PA026 Artificial Intelligence Project

Classification and Segmentation of Brain MRI

Marián Pukančík

506485

Faculty of Informatics
Masaryk University

May 29, 2023

1. Introduction

Biomedical image classification and segmentation play a significant role in computer-aided diagnosis due to their influence on overall efficiency and accuracy. In recent years, deep learning models have achieved state-of-the-art performance rates in many computer vision tasks, including image classification and segmentation [2].

The aim of this project is to first classify MRIs (magnetic resonance imaging) into two classes (does/doesn't contain brain tumor). And subsequently, for those classified as containing the tumor, produce a binary segmentation label map that marks the pixels belonging to the tumor region within the MRI.

2. Existing solutions

In medical image processing, dataset acquisition and annotation is a very difficult and time-consuming process. Therefore, a lot of solutions come from various competitions centered around a specific dataset. *Brain Tumor AI Challenge (2021)*[1] was an example of such a competition, which involved both, brain tumor detection and classification of multi-parametric magnetic resonance imaging (mpMRI) scans. It involved a decade of Brain Tumor Segmentation (BraTS) challenges, where the participants performed a multilabel segmentation, effectively dividing the brain tumor into several sub-regions. In the classification part, the participants tried to predict MGMT promoter *methylation status*, an important biomarker for the treatment of brain tumors.

3. Data

In this project, I've used data from two datasets. The first, from now on also referred to as *main* dataset, was originally created for the work of *Buda et al.* [1]. They proposed a fully automatic way to quantify tumor imaging characteristics using deep learning-based segmentation and tested whether such characteristics are predictive of *lower-grade glioma* tumor *genomic subtypes*. Their whole pipeline is illustrated in Fig. 1. The entire *main* dataset is available at *Kaggle*[2]. It contains data from 110 patients from 5 different institutions. The number of data samples for one patient varies between 22-80. Together there are 3929 data samples, however, only around 1300 contain the tumor itself, making the whole dataset quite imbalanced. Samples of 101 patients contain 3 modalities: the grayscale original, post-contrast T1 weighted, and T2 Fluid Attenuated Inversion Recovery (T2-FLAIR) versions. These were used together as a 3-channel image (see in fig. 2). The rest of the samples contain only the FLAIR version, which was copied into all 3 channels. For all samples, there are ground truth binary segmentation maps available.

The second, *additional* dataset, also available at *Kaggle*[3], contains 7023 images of brain MRIs which are classified into 4 classes: *glioma, meningioma, non-tumor* and *pituitary.* I've used all *glioma* images (1621) and as many *non-tumor* ones (798) to make the entire dataset

[1]`https://www.rsna.org/education/ai-resources-and-training/ai-image-challenge/brain-tumor-ai-challenge-2021`

[2]`https://www.kaggle.com/datasets/mateuszbuda/lgg-mri-segmentation`

[3]`https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset?resource=download`

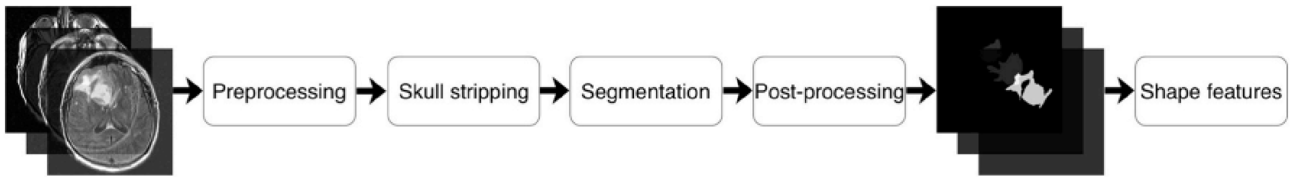


Figure 1: Pipeline used by *Buda et al.* in [1]. Image source:[1]

more balanced. All these images contain only one of the three aforementioned channels, therefore it is copied 3 times, so it could be used as a 3-channel input.

