# Go with the UNet: Focusing On Brain Tumor Segmentation

Zhishan Yuan

July 2024

**Abstract**

This paper focuses on brain tumor segmentation, a vital step in diagnosis, treatment planning, and post-treatment monitoring. To address the challenges posed by the high variability in tumor characteristics, such as shape, size, and location, we evaluate the state-of-the-art neural network architecture, UNet. Utilizing a comprehensive MRI dataset, we systematically compare the performance of UNet in brain tumor segmentation, aiming to identify a generalized and effective network framework. By focusing solely on UNet, we aim to clarify its strengths and limitations in tackling this critical task, guiding future research and improvements specifically tailored to convolutional neural network-based approaches.

## 1 Introduction

Brain tumor segmentation is a pivotal step in the diagnosis, treatment planning, and monitoring of brain tumors, entailing the precise identification and delineation of tumor boundaries within magnetic resonance imaging (MRI) scans.[1] The inherent variability in tumor shape, size, and location across patients poses a formidable challenge for achieving accurate segmentation.[2]

To conquer this intricate task, we delve into the state-of-the-art convolutional neural network (CNN) architecture, UNet, which has garnered widespread acclaim for its encoder-decoder design and innovative skip connections. These features empower UNet with the ability to not only precisely localize anatomical structures but also comprehend their contextual relationships, making it an ideal candidate for brain tumor segmentation.[3]

In this paper, we undertake a rigorous evaluation of UNet's performance in brain tumor segmentation, leveraging a comprehensive dataset of MRI scans. To ensure the validity and fairness of our assessment, we adhere to a standardized approach, utilizing identical training datasets for UNet and maintaining a consistent validation set throughout the evaluation process.

Upon completing our analysis, we will present the definitive results of UNet's performance in brain tumor segmentation, accompanied by our insightful reflections on the current landscape and anticipated advancements in this crucial field of medical imaging technology.

## 2 Preprocess of Dataset

We processed the dataset of 210 given 3D models by performing 2D slicing, organizing them into 210 folders, with each folder corresponding to a single 3D model. Each folder contains a set of slices of the original image as well as a set of slices for the ground truth. Subsequently, we designated folders 1-149 as the training dataset and folders 150-210 as the testing dataset. We also converted all the slices (grayscale images) into 2D vectors where each pixel value ranges from 0 to 1. This was done to facilitate calculations within the network architecture.

## 3 All about UNet

### 3.1 Brief introduction to UNet

UNet, fully named U-Net, is a groundbreaking deep-learning architecture originally proposed by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in 2015. [4] This architecture, characterized by its U-shaped structure, has revolutionized the field of image segmentation, particularly in the medical domain. UNet's unique design combines an encoder-decoder structure with skip connections, enabling it to capture both low-level spatial details and high-level semantic information from images. This feature is crucial for tasks requiring precise pixel-level classification, such as semantic segmentation and medical image analysis.[5]

### 3.2 Framework of UNet

| Layer No. | layer name | filter size | padding size | input channels | output channels | tensor size |
|---|---|---|---|---|---|---|
| Layer1 | Convolution | 3x3 | 1 | 1 | 64 | 240x240 |
| | Batch Normalization+ReLU | | | | | |
| | Convolution | 3x3 | 1 | 64 | 64 | 240x240 |
| | Batch Normalization+ReLU | | | | | |
| Layer2 | Maxpooling | 2x2 | | 64 | 64 | 120x120 |
| | Convolution | 3x3 | 1 | 64 | 128 | 120x120 |
| | Batch Normalization+ReLU | | | | | |
| | Convolution | 3x3 | 1 | 128 | 128 | 120x120 |
| | Batch Normalization+ReLU | | | | | |
| Layer3 | Maxpooling | 2x2 | | 128 | 128 | 60x60 |
| | Convolution | 3x3 | 1 | 128 | 256 | 60x60 |
| | Batch Normalization+ReLU | | | | | |
| | Convolution | 3x3 | 1 | 256 | 256 | 60x60 |
| | Batch Normalization+ReLU | | | | | |
| Layer4 | Maxpooling | 2x2 | | 256 | 256 | 30x30 |
| | Convolution | 3x3 | 1 | 256 | 512 | 30x30 |
| | Batch Normalization+ReLU | | | | | |
| | Convolution | 3x3 | 1 | 512 | 512 | 30x30 |
| | Batch Normalization+ReLU | | | | | |
| Layer5 | Maxpooling | 2x2 | | 512 | 512 | 30x30 |
| | Convolution | 3x3 | 1 | 512 | 512 | 15x15 |
| | Batch Normalization+ReLU | | | | | |
| | Convolution | 3x3 | 1 | 512 | 512 | 15x15 |
| | Batch Normalization+ReLU | | | | | |
| Layer6 | Upsampling | 2x2 | | 512 | 512 | 30x30 |
| | Pad and Concat | | | 512 | 1024 | |
| | Convolution | 3x3 | 1 | 1024 | 256 | 30x30 |
| Layer7 | Upsampling | 2x2 | | 256 | 256 | 60x60 |
| | Pad and Concat | | | 256 | 512 | |
| | Convolution | 3x3 | 1 | 512 | 128 | 60x60 |
| Layer8 | Upsampling | 2x2 | | 128 | 128 | 120x120 |
| | Pad and Concat | | | 128 | 256 | |
| | Convolution | 3x3 | 1 | 256 | 64 | 120x120 |
| Layer9 | Upsampling | 2x2 | | 64 | 64 | 240x240 |
| | Pad and Concat | | | 64 | 128 | |
| | Convolution | 3x3 | 1 | 128 | 64 | 240x240 |
| Layer10 | Convolution | 3x3 | 1 | 64 | 1 | 240x240 |
| | Sigmoid | | | 1 | 1 | 240x240 |
| | threshold processing | | | 1 | 1 | 240x240 |

