JOURNAL OF DECISION MAKING AND HEALTHCARE
Volume 2 (2025), No. 1, 83–92
https://doi.org/10.69829/jdmh-025-0201-ta05
Tulipa Opera Scholarum

2.5D ENSEMBLE LEARNING FOR INTRACRANIAL TUMOR SEGMENTATION

CHIH-FAN KUO[1] AND FENG-SHENG TSAI[2,3,*]

[1] *Department of Education, China Medical University Hospital, China Medical University, Taichung 40402, Taiwan*
[2] *Department of Biomedical Informatics, China Medical University, Taichung 40402, Taiwan*
[3] *Research Center for Interneural Computing, China Medical University Hospital, Taichung 40447, Taiwan*

Abstract. Localizing tumors in medical images for radiation therapy is a time-intensive task, potentially addressable with AI, such as deep learning. However, processing 3D medical images presents challenges, particularly regarding the significant computational resources needed. To tackle this, our research introduces a 2.5D ensemble learning framework. This approach utilizes semi-2D images as the dataset, employing an ensemble of models trained on axial, coronal, and sagittal views. Our study utilized 1500 brain magnetic resonance imaging (MRI) scans along with their corresponding tumor segmentation masks collected from 2020 The 3rd Asia Cup Brain Tumor Segmentation Challenge. We employed a modified U-Net architecture and established a 3D model as a baseline for comparison against our 2.5D ensemble learning framework. The 3D model achieved dice, precision, and recall scores of 0.5694, 0.6304, and 0.5789, respectively. In contrast, our 2.5D ensemble model demonstrated superior performance with dice, precision, and recall scores of 0.6011, 0.6527, and 0.6162, while requiring fewer computational resources.

Keywords. Brain MRI, Deep learning, 2.5D ensemble learning, Tumor segmentation, U-Net.

1. Introduction

Radiation therapy, a common cancer treatment, delivers energy to destroy tumor cells and can be used for various malignancies [8]. It can be the primary treatment or combined with surgery or chemotherapy. Precise planning is crucial, involving patient immobilization, imaging, target region delineation, and dose/schedule determination. This complex and time-consuming process requires collaboration among radiation oncologists, medical physicists, radiation therapists, and other specialists.

Manual contouring is a labor-intensive and time-consuming step in the radiation therapy planning workflow, placing a significant burden on clinicians [6]. Also, delineating tumors in anatomically complex regions or those with irregular shapes and indistinct boundaries can be particularly challenging and prone to variability. These limitations underscore the need for automated or semi-automated solutions to improve efficiency and enhance consistency of radiation therapy for cancer patients [14].

Tumor contour localization during treatment planning is akin to image segmentation in computer vision. Artificial intelligence, particularly computer vision (a well-developed field in deep learning with models like LeNet [5], AlexNet [4], VGG [10], ResNet [2], Inception [12] for classification; Mask RCNN [1], YOLO [7] for object detection; and U-Net [9, 15] for segmentation), holds potential to assist in this task.

While 2D vision models are relatively advanced, 3D applications face challenges, primarily computational power limitations. Training 3D vision models remains difficult despite GPU advancements. Directly converting 2D models to 3D often exceeds hardware memory constraints due to the added dimension's memory increase.

*Corresponding author.
E-mail address: stephan.kuo@gmail.com (C.-F. Kuo), fstsai@mail.cmu.edu.tw (F.-S. Tsai)
Accepted: April 21, 2025.

One approach to address this is using smaller input sizes (cropping 3D data into cubes) with post-processing for final predictions. However, in segmentation, this can create checkerboard effects at cube borders during post-processing. A significant limitation is also the reduced receptive field, hindering the model's ability to capture broader spatial information, relying only on local voxel values.