The *main* dataset is usable for both, classification and segmentation. Therefore, I've randomly picked 20% of samples (786) into the *test* dataset. It contains both, tumor (262) and non-tumor (524) samples. So, the classification model was tested on all these samples, and the segmentation one only on those containing the tumor. The rest of the non-tumor samples from the *main* dataset was used for the training of the segmentation model. The classification model was trained on all *train* data from the *main* dataset, plus all data from the *additional* dataset (a total of 5562 samples).

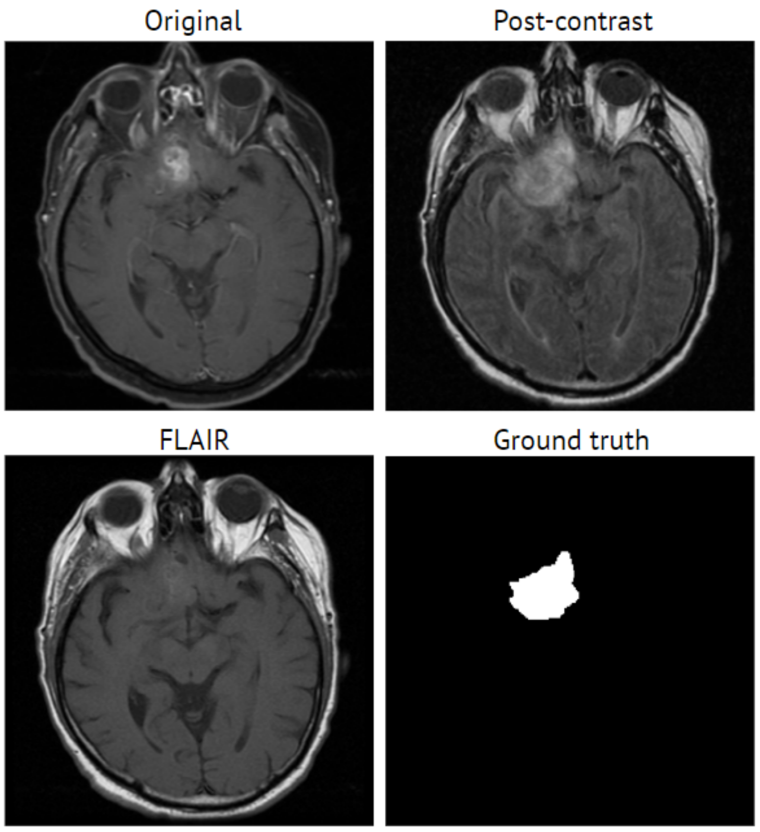


Figure 2: An example of individual channels and binary ground truth label map for one data sample from the *main* dataset.

4. Methodology

The overall task was to create a pipeline consisting of two models, one for image classification, second for segmentation. In what follows, the training and evaluation processes of both of these models are described.

4.1. Image classification

The classification model is an ensemble of three individual models performing binary classification. The final label is decided by a majority voting scheme of these three models. They were all trained via transfer learning, meaning that they all use a feature extractor with weights pre-trained on the *ImageNet*[4] dataset with several added final custom layers. The used feature extractors are the following CNNs without the final classification layer: *ResNet50* [3], *InceptionV3* [4] and *InceptionResNetV2* [5].

Training process

The training (or rather re-training) was done using the classification *train* dataset described in the previous section, from which I used 10% as *validation* dataset. All three models shared all pre-processing steps, as well as all hyper-parameter settings.

The input was resized to 256×256 pixels, augmented via random rotations/flips, random width/height shifts to diversify the data [6], and normalized to the interval $[0, 1]$.

The training was performed on GPU in *Google Colaboratory*[5] environment for 40 epochs, using the Adam optimizer, with a fixed learning rate of 0.001, batch size of 30 images, Categorical cross-entropy loss function, and Early stopping regularization. For each model, it took approximately 80 minutes to train.