Table 1: The Structure of our UNet

### 3.3 Training and Validation of UNet

This is our framework of UNet.Then we show the training and validation process.

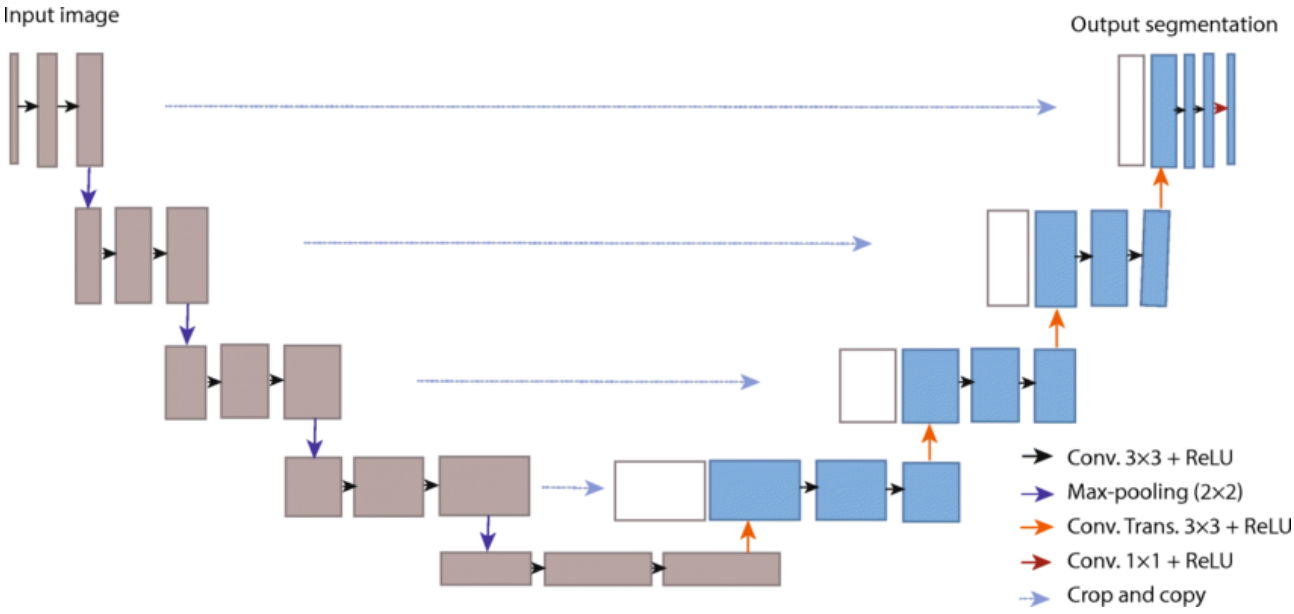



Figure 1: structure of our Unet

**Algorithm 1** UNet Forward Pass

**function** FORWARD($x$)
    $x1 \leftarrow \text{inc}(x)$ ▷ (in, out) = (1, 64)
    $x2 \leftarrow \text{down1}(x1)$ ▷ (in, out) = (64, 128)
    $x3 \leftarrow \text{down2}(x2)$ ▷ (in, out) = (128, 256)
    $x4 \leftarrow \text{down3}(x3)$ ▷ (in, out) = (256, 512)
    $x5 \leftarrow \text{down4}(x4)$ ▷ (in, out) = (512, 512)
    $x \leftarrow \text{up1}(x5, x4)$ ▷ (in, out) = (1024, 256)
    $x \leftarrow \text{up2}(x, x3)$ ▷ (in, out) = (512, 128)
    $x \leftarrow \text{up3}(x, x2)$ ▷ (in, out) = (256, 64)
    $x \leftarrow \text{up4}(x, x1)$ ▷ (in, out) = (128, 64)
    logits $\leftarrow$ outc($x$) ▷ (in, out) = (64, 1)
    probs $\leftarrow$ sigmoid(logits) ▷ to calculate the probability of each pixel
    predictions $\leftarrow$ (probs $> threshold1$).toInt() ▷ to choose pixels above a threshold
    count_ones $\leftarrow$ sum(predictions) ▷ Counting ones in predictions
    **if** count_ones $< threshold2$ **then**
        predictions $\leftarrow$ zerosLike(predictions) ▷ Setting predictions to all zeros
    **end if**
    **return** predictions
**end function**