Full-size medical images as training data can be particularly beneficial. Given the universal nature of human anatomy, models can learn normal anatomical structures and features, potentially enhancing abnormality detection, similar to how physicians learn to recognize normal images before identifying anomalies. Training on cropped volumes deprives the model of this opportunity. Furthermore, the volume of interest (tumors) might be split across cubes, disrupting tumor contours and potentially impairing the model's ability to accurately segment them.

To mitigate these challenges in training 3D convolutional neural networks, this research proposes a 2.5D ensemble learning framework. We first established a baseline by training a model on cropped 3D medical image cubes. Subsequently, we designed a workflow to implement each component of the 2.5D ensemble learning framework, utilizing semi-2D images for training and leveraging 3D image characteristics for post-processing and assembling predictions from multiple models.

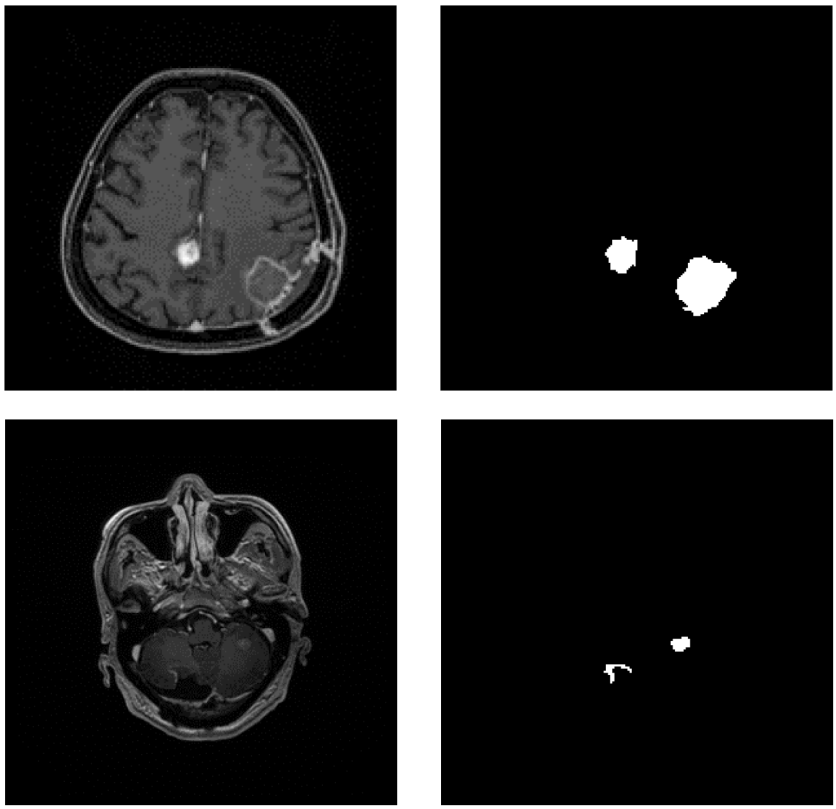

FIGURE 1. Samples of brain MRI images (left) and corresponding tumor segmentation masks (right)

2. METHODS

2.1. **Data acquisition and augmentation.** Data used in this research were collected from 2020. The 3rd Asia Cup Brain Tumor Segmentation Challenge, which consists of 1500 contrast-enhanced T1-weighted brain MRI scans along with their corresponding tumor segmentation masks (see Figure 1 for examples). Key characteristics of this original dataset are summarized in the first row of Table 1. These

