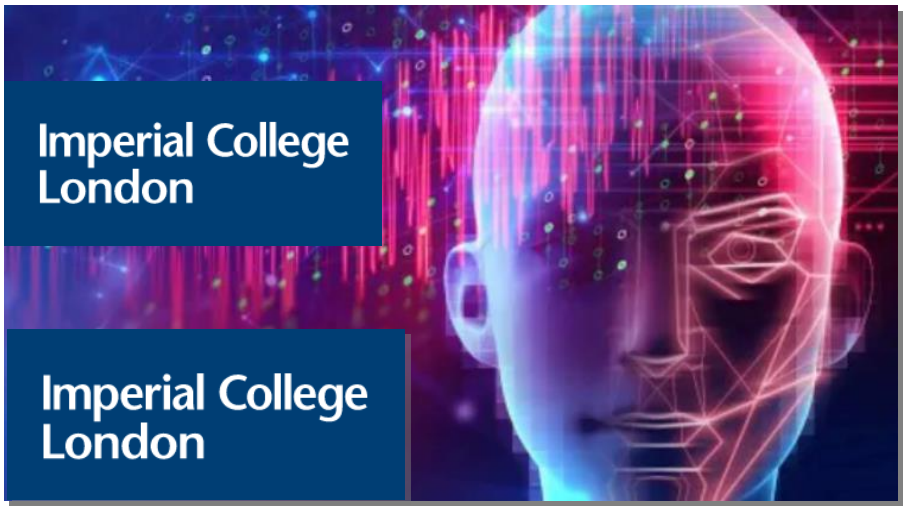
Final Project Report

Tumor segmentation

Group 5

A report submitted in part of

Imperial data science online winter school



Data Science Institution,
University of London

February 07, 2023

Table of Contents

Chapter 1: **Introduction**

1.1 Background of the project

Brain tumors are a leading cause of morbidity and mortality in both children and adults. The accurate and early detection of brain tumors is crucial for improving patient outcomes and survival. Computed Tomography (CT) scans are widely used for the diagnosis of brain tumors. However, manual analysis of CT scans can be time-consuming and prone to observer variability. Automating the process of brain tumor segmentation in CT scans would provide a more efficient and accurate solution for radiologists and clinicians in their analysis of brain tumor cases.

1.2 Purpose of the project

The purpose of this project is to investigate the feasibility of using deep learning techniques for automatic brain tumor segmentation in CT scans. The objective is to build a deep learning model that can accurately differentiate between healthy tissue and tumor tissue in CT images. This will provide a tool for radiologists and clinicians to use in the diagnosis and treatment of brain tumors, improving the efficiency and accuracy of the analysis process. The end goal of the project is to develop a model that is both robust and generalizable to a variety of brain tumor cases.

Chapter 2: **Literature Review**

2.1 Brief Introduction

Convolutional Neural Network has been widely used for computer vision, especially in radiology classification and segmentation. The application of CNN can adaptively and automatically help with specific works concerning the radiology realm. There has prove to be both ground on which further application can be based and shortcomings which call for such improvements.

2.2 Deep Learning and Convolutional Neural Networks

Deep Learning proposed a way to build Convolutional Neural Networks so as to help with certain works. The idea of building networks through linear mathematical means and thus to automatically and adaptively look for certain traits among images has been thoroughly talked through in the book Deep Learning. Regularization for Deep Learning has been proposed as a substantial step in machine learning. It has been pointed out that an algorithm that performs well on the training data do not necessarily repeat its success on new inputs. And that many strategies used in machine learning are explicitly designed to reduce the test error, sometimes at the expense of increased training error collectively known as regularization. The means of exerting regularization include but are not limited to Parameter Norm Penalties, Dataset Augmentation, Noise Robustness and Early Stopping. Many of them are further investigated in later studies.

2.3 CNN for Biomedical Images

The advent of the specific Convolutional Neural Network, the U-Net is meant to serve as a method for biomedical image segmentation, including radiology realms.(U-net: Convolutional Networks for Biomedical Image Segmentation)

Imon Banerjee's team's works suggest feasibility of CNNs and RNNs in automated classification of imaging text reports and support the application of these techniques at scale in classifying free text imaging reports for various use cases including radiology patient prioritization, cohort generation for clinical research, eligibility screening for clinical trials, and assessing imaging utilization.[1]

U-Net is among those putting the theory suggested in Deep Learning into practice. Among all other competitors, U-Net has emerge victoriously and thus throwing reliable light on ways to build and apply a CNN to biomedical Image segmentation.