Evaluation

During the training, I saved the parameters for which the model had the lowest loss value on the validation set. These parameters are then loaded and the model is evaluated on the *test* dataset. The following measures are observed [7]:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

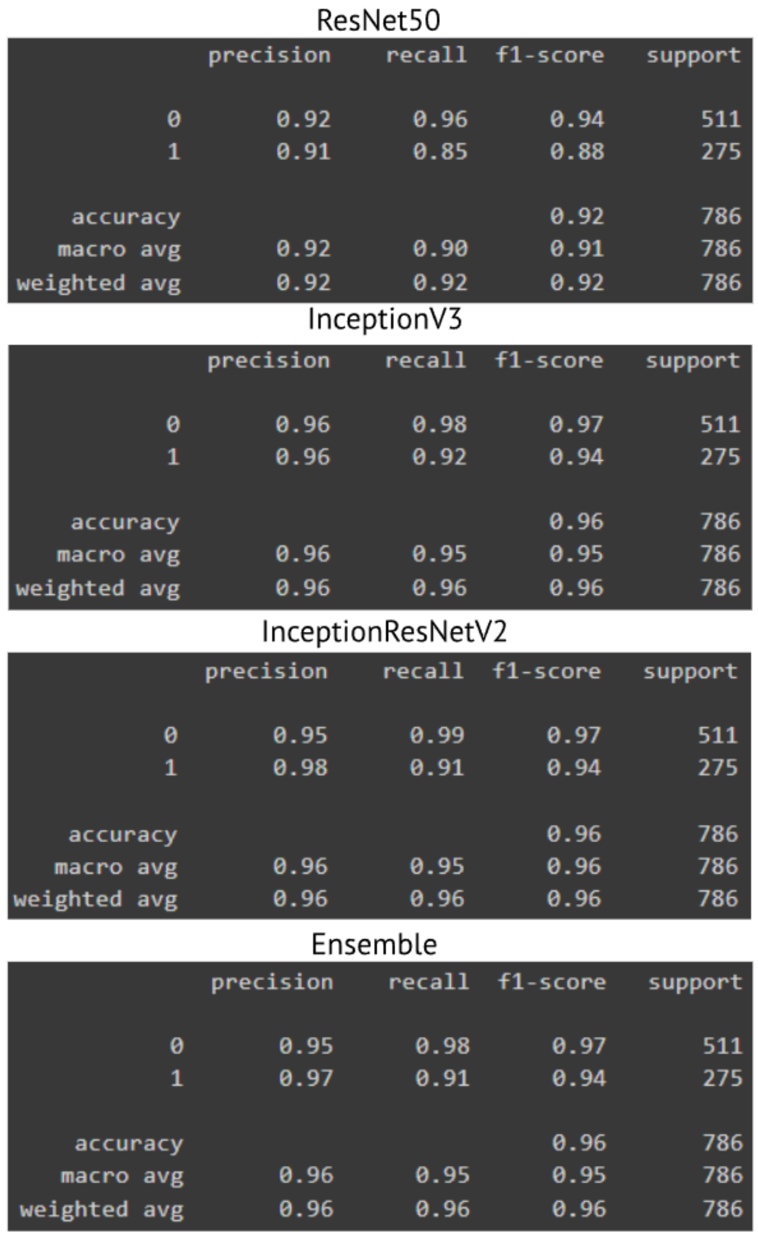
$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F_1 = \frac{(2 \times Precision \times Recall)}{(Precision + Recall)} \tag{4}$$

[4]`https://www.image-net.org/update-mar-11-2021.php`

[5]`https://colab.research.google.com/`

Fig. 3 presents evaluation results on *test* dataset for all three individual models, as well as for the ensemble model. And the fig. 4 shows a case study of ensemble model predictions. The top row is for false negatives, and the bottom one is for false positives.



