NORDIC MACHINE INTELLIGENCE

Nordic Machine Intelligence, MedAI 2021
https://doi.org/10.5617/nmi.9130

Kvasir-Instruments and Polyp Segmentation Using UNet

Sumit Pandey[1], Arvind Keprate[2 3]

1. *Department of Medical Imaging and Intervention, Chang Gung Memorial Hospital, Linkou, Taiwan*
2. *OsloMet – Oslo Metropolitan University, Oslo, Norway*
3. *E-mail any correspondence to: arvindke@oslomet.no*

Abstract

This paper describes the methodology used to develop, fine-tune, and analyze a UNet-based model for generating segmentation masks for the polyp and instrument segmentation tasks held at MedAI 2021. We used the same methodology on both tasks, where the evaluation on the hidden testing dataset resulted in an IOU of 0.73 and dice score of 0.7980 for the instrumentation task, and an IOU of 0.41 and dice score of 0.41 for the polyp segmentation task.

Keywords: UNet; segmentation; deep learning; polyp; instrumentation

Introduction

Over the last few years, the use of deep learning for medical image segmentation (MIS) has gained a lot of interest amongst the medical community. MIS is characterized as a complicated task due to factors, such as data complexity, the complexity of the objects of interest, and complex validation [1].

Compared to classical machine learning and computer vision techniques, deep learning offers higher segmentation accuracy and speed when it comes to MIS [2]. In particular, fully connected networks, generative adversarial networks (GAN), and U-Net have emerged as the most commonly used models for the MIS task. This paper summarizes our work to develop deep learning models for the instrument and polyp segmentation tasks as part of the MedAI 2021 [3].

We used the U-Net model for both segmentation tasks (polyp and instrument segmentation task) and the details of the materials and method is presented in the following.

Materials and methods

The methodology used for training and testing of the U-Net model has been divided into three parts:

Data pre-processing

The datasets used in this paper are obtained from Simula open datasets (Kvasir-Instrument Dataset [4] and Polyp Dataset [5]). Both datasets were divided into two parts: Development set and testing set with the ratio of 80:20. Before the training, the image data were at first resized into 256*256 and then normalized between 0 and 1.

UNet Model

The U-Net (as shown in Figure 1) architecture based on a fully convolution neural network. In this work its architecture was modified and extended to work with fewer training images and produces the output of the same size as input.

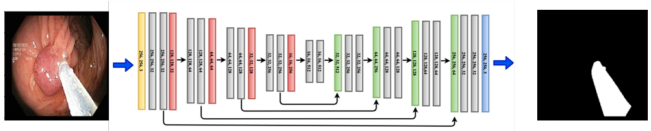

Figure 1: UNet architecture used in this work.

Initialization of model parameters and weights

The weights of the U-Net models were randomly initiated. During training, the hyperparameters were setup using methods as described in the following.

1. **Learning rate:** The learning rate determines how fast or slow the model will learn the task. In this work, the learning rate is regulated by TensorFlow's [6] '*reduceLROnPlat*' function. In this function, we define the following input parameters: '*monitor*': it continuously monitors the validation loss. At first, we define the lowest learning rate, which is 106 in our study for U-net. If the Dice similarity coefficient loss rate was not changed for 3 continuous epochs (for 5 decimal points), then the learning rate was reduced by the factor 0.05.

2. **Number of Epochs:** The number of epochs determines how many times the data will be passed through the model. In this work, it was determined by (TensorFlow's) '*EarlyStopping*' function. If the validation loss does not change for continuous 15 epochs, the model stops training. For UNet the number of epochs is 1000.

3. **Batch Size:** Batch size indicates how the data will feed to the model during training. The batch size for both works was set by using training the models iteratively on different batch sizes. Due to memory limitations, the UNet model cannot be trained for more than 12 batch sizes. So based on performances, we chose 8 as the best batch size.

4. **Activation Function:** In both tasks ReLU activation function was used in the hidden layers.

5. **Optimizer:** In order to minimize the loss, we used Adam optimizer for both tasks.

6. **Loss Function:** The loss function used in this work is the negative Dice coefficient (DSC).

Results

Figure 2, shows two plots between DSC loss vs number of epochs: (a), is for instruments task, and (b), is for Polyp task. It is evident from both plots that the model is neither over-fitting nor over-fitting.For the instrumentation task, the training was stopped by the learning rate scheduler when the loss was same for 15 epochs, the DSC for training and validation is 0.8. And for the polyp task the training by the training was stopped by the learning rate scheduler when the loss was same for 15 epochs, the DSC for training and validation is 0.65.