TABLE 1. Characteristics of the datasets

Dataset	Number of Data	Number of TCD	Total Tumor Voxels	Tumor Ratio (%)	Number of NATR	Average Tumor Voxels in Each NATR
Original	1500	1500	8456.71 ± 11810.34	0.0468 ± 0.0705	1.7920 ± 2.0665	6092.76 ± 8873.04
3D_128	21745	3007	4218.51 ± 7262.35	0.2012 ± 0.3463	1.4014 ± 0.9676	3463.17 ± 6202.33
3D_64	67348	12447	4406.91 ± 7343.30	0.2101 ± 0.3502	1.4461 ± 1.0627	3577.83 ± 6299.86
3D_32	241636	56340	4298.19 ± 7166.04	0.2050 ± 0.3417	1.4733 ± 1.1056	3447.24 ± 6082.40
3D_16	971596	267993	4263.48 ± 7118.11	0.2033 ± 0.3394	1.5068 ± 1.1426	3364.38 ± 5977.20
Axial_1	147060	22868	554.71 ± 681.77	0.5417 ± 0.6658	1.1868 ± 0.4940	497.26 ± 625.68
Coronal_1	689584	64647	196.22 ± 229.27	0.1916 ± 0.2239	1.3533 ± 0.7443	164.11 ± 200.93
Sagittal_1	689706	65690	193.10 ± 215.85	0.1886 ± 0.2108	1.2952 ± 0.6407	166.76 ± 193.92
Axial_4	142560	28615	1772.68 ± 2490.68	0.4328 ± 0.6081	1.2268 ± 0.5798	1563.64 ± 2261.38
Axial_8	135660	35751	2835.94 ± 4319.36	0.3462 ± 0.5273	1.2811 ± 0.7039	2439.37 ± 3846.46
Axial_16	124560	48271	4166.47 ± 6581.59	0.2545 ± 0.4017	1.3716 ± 0.9339	3447.29 ± 5670.03
Coronal_16	667084	92690	2189.67 ± 3186.36	0.1336 ± 0.1945	1.3719 ± 0.7969	1842.46 ± 2815.00
Sagittal_16	667206	94108	2156.68 ± 3054.60	0.1316 ± 0.1852	1.3260 ± 0.7165	1862.29 ± 2727.33

Abbreviations: TCD: tumor-containing data; NATR: non-adjacent tumor regions.
Data are expressed as mean ± standard deviation.

characteristics include the total number of data, the number of tumor-containing data, the total count of tumor voxels, the tumor ratio (presumably the proportion of tumor voxels to total voxels), the number of non-adjacent tumor regions, and the average number of tumor voxels within each of these separate regions.

2.2. **3D Approach.** In the 3D approach, each MRI image was first padded with zeros at the end of each axis to make the length of each dimension a multiple of 128. Then the image was cropped into cubes of size $(128, 128, 128)$ which would compose the training dataset. The coordinates of the first cube were $(0\text{-}128, 0\text{-}128, 0\text{-}128)$ in the images. We then slide through the MRI images with various strides to generate the cubes. For an image of size (H, W, D) after padding, usage of a stride of (s, s, s) would generate

$$\left\lfloor \frac{H-128}{s} + 1 \right\rfloor \times \left\lfloor \frac{W-128}{s} + 1 \right\rfloor \times \left\lfloor \frac{D-128}{s} + 1 \right\rfloor$$

cubes. The strides we experimented with were 128, 64, 32, and 16. As the stride decreased, the number of cubes increased exponentially. Normalization was done by dividing the voxel values by the maximum voxel value of each dataset. Characteristics of the datasets with the strides were listed in the rows in Table 1 whose dataset names were 3D_128, 3D_64, 3D_32, and 3D_16 for each of the strides used in data preprocessing.

The network architecture used in this research was the modified 3D U-Net from [3]. Like the U-Net, the architecture comprised an encoder and a decoder component with connections to each other to combine features from various scales. The encoder part was like a convolutional neural network for classification without the final output layer, decreasing the spatial resolution and increasing the feature maps as the architecture went deeper. The encoder modules contained here were "pre-activation residual blocks," composed of two $3 \times 3 \times 3$ convolutional layers with a dropout layer ($p = 0.3$) in between. A convolutional layer was defined as an activation followed by a convolution. The encoder modules were connected with a convolution layer of stride 2, which down-sampled the feature maps and reduced the memory required. The number of filters was doubled in each module. There were five modules contained in the encoder component. After that were the decoding modules. In each decoding module, the feature maps were first upsampled with the nearest neighbor method and then concatenated with the output of the corresponding encoder module that shared the same size. Connecting decoder modules was a $1 \times 1 \times 1$ convolution which halved the number of filters. Deep supervision was also applied