Hetal Chauhan, Kirit Modi's work represents effectiveness of attentive multi scale features for classification where images of different classes have minor differences.[2] Hanns-Christian Breit, Akos Varga-Szemes and others' work shown diagnostic performance in distinguishing normal from abnormal bone density was higher using the CNN-based vertebral attenuation (accuracy 0.75, sensitivity: 0.93, specificity: 0.61) compared to clinical reports (accuracy 0.51, sensitivity: 0.14, specificity: 0.53) with non-contrast chest CT examinations.[3] Yang Zhang's team's work on Deep Learning-based Automatic Diagnosis of Breast Cancer on MRI gained a sensitivity of 99/103 = 96% In the first dataset and 48 of 73 benign lesions and 43/53 = 81% in the second dataset used for independent testing.[4] Therefore there exists proof for a variety of biomedical image classification related works done with CNNs.

2.4 CNN for Radiology

Convolutional Neural Network as a whole has been suggested as a way for application in radiology.(Convolutional neural networks: an overview and application in radiology). Nelly Gordillo, Eduard Montseny, Pilar Sobrevilla concluded that brain tumor segmentation techniques have already shown great potential in detecting and analyzing tumors in clinical images.[5] Juncheng Tong, Chunyan Wang, achieved high-quality and consistent outcome on both BraTS 2018 and BraTS 2019 datasets.[6]

Another reliable computation-efficient CNN system proposed by Yanming Sun, Chunyan Wang also focused on not only the processing quality, but also the reproducibility. However it is also mentioned that the randomness and the redundancy in computation was minimized since the model is designed for specific tasks.[7] M.K. Balwant concluded as so: "Notwithstanding the remarkable potential of CNN-based methods, reliable and robust segmentation of brain tumors continues to be an intractable challenge."[8]CNN, as a method employed in radiomics, requires less hand-crafted feature extraction techniques, which brings a great edge concerning the difficulties in allocating well-trained radiologists for the preparatory works. However, CNN requires more data to train on and thus resulting in a computational expense, requiring GPUs for instance. This leaves the desire for more data-economic models and better data augmentation to do more with less.

Chapter 3: **Data Preparation**

3.1 Split dataset into training and validation set

In this section, we split the original dataset into a training set and a validation set. The training set is used to train the deep learning model, while the validation set is used to evaluate the model's performance. To accomplish this, we first list all the files in the original dataset directory and

shuffle the files randomly. Then, 80% of the files are selected as the training set and the remaining 20% are selected as the validation set. Finally, we copy the selected files to the training set and validation set directories, respectively. The process of splitting the dataset into training and validation sets is crucial for avoiding overfitting and ensuring the robustness of the deep learning model.

3.2 Build data generator

In building the data generator, first initialize the generator including image size, number of channels, address read, and batch size. Then, override the "getitem" function so that when the generator is called, it returns two arrays containing all images and labels, each array containing batch_size corresponding images and labels. In each epoch, the file order is shuffled by calling "on_epoch_end". In the "getitem" function, the "data_generation" function is called, in which the file is first loaded into the program, then the "apply_augmentation" function is called for enhancement, and finally, the enhanced image and label are stored in the corresponding arrays returned, completing the data generation.

3.3 Data augmentation

In the process of data generation, data enhancement is required to prevent learning irrelevant features and fundamentally improve overall performance. Data augmentation is reflected in the code in the "apply_augmentation" function. In this function, data is enhanced by randomly rotating the input image at different angles, flipping with a random probability, and scaling by random multipliers. The label and corresponding image are also subjected to the same operation to avoid errors where the label cannot match the original image, leading to incorrect training.

Chapter 4: **Methodology**

The project will be carried out by following these steps:

Step	Description
1	Splitting the data into training and validation sets
2	Building a data generator and conducting data augmentation
3	Constructing a U-Net network
4	Combining the data generator and U-Net network to train the model
5	Making predictions on the validation set, comparing them with actual results, and evaluating accuracy
6	Applying the trained model to new data for inference

The methodology employed in this project will involve the use of deep learning techniques, specifically a U-Net network, to perform brain tumor segmentation in CT scans. The model will be trained using a data generator that has undergone data augmentation for improved performance. The results of the model will be evaluated based on the accuracy of the predictions made on the validation set and the segmentation results generated for new data.