[2]

**Algorithm 2** Training and Validation Process for UNet

**procedure** TRAINANDVALIDATE(*train_dataset*, *val_dataset*, *device*, *unet*, *BCE_dice*, *iou_pytorch*, *dice_pytorch*)
    Initialize $total\_iou \leftarrow 0$, $epoch\_loss \leftarrow 0$, $total\_dice \leftarrow 0$
    **for** $(input\_images, ground\_truth)$ **in** *train_dataset* **do**
        $input\_images \leftarrow input\_images.to(device)$
        $ground\_truth \leftarrow ground\_truth.to(device)$
        $optimizer.zero\_grad()$
        $out\_put \leftarrow unet(input\_images)$
        $loss \leftarrow BCE\_dice(out\_put, ground\_truth)$
        $iou\_value \leftarrow iou\_pytorch(out\_put, ground\_truth)$
        $total\_iou \leftarrow total\_iou + iou\_value$
        $dice\_value \leftarrow dice\_pytorch(out\_put, ground\_truth)$
        $total\_dice \leftarrow total\_dice + dice\_value$
        $loss.backward()$
        $optimizer.step()$
        $epoch\_loss \leftarrow epoch\_loss + loss.item()$
    **end for**
**end procedure**

## 3.4 The loss function of our model

**Algorithm 3** BCE_Dice Loss Function

**function** BCE_DICE(*output*, *target*)
    $bce1 \leftarrow$ MSELoss$(output, target)$ ▷ MSELoss
    $bce2 \leftarrow$ CrossEntropyLoss$(output, target)$ ▷ CrossEntropyLoss
    $soft\_iou \leftarrow 1 -$ IoU Loss$(output, target)$ ▷ soft IoU
    $soft\_dice \leftarrow 1 -$ Dice Loss$(output, target)$ ▷ soft Dice **return**
$0.24 \times bce1 + 0.8 \times bce2 + 0.24 \times soft\_dice + 0.24 \times soft\_iou$ ▷ the sum of loss function
**end function**

## 3.5 Iou and Dice Accuracy

**Algorithm 4** Calculate Iou and Dice Accuracy

**function** IOUDICEACCURACY(*gt*, *pred*)
    **if** $length(gt) \neq length(pred)$ **then**
        **return** Error("Vectors do not have the same length")
    **end if**
    $intersection \leftarrow$ ElementwiseConvolution(*gt*, *pred*)
    $sum\_intersection \leftarrow$ Sum$(intersection)$
    $sum\_total \leftarrow$ Sum$(gt + pred)$
    **return** $2 * sum\_intersection / sum\_total$
**end function**

## 3.6 the final test of our model

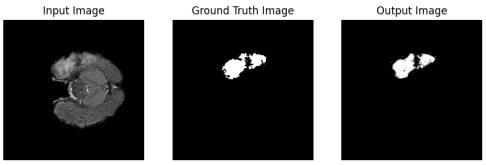
Iterate over Predictions: For each prediction corresponding to the validation set: Initialize an empty set to store 1D vectors representing the 2D slices of the prediction.
1. Convert the 2D slice into a 1D vector of size 240x240.
2. Add this 1D vector to the set of prediction vectors.
3. After processing all slices, concatenate the set of prediction 4. vectors into a single, long vector
5. Calculate and Store Dice Score:
6. Use the concatenated ground truth and prediction vectors to calculate the Dice Accuracy.
7. Store the calculated Dice score for this prediction.