in the decoder component, combining segmentation layers from each level of modules. The segmentation layers from each module were calculated by applying a $1 \times 1 \times 1$ convolution to the outputs of each module with the number of filters same as the number of classes and then upsampled to the final segmentation size. Element-wise summation was applied to the segmentation layers to form the final output of the network. The activation used in the architecture was a leaky ReLU with a slope of 0.01, and the batch normalizations were replaced by instance normalizations to cope with the stochasticity of using smaller batches.

5-fold cross-validation was applied in the training process. There were data from 1200 patients in the training set and 300 in the validation set. During training, only those image cubes which contained tumor voxels were fed into the network. The loss function was the dice CE loss function, which was the sum of the dice loss and the cross-entropy loss [13]. One challenge in a tumor segmentation task was label imbalance since tumor voxels were remarkably fewer than those of normal voxels. Therefore, we added the dice loss element to our loss function to cope with the imbalance problem and preserved the cross-entropy component to make the loss landscape smoother to get faster convergence. During training, we used the one-cycle policy inspired by Leslie Smith [11] to schedule the learning rate and momentum. The learning rate gradually increased in the first 0.25 part of training and decreased in the remaining process in a cosine annealing manner. On the other hand, the momentum started with a higher value and decreased in the first 0.25 and decreased in the latter part. The maximum value of the learning rate was 0.003. We used a batch size of 8. We trained the model for 30, 12, 10, and 4 epochs for the data preprocessing strides of 128, 64, 32, and 16. The optimizer was Adam. The L2 regularization was a weight decay of 0.1. Data augmentations were performed on each batch, which included dihedral transformations, rotations, and perspective transformations.

During inference, all image cubes were used to make predictions. We started with a zero-value tensor as the predicted tumor mask with the same shape as the original MRI image and added the values of the predicted cubes to their corresponded position in the tumor mask, which were values of 1 if the probability of the voxel being a tumor was greater or 0 otherwise. When the stride used in data preprocessing was less than 128, the size of the training data, there would be overlap volumes between the predicted cubes, making the largest possible value in the tumor mask greater than 1. For a stride of s, the largest possible value would be $(128/s)^3$. A threshold needed to be selected to determine whether a voxel was predicted as a tumor one. We then calculate the dice, precision, and recall score for each threshold. Since the higher the threshold was, the more agreements between image cube predictions were required for a positive prediction, we could expect that as the threshold increased, the precision score would increase, and the recall score would decrease. We then chose the threshold which brought about the highest dice score.

2.3. **2D Approach.** In the 2D approach, each MRI image was sliced into three sets of 2D images which were the axial, coronal, and sagittal views. The width and height were both 320 for all images. They were center cropped if the sizes were longer than 320 or padded to 320 otherwise, and the shape of the images was adjusted to $(1, 320, 320)$. Normalization was done by dividing the voxel values by the maximum of each MRI image. Three models were trained with the three sets of images respectively. Characteristics of the datasets were listed in the rows in Table 1 whose dataset names were Axial_1, Coronal_1, and Sagittal_1.

The network architecture used in the 2D approach was the same modified U-Net used in the 3D approach. The only difference was that the downsampling convolution layers connecting encoder modules were adjusted so that the stride of the thickness axis was 1 and those of the height and width axis remained at 2. This adaptation was made to accommodate the shorter length of the thickness axis of the image slices in the 2D approach.

