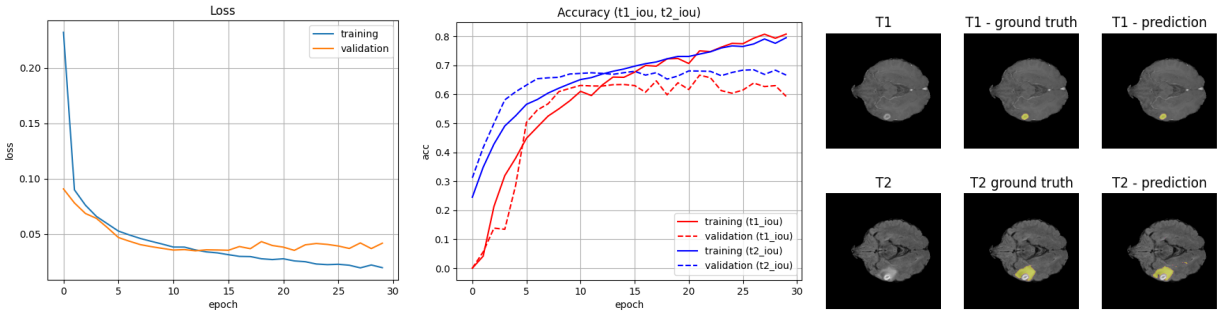
Project 1: Image Segmentation

Aanchal Sahu (ID: 3417105)
Takahiro Manabe (ID: 3550620)

Step 1. U-Net Implementation by Takahiro Manabe

Figure 1 (a) and (b) show the loss and intersection-over-union (IoU) curves of training and validation on the model (structure is shown in Figure A1). The number of trainable parameters was 1884227, the same amount as the sample code (there was no non-trainable parameter). We can see convergence around the 10th epoch on both loss and IoU curves, achieving around 60% and 65% training accuracy, T1 and T2 respectively. The test result from the trained model (Table 1) achieves lower accuracy than validation; 51.91% and 63.07%. (c) highlights the prediction and true labels.



(a) (b) (c)

Figure 1. (a) the loss curve, (b) the intersection-over-union (IoU) curves of T1 and T2 classes. (c) An example of the highlight of prediction and true label. The pair on the left side shows the original dataset image, and the middle shows the true labels. The pair on the right side shows the label predicted from the trained model.

Table 1. A test result of U-Net-like model

	T1 IoU [%]	**T2 IoU [%]**
Evaluation (Baseline)	51.91	63.07

Step 2. Modification by Takahiro Manabe

・Test-Time Augmentation (TTA)

Table 2 shows the evaluation result of the test data between with and without Test-Time Augmentation (TTA). TTA enables to improve the accuracy of the evaluation in both T1 and T2 classes, which is especially significant in T2. TTA is based on the evaluation from the smoothed (=averaged) predictions of the various transformed versions of the data set after each prediction. The average predictions from various versions of test data are robust [2] to erroneous predictions from a single version, improving accuracy.

Table 2. The IoU accuracy with TTA

	T1 IoU [%]	**T2 IoU [%]**
Evaluation (TTA)	52.78	66.31

Step 1: U-Net Implementation by Aanchal Sahu

The following graphs depict the training progress over 30 epochs. In Fig. 1(a), the Training Loss starts high and decreases sharply, indicating the model is learning effectively from the training data. In Fig. 1(b), the model's accuracy is measured for T1 and T2. For both T1 and T2, the accuracy on the training data increases over epochs. The model shows good generalization for T1, as indicated by the close proximity of training and validation lines. For T2, there's a gap between training and validation, hinting at some possibility of overfitting. Fig. 2 shows the prediction for one training set sample. Fig. A1 in the appendix shows the model summary for reference. The model has a total of 7825475 parameters with 0 non-trainable parameters. Accuracy for T1 IoU is 57.74% and T2 IoU is 63.48%.