ResNet50

```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       511
           1       0.91      0.85      0.88       275

    accuracy                           0.92       786
   macro avg       0.92      0.90      0.91       786
weighted avg       0.92      0.92      0.92       786
```

InceptionV3

```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       511
           1       0.96      0.92      0.94       275

    accuracy                           0.96       786
   macro avg       0.96      0.95      0.95       786
weighted avg       0.96      0.96      0.96       786
```

InceptionResNetV2

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97       511
           1       0.98      0.91      0.94       275

    accuracy                           0.96       786
   macro avg       0.96      0.95      0.96       786
weighted avg       0.96      0.96      0.96       786
```

Ensemble

```
              precision    recall  f1-score   support

           0       0.95      0.98      0.97       511
           1       0.97      0.91      0.94       275

    accuracy                           0.96       786
   macro avg       0.96      0.95      0.95       786
weighted avg       0.96      0.96      0.96       786
```

Figure 3: Evaluation results of classification models on *test* dataset.

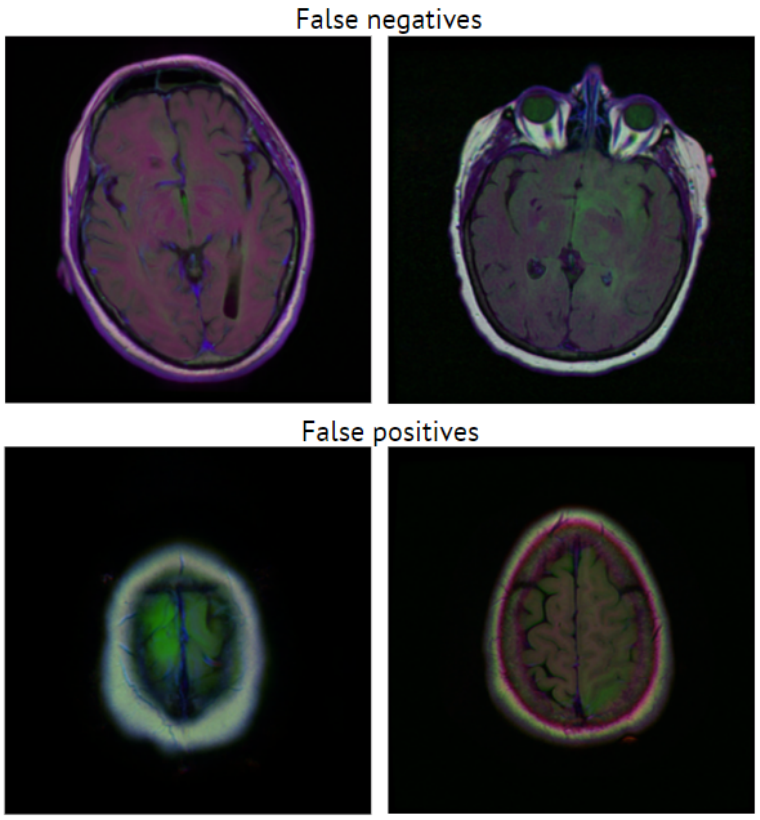


Figure 4: An example of false negatives (wrongly classified as non-tumor images) and false positives (wrongly classified as tumor images).

4.2. Image segmentation

Two segmentation models were trained. First was trained from scratch and followed the *U-Net* architecture [8] (see in fig. 5), and second was trained using transfer learning and followed the *DeepLabV3Plus* architecture with pre-trained ResNet101 as an encoder. Both models shared all pre-processing steps, as well as all hyper-parameter settings.

Training process

The input was resized to 256×256 pixels, augmented via random rotations/flips, random brightness/contrast change to diversify the data [6], and normalized to the interval $[0, 1]$.