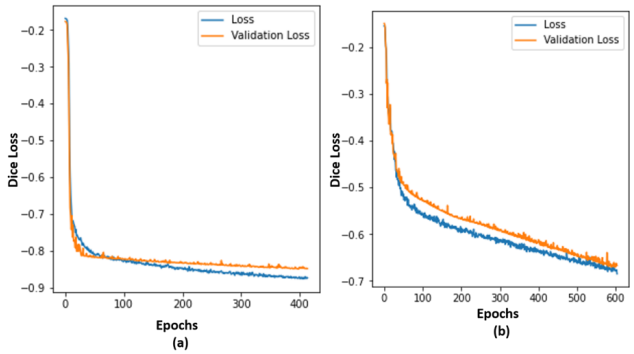

Figure 2: (a). For Kvasir-Instruments: AUC curve for training and validation data, (b). For Kvasir-Polyp AUC curve for training and validation data.

Table 1 shows the model's performance on Acc.: Accuracy, Jac.: jaccard, Dice: Dice Coeff., F1.:F1 Score, Rec.: Recall, Prec.: Precision, for the testing set of Kvasir-Instruments and Polyp data. It is evident that model has performed better on instrument dataset, the average dice score on instrument dataset is 0.80 whereas on Ployp dataset it is 0.41.

Table 1: Accuracy, Jaccard, Dice Coeff., Recall, F1 score, Precision for the testing set of Kvasir-Instruments and Polyp data

Data	Acc.	Jac.	Dice.	F1.	Rec.	Prec.
Instr.	0.97	0.73	0.80	0.80	0.79	0.85
Polyp.	0.56	0.29	0.41	0.41	0.71	0.49

Discussion

In this work, we established a methodology using UNet for generating the segmentation masks for the instrumentation and polyp datasets. The fair advantage of this methodology is that it automatically fine-tunes the learning on a defined range. Hence the model does not overfit on testing validation dataset, as shown in Figure 2.

If we closely look at Table 1 and Figure 2, for instruments dataset: the model performed well (dice score is around 0.8, as shown in Figure 2(a)) on training, validation and for testing dataset (dice score is 0.7980, as shown in Table 2). While in case of the polyp dataset(as shown in Figure 2 (b)), the model also does not look overfit (DSC is around 0.6 for both sets) but by analyzing Table 1, it is evident that the model was failed to perform segmentation task on testing dataset the DSC score was only 0.41.

By analyzing the model on both datasets, we conclude that the model performed well on the instrument segmentation task. The reason behind this is quite apparent: the features (color, pixels intensity, etc.) of instruments are different than the skin, so it was an easier task. While in the case of the polyp segmentation task, the region of interest (ROI) was on the skin, making it quite difficult for the UNet model to distinguish between ROI and skin (due to similar pixels features). A solution to this problem could be using other models like deeplabv3 or Pix2Pix-GAN for segmentation as future work.

References

1. Frederik Maes David Robben DV and Suetens P. The Role of Medical Image Computing and Machine Learning in Healthcare. Artificial Intelligence in Medical Imaging 2019; 1:9–23
2. Liu X. Song L. LS and Y. Z. A Review of Deep-Learning-Based Medical Image Segmentation Methods. Sustainability 2021 2021; 13:12–24. DOI: https://doi.org/10.3390/su13031224
3. MED-AI:Transparency in Medical Image Segmentation. Nordic Machine Intelligence. 2021 Oct 12. Available from: https://www.nora.ai/Competition/image-segmentation.html [Accessed on: 2021]
4. Jha D, Ali S, Emanuelsen K, Hicks SA, Thambawita V, Garcia-Ceja E, Riegler MA, Lange T de, Schmidt PT, Johansen HD, Johansen D, and Halvorsen P. Kvasir-Instrument: Diagnostic and Therapeutic Tool Segmentation Dataset in Gastrointestinal Endoscopy. *MultiMedia Modeling*. 2021 :218–29
5. Jha D, Smedsrud PH, Riegler MA, Halvorsen P, Lange T de, Johansen D, and Johansen HD. Kvasir-seg: A segmented polyp dataset. *International Conference on Multimedia Modeling*. Springer. 2020 :451–62
6. tf.keras.callbacks.LearningRateScheduler. Tensorflow. 2021 Oct 26. Available from: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/LearningRateScheduler [Accessed on: 2021]