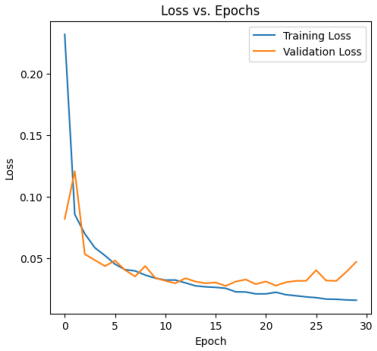


Fig. 1(a)

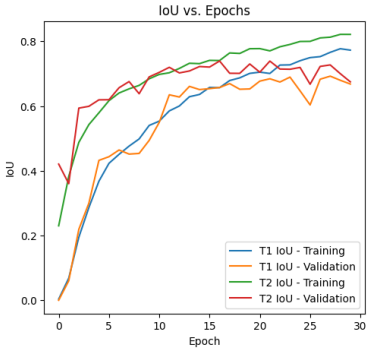


Fig. 1(b)

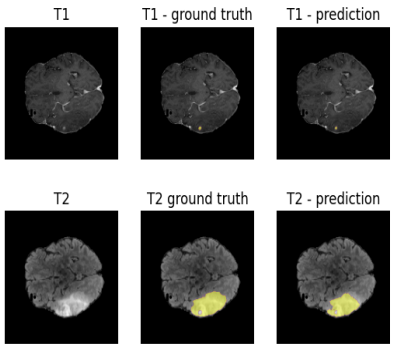


Fig. 2.

Step 2: Data Augmentation by Aanchal Sahu

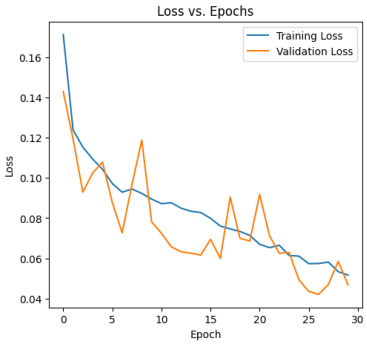


Fig. 3(a)

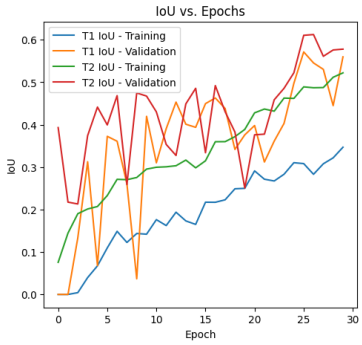


Fig 3(b)

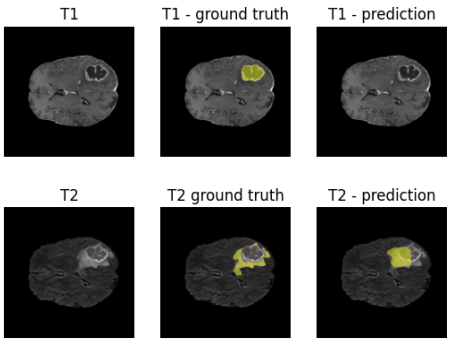


Fig 4.

Figures 3(a) and 3(b) highlight the model's performance drop when using data augmentation. The diminished accuracy suggests the model is challenged by augmented image inconsistencies. Through testing various techniques, image contrast and orientation emerged as significant factors (Fig. A2 shows results with contrast and translation changes with significantly dropped accuracy). Figure 4 displays predictions for a training sample. With parameters consistent to Step 1, the IoU for T1 reached 53.45%, and T2 was at 56.88%, with random horizontal flipping during training.