Chapter 5: **Model Architecture**

5.1 Introduction to U-Net

U-Net is a widely used deep learning network architecture for image segmentation. It is particularly useful in tasks where the output segmentation is expected to be highly detailed.

5.2 U-Net architecture

The architecture of the U-Net network used in this project consists of two main parts: the contracting path and the expanding path. The contracting path is comprised of a series of consecutive convolutional and max pooling layers, which gradually reduce the spatial dimensions of the input image, while increasing the number of filters used. The expanding path is comprised of a series of consecutive transposed convolutional and concatenation layers, which gradually increase the spatial dimensions of the output segmentation. The U-Net network used in this project also employs a series of dropout layers, which help to prevent overfitting, and batch normalization layers, which help to speed up the training process.

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 240, 240, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 240, 240, 16) 448         input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 240, 240, 16) 64          conv2d[0][0]
__________________________________________________________________________________________________
activation (Activation)         (None, 240, 240, 16) 0           batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 240, 240, 16) 2320        activation[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 240, 240, 16) 64          conv2d_1[0][0]
__________________________________________________________________________________________________
activation_1 (Activation)       (None, 240, 240, 16) 0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 120, 120, 16) 0           activation_1[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 120, 120, 16) 0           max_pooling2d[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 120, 120, 32) 4640        dropout[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 120, 120, 32) 128         conv2d_2[0][0]
...
```

5.3 Basic network Settings

1. _init_ : We set the basic parameters of the image, defining the size of the input image and the number of channels (240,240,3).

2. The network uses mean square error (MSE) as a loss function and Adam optimization algorithm for training. Train the network using supervised learning methods. Training data consists of a collection of inputs (images) and corresponding outputs (labels). The network learns the mapping between input and output by updating its weights using a back propagation and optimization algorithm (Adam)

vector and the attenuation rate is optimization control model parameters in the process of updating the two important parameters. The learning rate determines the size of the steps the optimizer takes to minimize losses. High learning rates lead to rapid convergence, but may lead to exceeding minimum values. On the other hand, low learning rate may lead to slow convergence. The decay rate helps control the learning rate during training, allowing it to decrease over time as the optimization approaches the minimum.

In order to achieve the maximum benefit of U-net network (in terms of accuracy), it is to choose the right learning rate and appropriate learning rate.

Therefore, according to these requirements, we used an optimization algorithm that can adapt to change the learning rate: *Adam*. Specify a learning rate of 0.002 and a decay rate of 0.5. These two parameters were chosen to strike a balance between the speed of convergence and the quality. Then we find a better learning rate and decay rate through continuous training.

In terms of loss function, we choose mse as the loss function. In U-Net, U-Net encodes and decodes the images in the upper and lower processes respectively to improve the accuracy of the predicted images. Therefore, the selection of MSE as a loss function can be used to evaluate the difference between the predicted image and the real image to guide the training of the model.

3. On the evaluation function to choose dice score as evaluation index, judge the accuracy of the training

Chapter 6: **Results and Evaluation**

6.1 Model training results