Again, training was performed on GPU in *Google Colaboratory*[6] environment for 60 epochs, using the Adam optimizer, with a fixed learning rate of 0.0003, batch size of 30 images. For each model, it took $40 - 50$ minutes to train. A 'smooth' approximation (meaning that we replace predicted labels with probabilities) of Dice coefficient was used as a loss function. It belongs to the subclass of *region-based* losses. These losses try to maximize the overlap between ground truth and predicted segmentation output [10].

$$L_{Dice} = 1 - \frac{2 \sum_{c \in C} \sum_{p \in \Omega} y_p^c \hat{y}_p^c}{\sum_{c \in C} \sum_{p \in \Omega} y_p^c + \sum_{c \in C} \sum_{p \in \Omega} \hat{y}_p^c}, \tag{5}$$

[6]`https://colab.research.google.com/`

where C is set of classes, Ω entire spatial domain of predicted segmentation map, y_p^c ground truth binary indicator of class c of pixel p and $\hat{y}_p^c$ predicted label (probability) of class c of pixel p.

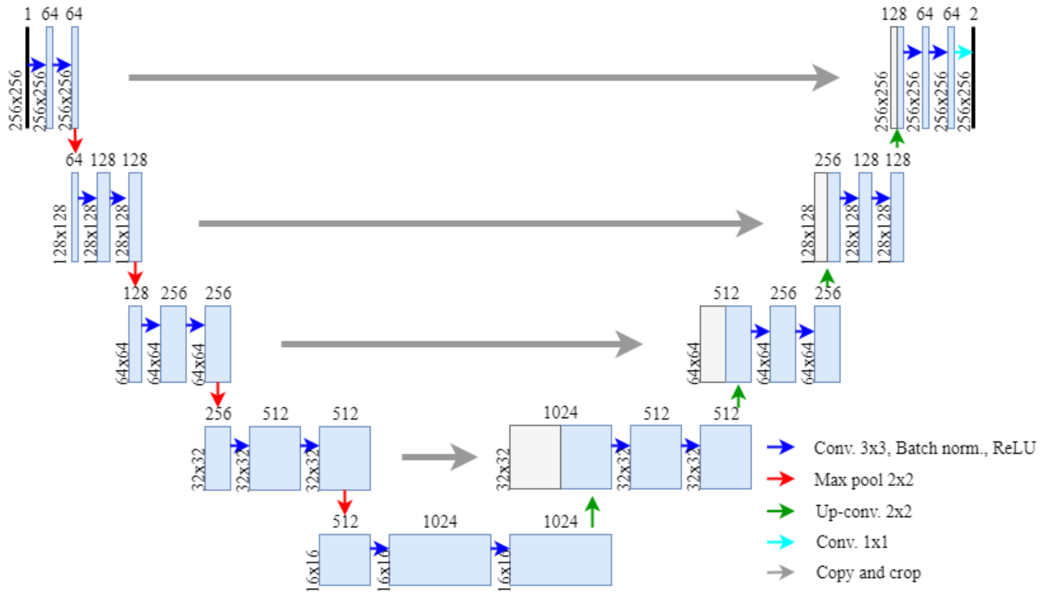


Figure 5: U-Net used in this project.

Evaluation

For both architectures, 5-fold cross-validation on *train* segmentation dataset (described in section 3) was performed. Therefore I've obtained five models for each used architecture. Again, the model parameters (weights) for which the loss on *validation* set was the lowest were used. All five models were evaluated on *validation* and *test* datasets. The quantitative results are available in tables 1 and 2 and qualitative comparison in fig. 6. Two kinds of metrics were observed: *counting* and *boundary-based*. The *counting* metrics are computed from the cardinalities of a fixed confusion matrix. I mainly monitored *Dice similarity coefficient* (DSC) and *Jaccard index*, which are members of *overlap-based* subgroup of *counting* metrics [7].

$$DSC = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{6}$$

$$Jaccard = \frac{TP}{TP + FP + FN} \tag{7}$$

From the *boundary-based* metrics I monitored *Hausdorff distance* (HD) and *Average symmetric surface distance* (ASSD). HD is the largest of all distances from a point on one boundary to the closest point on the other boundary. ASSD represents the average of all shortest boundary distances between one boundary to any point on the other boundary and vice versa, symmetrically. In both cases, the lower the value, the more similar the two objects are. The downside of these metrics is their sensitivity to outliers [7].