Appendix by Aanchal Sahu

```
Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_2 (InputLayer)        [(None, 240, 240, 2)]     0         []

 lambda_1 (Lambda)           (None, 240, 240, 2)       0         ['input_2[0][0]']

 c1_0 (Conv2D)               (None, 240, 240, 16)      304       ['lambda_1[0][0]']

 c1_1 (Conv2D)               (None, 240, 240, 16)      2320      ['c1_0[0][0]']

 max_pooling2d_4 (MaxPoolin  (None, 120, 120, 16)      0         ['c1_1[0][0]']
 g2D)

 c2_0 (Conv2D)               (None, 120, 120, 32)      4640      ['max_pooling2d_4[0][0]']

 c2_1 (Conv2D)               (None, 120, 120, 32)      9248      ['c2_0[0][0]']

 max_pooling2d_5 (MaxPoolin  (None, 60, 60, 32)        0         ['c2_1[0][0]']
 g2D)

 c3_0 (Conv2D)               (None, 60, 60, 64)        18496     ['max_pooling2d_5[0][0]']

 c3_1 (Conv2D)               (None, 60, 60, 64)        36928     ['c3_0[0][0]']

 max_pooling2d_6 (MaxPoolin  (None, 30, 30, 64)        0         ['c3_1[0][0]']
 g2D)

 c4_0 (Conv2D)               (None, 30, 30, 128)       73856     ['max_pooling2d_6[0][0]']

 c4_1 (Conv2D)               (None, 30, 30, 128)       147584    ['c4_0[0][0]']

 max_pooling2d_7 (MaxPoolin  (None, 15, 15, 128)       0         ['c4_1[0][0]']
 g2D)

 b_0 (Conv2D)                (None, 15, 15, 256)       295168    ['max_pooling2d_7[0][0]']

 b_1 (Conv2D)                (None, 15, 15, 256)       590080    ['b_0[0][0]']

 up_sampling2d_4 (UpSamplin  (None, 30, 30, 256)       0         ['b_1[0][0]']
 g2D)

 concatenate_4 (Concatenate  (None, 30, 30, 384)       0         ['up_sampling2d_4[0][0]',
 )                                                                 'c4_1[0][0]']

 c5_0 (Conv2D)               (None, 30, 30, 512)       1769984   ['concatenate_4[0][0]']

 c5_1 (Conv2D)               (None, 30, 30, 512)       2359808   ['c5_0[0][0]']

 up_sampling2d_5 (UpSamplin  (None, 60, 60, 512)       0         ['c5_1[0][0]']
 g2D)

 concatenate_5 (Concatenate  (None, 60, 60, 576)       0         ['up_sampling2d_5[0][0]',
 )                                                                 'c3_1[0][0]']

 c6_0 (Conv2D)               (None, 60, 60, 256)       1327360   ['concatenate_5[0][0]']

 c6_1 (Conv2D)               (None, 60, 60, 256)       590080    ['c6_0[0][0]']

 up_sampling2d_6 (UpSamplin  (None, 120, 120, 256)     0         ['c6_1[0][0]']
 g2D)

 concatenate_6 (Concatenate  (None, 120, 120, 288)     0         ['up_sampling2d_6[0][0]',
 )                                                                 'c2_1[0][0]']

 c7_0 (Conv2D)               (None, 120, 120, 128)     331904    ['concatenate_6[0][0]']

 c7_1 (Conv2D)               (None, 120, 120, 128)     147584    ['c7_0[0][0]']

 up_sampling2d_7 (UpSamplin  (None, 240, 240, 128)     0         ['c7_1[0][0]']
 g2D)

 concatenate_7 (Concatenate  (None, 240, 240, 144)     0         ['up_sampling2d_7[0][0]',
 )                                                                 'c1_1[0][0]']

 c8_0 (Conv2D)               (None, 240, 240, 64)      83008     ['concatenate_7[0][0]']

 c8_1 (Conv2D)               (None, 240, 240, 64)      36928     ['c8_0[0][0]']

 conv2d_1 (Conv2D)           (None, 240, 240, 3)       195       ['c8_1[0][0]']

==================================================================================================
Total params: 7825475 (29.85 MB)
Trainable params: 7825475 (29.85 MB)
Non-trainable params: 0 (0.00 Byte)
```