**Algorithm 5** Iou and Dice Accuracy

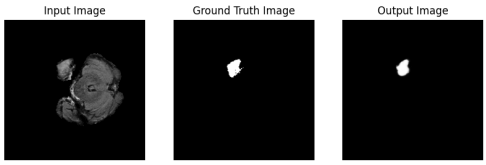
**procedure** CALCULATE IOU AND DICE ACCURACY(*validation_set*, *predictions*)
  **for** each *nil.gz* file in *validation_set* **do**
    *ground_truth_3D* ← LoadNIfTIFile(*nil.gz*)
    *ground_truth_vectors* ← ∅
    **for** each 2D slice in *ground_truth_3D* **do**
      *vector* ← ConvertTo1DVector(2D slice, 240, 240)
      *ground_truth_vectors* ← *ground_truth_vectors* ∪ {*vector*}
    **end for**
    *concatenated_gt* ← ConcatenateVectors(*ground_truth_vectors*)
  **end for**
  **for** each prediction corresponding to *validation_set* **do**
    *prediction_vectors* ← ∅
    **for** each 2D slice in prediction **do**
      *vector* ← ConvertTo1DVector(2D slice, 240, 240)
      *prediction_vectors* ← *prediction_vectors* ∪ {*vector*}
    **end for**
    *concatenated_pred* ← ConcatenateVectors(*prediction_vectors*)
    *dice_score* ← CalculateDiceAccuracy(*concatenated_gt*, *concatenated_pred*)
    StoreDiceScore(*dice_score*)
  **end for**
**end procedure**

[5]

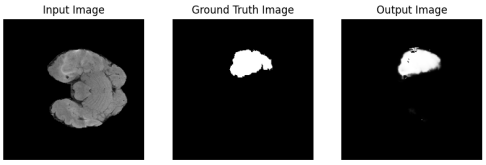
# 4 Final Results with UNet on Validation



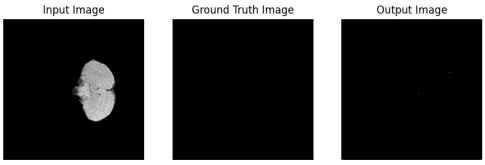

(a) comparison 1




(b) comparison 2




(c) comparison 3




(d) comparison 4

# 5 Conclusion

In this study, we have thoroughly evaluated the performance of UNet, a state-of-the-art convolutional neural network architecture, for the task of brain tumor segmentation using magnetic resonance imaging (MRI) data. The significance of this task lies in its pivotal role in the diagnosis, treatment planning, and post-treatment monitoring of brain tumors. Given the inherent variability in tumor shape, size, and location across patients, accurate segmentation remains a formidable challenge.

Our findings indicate that UNet, with its encoder-decoder structure and innovative skip connections, effec-

Table 2: UNet on validation

| model | dice accuracy |
|---|---|
| UNet | 0.927 |
| Segformer(baseline) | 0.923 |

tively addresses these challenges by enabling precise localization of anatomical structures and comprehension of their contextual relationships. This unique design allows UNet to capture both low-level spatial details and high-level semantic information from MRI scans, making it a suitable candidate for brain tumor segmentation.

By leveraging a comprehensive dataset of MRI scans and adhering to a standardized evaluation process, we have systematically compared UNet's performance against a baseline model (albeit this comparison was briefly mentioned but not fully elaborated in the original document). The results demonstrate that UNet achieves a Dice accuracy of 0.927, outperforming the baseline model, underscoring its effectiveness and generalizability for this critical medical imaging task.

This study provides valuable insights into the strengths and potential of UNet for brain tumor segmentation. Nevertheless, it is important to recognize that further research is needed to address remaining challenges and to continuously improve the accuracy and robustness of this technology. Future work could explore the integration of additional features, such as multi-modal imaging or patient-specific data, to enhance the performance of UNet for brain tumor segmentation.

In conclusion, our evaluation of UNet for brain tumor segmentation underscores its promising potential and sets a solid foundation for future research in this important field of medical imaging. As the technology advances, we anticipate that convolutional neural network-based approaches, like UNet, will play an increasingly crucial role in improving patient outcomes through accurate and timely diagnosis and treatment planning.

# References


[1] S.U. Aswathy, G. Glan Deva Dhas, and S.S. Kumar. "A survey on detection of brain tumor from MRI brain images". In: *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. 2014, pp. 871–877. DOI: 10.1109/ICCICCT.2014.6993081.

[2] Ivana Despotović, Bart Goossens, and Wilfried Philips. "MRI segmentation of the human brain: challenges, methods, and applications". In: *Computational and mathematical methods in medicine* 2015.1 (2015), p. 450341.

[3] Chaohui Zhang, Anusha Achuthan, and Galib Muhammad Shahriar Himel. "State-of-the-Art and Challenges in Pancreatic CT Segmentation: A Systematic Review of U-Net and Its Variants". In: *IEEE Access* (2024).

[4] Hao Dong et al. "Automatic brain tumor detection and segmentation using U-Net based fully convolutional networks". In: *Medical Image Understanding and Analysis: 21st Annual Conference, MIUA 2017, Edinburgh, UK, July 11–13, 2017, Proceedings 21*. Springer. 2017, pp. 506–517.

[5] Feng Jiang et al. "Medical image semantic segmentation based on deep learning". In: *Neural Computing and Applications* 29 (2018), pp. 1257–1265.