$$HD(A,B) = \max\{\max_{a \in A} d(a,B), \max_{b \in B} d(b,A)\}, \tag{8}$$

$$ASSD(A,B) = \frac{\sum\limits_{a\in A} d(a,B) + \sum\limits_{b\in B} d(b,A)}{|A| + |B|}, \tag{9}$$

$$d(a,B) = \min_{b\in B} d(a,b), \tag{10}$$

where A, B are sets of boundary pixels and $d(a,B)$ is the distance of the pixel a to the closest pixel from set B.

Model	DSC	Jaccard	HD	ASSD
U-Net	$0,83 \pm 0,02$	$0,75 \pm 0,02$	$13,34 \pm 1,52$	$0,96 \pm 0,24$
DeepLabV3Plus	$0,85 \pm 0,01$	$0,77 \pm 0,01$	$11,26 \pm 1,38$	$1,06 \pm 0,35$

Table 1: Evaluation results on *validation* data. It contains the mean and standard deviation for all observed metrics computed during the *cross-validation* across all 5 *validation* data folds

Model	DSC	Jaccard	HD	ASSD
U-Net	$0,81 \pm 0,01$	$0,72 \pm 0,01$	$14,42 \pm 2,49$	$1,69 \pm 0,69$
DeepLabV3Plus	$0,83 \pm 0,004$	$0,75 \pm 0,004$	$11,51 \pm 0,55$	$1,04 \pm 0,26$

Table 2: Evaluation results on *test* data. It contains the mean and standard deviation for all observed metrics computed across all five models.

5. Implementation details and startup instructions

The project directory containing all implemented scripts, input data, and obtained results can be downloaded from the project vault. All scrips were implemented using Python programming language. Specifically, the classification part was done via *TensorFlow*[7] and the segmentation via the *PyTorch*[8] framework.

The project directory contains the whole implementation. *tumorClassification.ipynb* and *tumorSegmentation.ipynb* control the data preparation, training process, and evaluation of both classification and segmentation parts respectively. *data* directory contains both used datasets, as well as *.csv* files with file paths with individual dataset splits. *savedModels* should contain saved parameters for all trained models, however, due to their memory requirements, all these models are available in my GitLab repository[9]. In *savedResults*, there are *.csv* files with validation and test results for all trained segmentation models. *scripts* contains several *.py* scripts with the implementation of evaluation/visualization functions, dataset classes, etc.

