0.5250	0.5467	0.5154	**0.2327**
2	0.4802	**0.5156**	0.4836	**0.5872**	0.5167	0.2403
3	**0.5012**	0.5042	**0.5817**	0.5390	**0.5315**	0.2644

Table 13. Segmentation scores in the first phase of test data (50% of final data)

Model	sscore	sstd
3 : DeepLabV3+/ResNet101/1024x	0.5459	0.2682

Table 14. Segmentation scores in the final test

The training parameters are listed in 3.2.4. All the results above are provided by the official leaderboard. In Table 13, Model 1 is trained with 512×512 images and a threshold of 0.5. Model 2 is the same as model 1 except changing the threshold to 0.7. Model 3 is trained with 1024×1024 images and a threshold of 0.7.

We resized the image to 512×512 at first. However, as discussed in Table 3, there are many bigger images. This can be found in the first phase of test images as well. Compared with the models only trained with images resized to 512×512, models trained with 1024×1024 get better scores in the validation set. Some predictions look smoother, as shown in Figure 2,3,4.

We find that adding segmentation data of EAD2019 to the training set also helps a little, although there is potential validation data leakage, making validation metrics unbelievable. However, it does not help in the detection task.

We chose Model 3 to predict the final test data and got scores as Table 14 shows.

5. DISCUSSION & CONCLUSION

In task 1, we compare Cascade R-CNN with Hyper Task Cascade to get a better detection model. FPN and Soft-NMS are used to improve the detection precision due to class imbalance. A proper threshold of NMS is helpful to improve the recall rate of small objects.

In task 2, we select DeepLabV3+ to solve the problem. We select bce+dice as the loss function to balance precision and recall. Image sizes of the dataset is a noticeable part at the training phase. Adjusting the threshold of predicting also contributes to a more balanced model.

6. REFERENCES

[1] Sharib Ali, Felix Zhou, Christian Daul, Barbara Braden, Adam Bailey, Stefano Realdon, James East, Georges Wagnieres, Victor Loschenov, Enrico Grisan, et al. Endoscopy artifact detection (ead 2019) challenge dataset. *arXiv preprint arXiv:1905.03209*, 2019.

[2] Sharib Ali, Felix Zhou, Adam Bailey, Barbara Braden, James East, Xin Lu, and Jens Rittscher. A deep learning framework for quality assessment and restoration in video endoscopy. *arXiv preprint arXiv:1904.07073*, 2019.

[3] Sharib Ali, Felix Zhou, Barbara Braden, Adam Bailey, Suhui Yang, Guanju Cheng, Pengyi Zhang, Xiaoqiong Li, Maxime Kayser, Roger D. Soberanis-Mukul, Shadi Albarqouni, Xiaokang Wang, Chunqing Wang, Seiryo Watanabe, Ilkay Oksuz, Qingtian Ning, Shufan Yang, Mohammad Azam Khan, Xiaohong W. Gao, Stefano Realdon, Maxim Loshchenov, Julia A. Schnabel, James E. East, Geroges Wagnieres, Victor B. Loschenov, Enrico Grisan, Christian Daul, Walter Blondel, and Jens Rittscher. An objective comparison of detection and segmentation algorithms for artefacts in clinical endoscopy. *Scientific Reports*, 10, 2020.

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. *CoRR*, abs/1712.00726, 2017.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[6] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[7] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. *CoRR*, abs/1901.07518, 2019.

[8] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Improving object detection with one line of code. *CoRR*, abs/1704.04503, 2017.

[9] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation, 2018.

[10] Maxim Berman, Amal Rannen Triki, and Matthew B. Blaschko. The lovsz-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks, 2017.

[11] E. Khvedchenya V. I. Iglovikov A. Buslaev, A. Parinov and A. A. Kalinin. Albumentations: fast and flexible image augmentations. *ArXiv e-prints*, 2018.

[12] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention –MICCAI 2015*, pages 234–241, 2015.