Fig. A1. Model summary of the baseline model

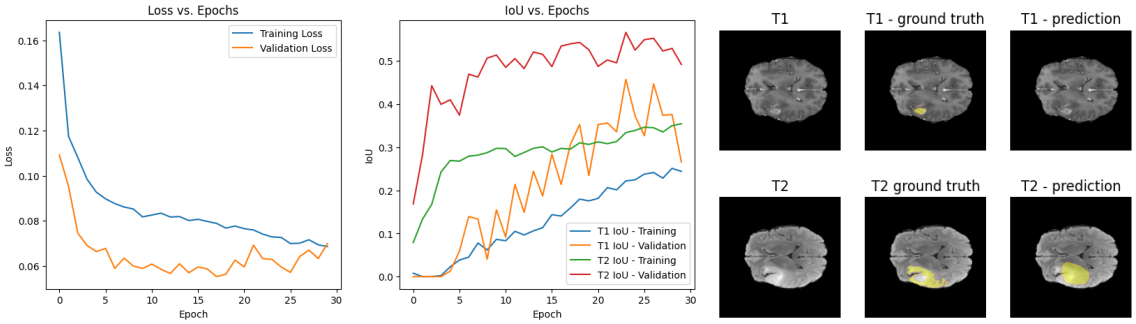


Fig. A2. Results with contrast and translation changes

Appendix by Takahiro Manabe

A. Supplement Figure for Step 1

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 240, 240, 2)]	0	[]
lambda (Lambda)	(None, 240, 240, 2)	0	['input_1[0][0]']
1_0 (Conv2D)	(None, 240, 240, 64)	1216	['lambda[0][0]']
1_1 (Conv2D)	(None, 240, 240, 64)	36928	['1_0[0][0]']
max_pooling2d (MaxPooling2D)	(None, 120, 120, 64)	0	['1_1[0][0]']
2_0 (Conv2D)	(None, 120, 120, 128)	73856	['max_pooling2d[0][0]']
2_1 (Conv2D)	(None, 120, 120, 128)	147584	['2_0[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 60, 60, 128)	0	['2_1[0][0]']
3_0 (Conv2D)	(None, 60, 60, 256)	295168	['max_pooling2d_1[0][0]']
3_1 (Conv2D)	(None, 60, 60, 256)	590080	['3_0[0][0]']
up_sampling2d (UpSampling2D)	(None, 120, 120, 256)	0	['3_1[0][0]']
concatenate (Concatenate)	(None, 120, 120, 384)	0	['2_1[0][0]', 'up_sampling2d[0][0]']
4_0 (Conv2D)	(None, 120, 120, 128)	442496	['concatenate[0][0]']
4_1 (Conv2D)	(None, 120, 120, 128)	147584	['4_0[0][0]']
up_sampling2d_1 (UpSampling2D)	(None, 240, 240, 128)	0	['4_1[0][0]']
concatenate_1 (Concatenate)	(None, 240, 240, 192)	0	['1_1[0][0]', 'up_sampling2d_1[0][0]']
5_0 (Conv2D)	(None, 240, 240, 64)	110656	['concatenate_1[0][0]']
5_1 (Conv2D)	(None, 240, 240, 64)	36928	['5_0[0][0]']
conv2d (Conv2D)	(None, 240, 240, 3)	1731	['5_1[0][0]']

Total params: 1884227 (7.19 MB)
Trainable params: 1884227 (7.19 MB)
Non-trainable params: 0 (0.00 Byte)

Figure A1. U-Net-like model structure (Baseline model). Up-sampling layers don't have any trainable parameters.

B. Supplement Document and figure for Test-Time Augmentation (TTA) in Step 2

1. Method

Test-Time Augmentation (TTA) is a method that applies data augmentation to the test data set, instead of the training dataset, to improve the accuracy and stability of the evaluation of the model.

The methods for image modification in this experiment inevitably could not include irreversible methods such as image shifting and zooming; unlike normal classification methods, in segmentation classification, which classifies each pixel in the data, the original position of each pixel must be memorized for the averaging process that is following each prediction of transformed versions (image modification should be reverted after prediction). In this experiment, we followed the patterns of data augmentation methods introduced by Moshkov et al. [1]. Preparing the original test data, 5 versions of the test data (all reversible processes) were first created: 1. 90-degree rotation, 2. 180-degree rotation, 3. 270-degree rotation, 4. vertical flip, and 5. parallel flip, either of which is revertable, and each data set version was predicted by the trained model.

Then, each version of the dataset was fed into the model for prediction (model.predict); calculated probability was stored in each version as an NxHxWxC tensor. Furthermore, the pixels of tensors of the modified versions were reverted to the original position. Lastly, the average probability was calculated along all versions and then classified manually into the three classes (background, T1, T2) referring to the highest probability per pixel across three channels, as shown in Figure B1.

Since we wanted to evaluate the improvement of the model's evaluation accuracy in TTA only, we did not perform training data augmentation (or simply Data Augmentation), which is generally performed in parallel.