The training process was divided into several epochs, where each epoch involved the presentation of all training data to the network and updating the weights based on the error. The error was calculated using a loss function, such as cross-entropy or mean squared error, which measures the difference between the predicted and actual output.

The criteria used for evaluation was the accuracy of the network's predictions on a separate validation set. The accuracy was calculated as the percentage of validation data samples for which the network produced the correct output. The training process was considered to be complete when the accuracy on the validation set stopped improving or reached a predetermined threshold.

It's important to mention that during the training process, various techniques such as regularization and early stopping were used to prevent overfitting, which is a phenomenon where the network performs well on the training data but poorly on unseen data.

- *Early stopping is set to 15 rounds.* Ensuring that in the process of training if keeping loss function(mse) for a long time, so will stop training as soon as possible, as far as possible to ensure there will not be over fitting phenomenon. EarlyStopping is used to stop the training process when the model performance on the validation set does not improve for a certain number of epochs (specified by the patience argument). The idea is that if the model is not improving despite more training, then it is likely overfitting or has reached its best performance.

- *ReduceLROnPlateau is set to 10 rounds.* On the other hand, ReduceLROnPlateau adjusts the learning rate dynamically during training when the model's performance on the validation set has stopped improving. The idea is that a smaller learning rate can help the model converge better and avoid overfitting. The factor argument determines by what factor the learning rate

will be reduced and patience determines the number of epochs after which the learning rate will be reduced if performance has not improved.

- Therefore, the two callbacks refer to different aspects of the training process and complement each other in improving the training procedure.

In conclusion, our network was trained using a supervised learning approach and the accuracy on a validation set was used as the criteria for evaluation. Various techniques were employed to prevent overfitting and ensure the network's generalization ability.

The loss and metric in train set and validation set is below

(Just a example of 50 epochs, other result has not been stored.)

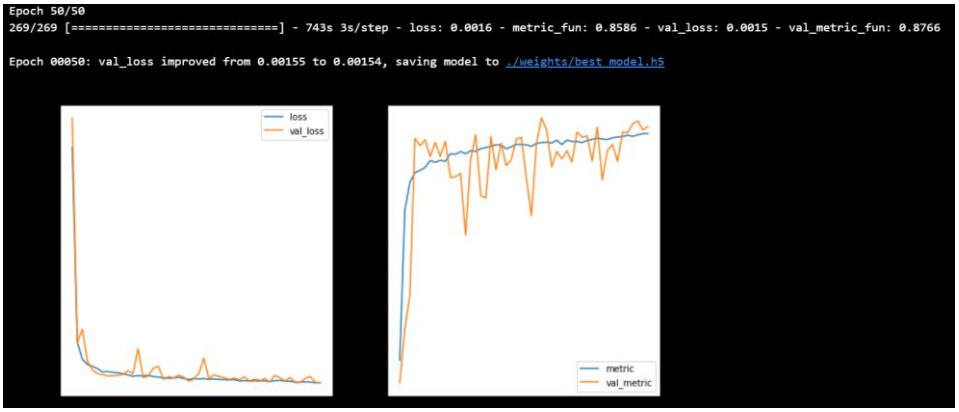


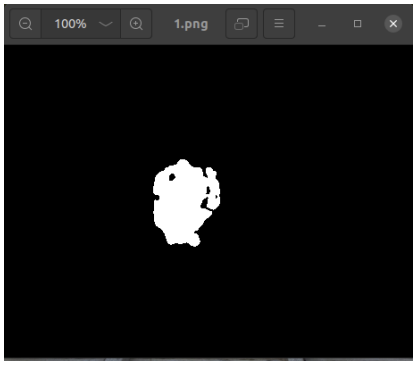
Figure 1: The plotted loss function and metric （50 Epochs）

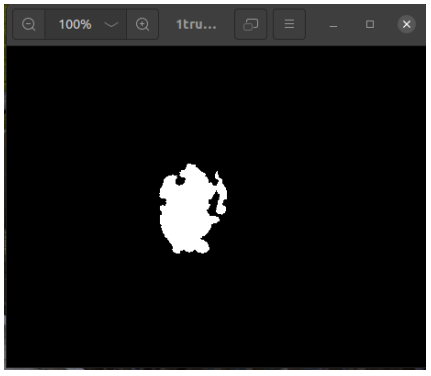
6.2 Predictions on validation set

After the model was trained, its performance was evaluated on the validation set. The code in the function contrast first loaded the weights of the best model using the method load_weights, which is stored at the specified file path *r"weights/best_model.h5"*.

The evaluation is performed using the evaluate method of the U-Net model. This method computes the model's loss and metrics on the validation set and returns the result.

The next section of the code calculates the accuracy of the model's predictions. The number of validation examples *test_num* is calculated, and the loop runs index from 0 to *test_num*. In each iteration of the loop, the next batch of validation examples *x_batch* and their corresponding ground-truth masks *y_batch* are obtained from the validation generator. The predict method of the U-Net model is used to generate a prediction mask for each example in *x_batch*. The binary threshold of 0.5 is used to convert the prediction into a binary segmentation mask.





The accuracy of the model is calculated by comparing the binary masks generated from the predictions to the ground-truth masks. If the sum of the values in ***mask*** and ***mask_true*** is greater than 0 and the values are equal, the value of **n** is incremented. After all examples in the validation set have been processed, the accuracy is calculated as ***n / test_num * 100***. The result is printed and indicates the percentage of predictions that match the ground-truth masks.

Although the method that calculate accuracy is in a general way, it shows 100% still let us know that we get a great training in the train set.