As already mentioned, this project was developed in a Google Colaboratory environment, so the easiest way to run it, is just by uploading the project directory to your Google Drive and performing the training/evaluation using the *.ipynb* files in this environment. Within the *.ipynb* files, you might need to adjust the $PROJECT\_PATH$ variable.

You should also be able to run it on your local machine by just setting the $PROJECT\_PATH$ variable to $'./'$. However, you have to have Python (3.7+) and libraries such as Pandas, Keras,

[7] `https://www.tensorflow.org/`

[8] `https://pytorch.org/`

[9] `https://gitlab.com/506485/pa026`

Numpy, Sklearn, Pytorch, etc. already installed. Also, the project wasn't fully tested in this way due to the long training times.

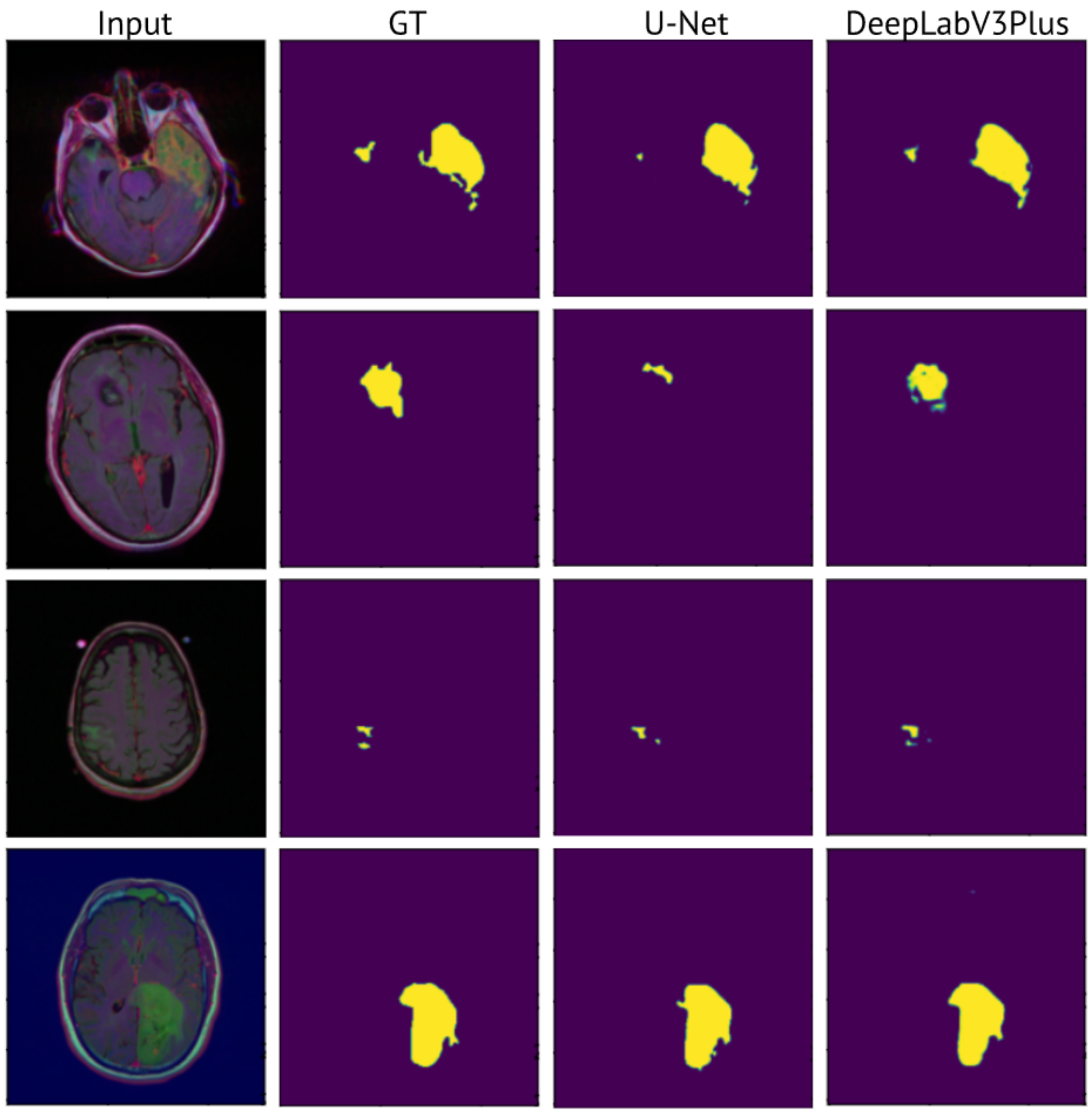


Figure 6: An example segmentation outputs. From left to right: Multi-modal input image, binary ground truth segmentation map, U-Net prediction, DeepLabV3Plus prediction.

Bibliography

[1] BUDA, Mateusz; SAHA, Ashirbani; MAZUROWSKI, Maciej A. Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm. Computers in biology and medicine, 2019, 109: 218-225.

[2] HESAMIAN, Mohammad Hesam, et al. Deep learning techniques for medical image segmentation: achievements and challenges. Journal of digital imaging, 2019, 32: 582-596

[3] HE, Kaiming, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.

[4] SZEGEDY, Christian, et al. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 2818-2826.

[5] SZEGEDY, Christian, et al. Inception-v4, inception-resnet, and the impact of residual connections on learning. In: Proceedings of the AAAI conference on artificial intelligence. 2017.

[6] TIAN, Yingjie; ZHANG, Yuqi. A comprehensive survey on regularization strategies in machine learning. Information Fusion, 2022, 80: 146-166.

[7] MAIER-HEIN, Lena, et al. Metrics reloaded: Pitfalls and recommendations for image analysis validation. arXiv. org, 2022, 2206.01653.

[8] RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-net: Convolutional networks for biomedical image segmentation. In: 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. p. 234-241.

[9] CHEN, Liang-Chieh, et al. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). 2018. p. 801-818.

[10] MA, Jun, et al. Loss odyssey in medical image segmentation. Medical Image Analysis, 2021, 71: 102035.