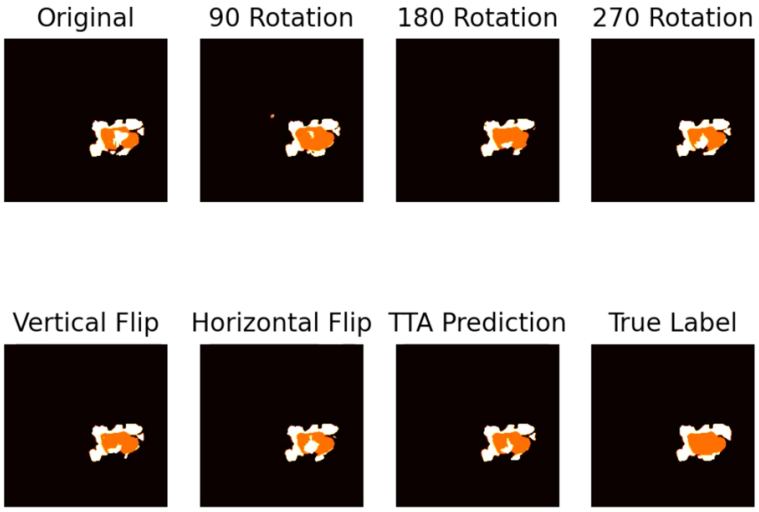


Figure B1. Examples of segmentation labeling predicted by the model. From the top left, the original image, 90-degree rotation, 180-degree rotation, 270-degree rotation, vertical flip, and parallel flip. Animation along the data samples is available on the ipynb file.

2. Results Supplement & Discussion

In Figure B1, the predictions (TTA) are made by averaging the probabilities of all 6 versions, and the true labels corresponding to this data sample are also shown. Some images feature out of the original and the five processed images are shown differently from the true label; however, by taking the average (TTA), the prediction could be observed as more similar to the true label.

To maximize this improvement with TTA, several ideas can be proposed. The first is to add elastic deformation as one of the data augmentation methods, as conducted in the U-Net paper [3] as a version of image deformation, since deformation is the most common change in biological cellular tissue, elastic deformation is a very important technique in segmentation classification in biomedical images because it can mimic deformation that can occur in real life [3]. Although our dataset was based on an MRI of the brain and hence elastic deformation may generate unrealistic shapes of tumors, considering the report from Chaitanya et al. [4] showing that visually unrealistic synthetic images from elastic deformation can improve cardiac MRI segmentation, it is possible that the elastic deformation can improve our classification of MR images of brain tumors [5].

C. Step 2 (Additional):

・Trainable up-convolutions

Here, we also implemented another U-Net-like model where up-sampling layers are displaced by a trainable up-convolutional layer. The model structure is shown in Figure C1. The number of parameters has increased to 2212291.

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 240, 240, 2)]	0	[]
lambda (Lambda)	(None, 240, 240, 2)	0	['input_1[0][0]']
1_0 (Conv2D)	(None, 240, 240, 64)	1216	['lambda[0][0]']
1_1 (Conv2D)	(None, 240, 240, 64)	36928	['1_0[0][0]']
max_pooling2d (MaxPooling2D)	(None, 120, 120, 64)	0	['1_1[0][0]']
2_0 (Conv2D)	(None, 120, 120, 128)	73856	['max_pooling2d[0][0]']
2_1 (Conv2D)	(None, 120, 120, 128)	147584	['2_0[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 60, 60, 128)	0	['2_1[0][0]']
3_0 (Conv2D)	(None, 60, 60, 256)	295168	['max_pooling2d_1[0][0]']
3_1 (Conv2D)	(None, 60, 60, 256)	590080	['3_0[0][0]']
up_conv2d (UpConv2D)	(None, 120, 120, 256)	262400	['3_1[0][0]']
concatenate (Concatenate)	(None, 120, 120, 384)	0	['2_1[0][0]', 'up_conv2d[0][0]']
4_0 (Conv2D)	(None, 120, 120, 128)	442496	['concatenate[0][0]']
4_1 (Conv2D)	(None, 120, 120, 128)	147584	['4_0[0][0]']
up_conv2d_1 (UpConv2D)	(None, 240, 240, 128)	65664	['4_1[0][0]']
concatenate_1 (Concatenate)	(None, 240, 240, 192)	0	['1_1[0][0]', 'up_conv2d_1[0][0]']
5_0 (Conv2D)	(None, 240, 240, 64)	110656	['concatenate_1[0][0]']
5_1 (Conv2D)	(None, 240, 240, 64)	36928	['5_0[0][0]']
conv2d (Conv2D)	(None, 240, 240, 3)	1731	['5_1[0][0]']

Total params: 2212291 (8.44 MB)
Trainable params: 2212291 (8.44 MB)
Non-trainable params: 0 (0.00 Byte)

Figure C1. U-Net-like model structure with Up-convolutional layers. Up-convolutional layers increase additional trainable parameters in the model.