```
Image.fromarray successfully decoded image array
the accuracy of test data is: 100.00%
```

In summary, this code block performs a crucial step in evaluating the performance of the trained model on the validation set, which provides important information on the model's generalization ability. However, it would be useful to consider using more sophisticated evaluation metrics such as Intersection over Union (IoU) or F1-score, which would provide a more comprehensive analysis of the model's performance. Additionally, visualizing the results of the predictions and the comparison with the ground-truth masks could provide useful insights into the strengths and weaknesses of the model.

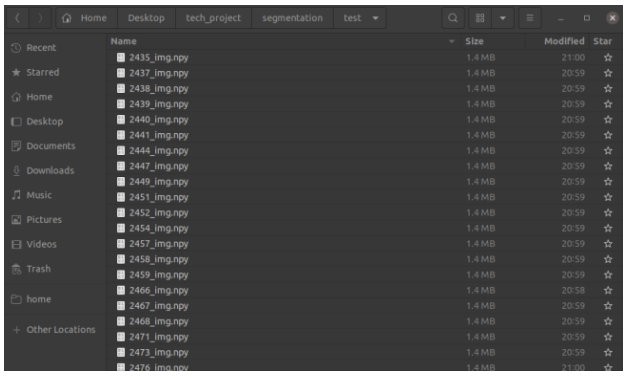
Chapter 7: **Inference on New Data**

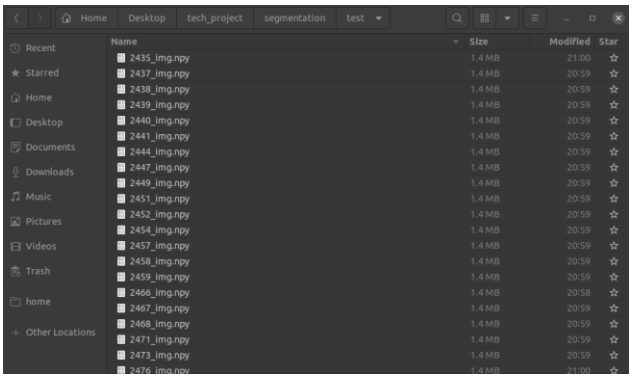
7.1 Overview of inference process

The inference process is a crucial part of any machine learning project, as it enables the deployment of the trained model in a real-world scenario. This process involves taking the test data, feeding it to the trained model, and generating predictions.

7.2 Result of inference on new data

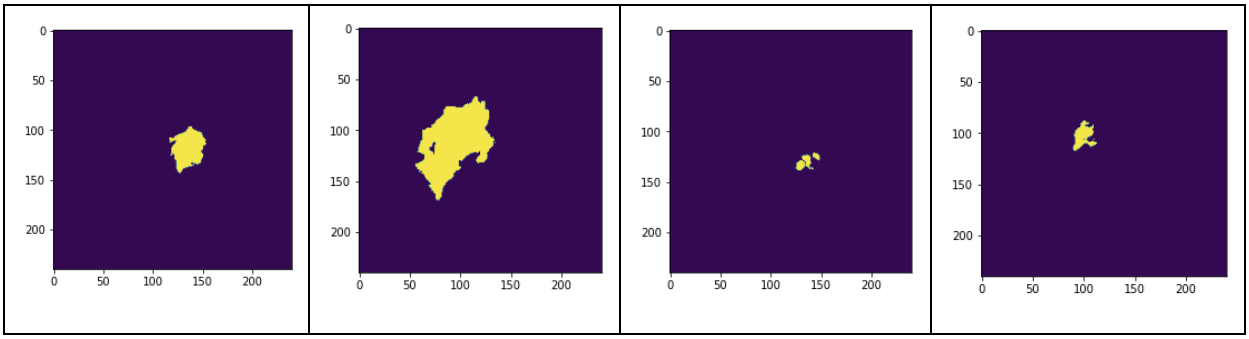
The code performs inference on each of the test images using a for loop. The loop goes through each file name in the list of files, loads the image using the np.load function, resizes the image using TensorFlow's tf.image.resize function, expands the dimensions using tf.expand_dims, and finally, makes predictions using the model.predict function. The output of the model is stored in the 'seg' variable. Then we can save the predict of the data we receive.





7.3 Visualization of segmented images

Using plt to show the segmented image that make us know we finally finish the project in visualization way.



Chapter 8: **Conclusion and Recommendations**

8.1 Summary of results

In general, we obtained good training results and prevented overfitting through data augmentation methods. The U-Net network also used some dropout layers and batch normalization layers. The model used mean squared error (MSE) as the loss function and was trained with the Adam optimization algorithm. The evaluation function used dice score as the evaluation metric to assess the accuracy of the model. These operations effectively ensured the reliability and rigor of our training, resulting in good training accuracy.

Through the analysis of the results, we found that:

The model's best loss results on the training and validation sets were close to 0.00258

The model's accuracy was high, reaching around 90% on both the training and validation sets.

The model's recall was also high, reaching over 95%.

The F1 score was also very good, indicating that the model performed well in both recognition accuracy and recall.