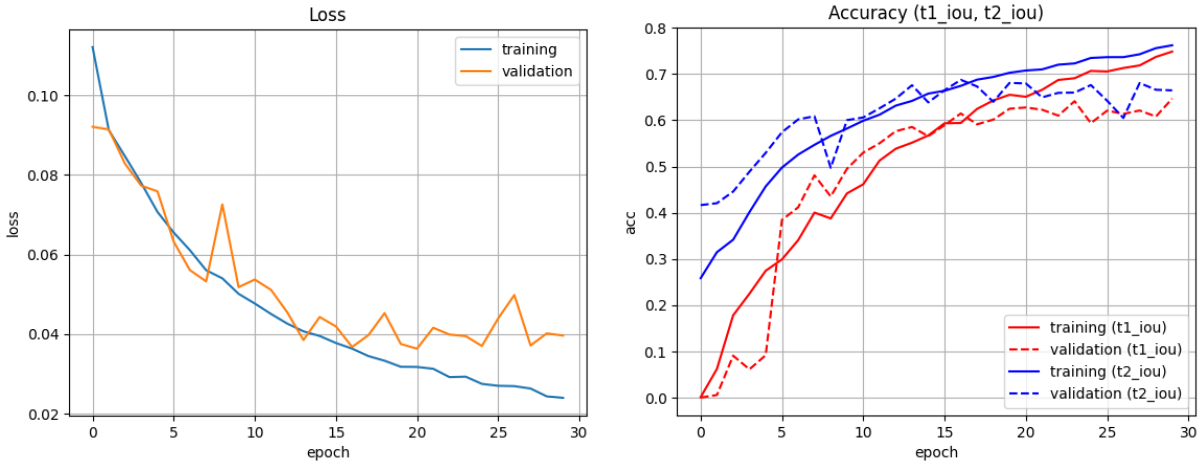


Figure C2. (b) the loss curve, (b) the intersection-over-union (IoU) curves of T1 and T2 classes.

Figure C2 shows the loss curve of training and validation on the U-Net model with up-convolutional layers. Compared to the result of the baseline model (Figure A1), the curve is taking longer to get its convergence, which is around the 15th epoch. Yet, the validation accuracy hasn't improved significantly. In addition, looking at the test result from Table C1, the T1 increases its validation accuracy; however, the T2 IoU test accuracy was decreased after replacing the up-sampling layer with up-convolutional layers. One possible reason for this result is that the amount of parameters on the Baseline model was already enough for training the dataset. One of the solutions for this problem is to increase the number of data such as conducting data augmentation of training dataset. This would help the model to learn the variety of the dataset, possibly resulting in higher accuracy.

Table C1. IoU comparison of U-Net-like model between with and without Up-convolutions layer

	T1 IoU [%]	T2 IoU [%]
Evaluation (Up-Conv)	**55.73**	**62.50**
Evaluation (Baseline)	**51.91**	**63.07**

[Reference]

1. Moshkov N, Mathe B, Kertesz-Farkas A, Hollandi R, Horvath P. Test-time augmentation for deep learning-based cell segmentation on microscopy images. Sci Rep. 2020 Mar 19;10(1):5068. doi: 10.1038/s41598-020-61808-3. Erratum in: Sci Rep. 2021 Feb 2;11(1):3327. PMID: 32193485; PMCID: PMC7081314.
2. Shanmugam, D., Blalock, D., Balakrishnan, G., & Guttag, J. (2021). Better aggregation in test-time augmentation. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1214-1223).
3. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. Springer International Publishing, 2015.
4. Chaitanya, Krishna, et al. "Semi-supervised and task-driven data augmentation." Information Processing in Medical Imaging: 26th International Conference, IPMI 2019, Hong Kong, China, June 2–7, 2019, Proceedings 26. Springer International Publishing, 2019.
5. Nalepa, Jakub, Michal Marcinkiewicz, and Michal Kawulok. "Data augmentation for brain-tumor segmentation: a review." Frontiers in computational neuroscience 13 (2019): 83.