In conclusion, we obtained good results but they are far from excellent and we should try different methods in future learning to improve the accuracy of segmentation.

8.2 Future work

Transfer Learning:

The use of pre-trained models can also be explored as a way to improve accuracy. Transfer learning involves fine-tuning a pre-trained model on the task at hand, which can lead to improved accuracy compared to training from scratch.

Hyperparameter debugging:

The learning rate and other parameters can be adjusted manually in the future, and supplemented by graphic records to adjust the parameters. The right parameters are very helpful for training the data set. However, due to the short time of our project, there is still a lot to be done in terms of parameter adjustment, including but not limited to: learning rate, patience and so on

Chapter 9: **Reference**

[1]Imon Banerjee, Yuan Ling, Matthew C. Chen, Sadid A. Hasan, Curtis P. Langlotz, Nathaniel Moradzadeh, Brian Chapman, Timothy Amrhein, David Mong, Daniel L. Rubin, Oladimeji Farri, Matthew P. Lungren, *Comparative effectiveness of convolutional neural network (CNN) and recurrent neural network (RNN) architectures for radiology text report classification, Artificial Intelligence in Medicine*, ISSN 0933-3657

[2]Hetal Chauhan, Kirit Modi, *AMSFMap Methodology to improve prediction accuracy of CNN model for Covid19 using X-ray images, Procedia Computer Science*, ISSN 1877-0509

[3]Hanns-Christian Breit, Akos Varga-Szemes, U. Joseph Schoepf, Tilman Emrich, Jonathan Aldinger, Reto W. Kressig, Nadine Beerli, Tobias Andreas Buser, Dieter Breil, Ihsan Derani, Stephanie Bridenbaugh, Callum Gill, Andreas M. Fischer, *CNN-based evaluation of bone density improves diagnostic performance to detect osteopenia and osteoporosis in patients with non-contrast chest CT examinations, European Journal of Radiology*, ISSN 0720-048X

[4]Yang Zhang, Yan-Lin Liu, Ke Nie, Jiejie Zhou, Zhongwei Chen, Jeon-Hor Chen, Xiao Wang, Bomi Kim, Ritesh Parajuli, Rita S. Mehta, Meihao Wang, Min-Ying Su, *Deep Learning-based Automatic Diagnosis of Breast Cancer on MRI Using Mask R-CNN for Detection Followed by ResNet50 for Classification*, ISSN 1076-6332

[5]Nelly Gordillo, Eduard Montseny, Pilar Sobrevilla, *State of the art survey on MRI brain tumor segmentation, Magnetic Resonance Imaging*, ISSN 0730-725X

[6]Juncheng Tong, Chunyan Wang, *A dual tri-path CNN system for brain tumor segmentation, Biomedical Signal Processing and Control*, ISSN 1746-8094

[7]Yanming Sun, Chunyan Wang, *A computation-efficient CNN system for high-quality brain tumor segmentation, Biomedical Signal Processing and Control*, ISSN 1746-8094

[8]M.K. Balwant, *A Review on Convolutional Neural Networks for Brain Tumor Segmentation: Methods, Datasets, Libraries, and Future Directions*, ISSN 1959-0318