5-fold cross-validation was applied in the training process. There were data from 1200 patients in the training set and 300 in the validation set. During training, only those image slices which contained

tumor voxels were fed into the network. The training procedure and hyperparameters were nearly the same as those in the 3D approach. The learning rate was 0.003 for training all three models. Epochs were 11 for the axial model and 8 for the coronal and sagittal models. The training process mentioned above was applied to all three data sets Axial_1, Coronal_1, and Sagittal_1.

During inference, all image slices were used to make predictions. We started with a zero-value tensor as the predicted tumor mask with the same shape as the original MRI image and added the values of the predicted image slices to their corresponded position in the tumor mask, which were values of 1 if the probability of the voxel being a tumor was greater or 0 otherwise. We then calculate the dice, precision, and recall score for the axial, coronal, and sagittal models.

The ensemble was done by summing up the predictions from the three models, which resulted in voxel values ranging from 0 to 3. Thresholds of 1, 2, and 3 were used to calculate the dice, precision, and recall scores. We could also expect that as the threshold increased, the precision score would increase, and the recall score would decrease. We then selected the threshold which brought about the highest dice score.

2.4. **2.5D Approach.** Like the 2D approach, the 3D images were sliced into 2D-like images, and the width and height were both 320 as well, center-cropped if the sizes were longer than 320 or padded to 320 otherwise. The difference was that instead of a depth of 1 in the 2D approach, the image slices here had a depth of greater than 1, which we referred to as a "2.5D image." When generating these 2.5D image slices, we used a stride of 1. This meant that for a depth of D and an image with length N on the depth-axis, the axis perpendicular to the image plane, there were $N - D + 1$ image slices generated. We adjusted the shape of the image slice to $(D, 320, 320)$. Normalization was done by dividing the voxel values by the maximum of each MRI image. We experimented with various depths of 4, 8, and 16. As for the image view used, we started with the axial view images and compared the performances of models trained with image slices of different depths. We expected that the "thicker" the image slices were, the better the performance would be. After identifying the best-performing model, we trained another two models with coronal and sagittal view images of the same image depth. Characteristics of the datasets were listed in the rows in Table 1 whose dataset names were Axial_4, Axial_8, Axial_16, Coronal_16, and Sagittal_16.

The network architecture used in the 2.5D approach was the same modified U-Net used in the 2D approach with the adjustments of the strides in the downsampling convolutions to fit for the shorter length of the depth-axis of the image slices.

5-fold cross-validation was applied in the training process. There were data from 1200 patients in the training set and 300 in the validation set. During training, only those image slices which contained tumor voxels were fed into the network. The training procedure and hyperparameters were nearly the same as those in the 2D approach. The learning rate was 0.003 for training all models in the 2.5D approach. Epochs were 11 for training all axial models and 8 for the coronal and sagittal models. The training process mentioned above was applied to all datasets Axial_4, Axial_8, Axial_16, Coronal_16, and Sagittal_16.

During inference, all image slices were used to make predictions. We started with a zero-value tensor as the predicted tumor mask with the same shape as the original MRI image and added the values of the predicted image slices to their corresponded position in the tumor mask, which were values of 1 if the probability of the voxel being a tumor was greater or 0 otherwise. Since a stride of 1 was used when generating the 2.5D images as described in the data preprocessing section, besides the periphery of the image, for a depth of D of the image slice, voxels in the original 3D image were contained in D image slices, making the greatest possible value of a voxel in the predicting tumor mask D. A threshold needed to be selected to determine whether a voxel was predicted as a tumor one. We then calculate the dice, precision, and recall score for each threshold. Since the higher the threshold was, the more

TABLE 2. Comparison of performances between models trained with different strides used in data preprocessing in the 3D approach. The input size of the training set was (128, 128, 128)

Stride	Threshold	Dice	Precision	Recall
128	1	0.2327 ± 0.0195	0.1836 ± 0.0189	0.5240 ± 0.0174
64	4	0.5071 ± 0.0191	0.6050 ± 0.0169	0.4976 ± 0.0260
32	11	0.5683 ± 0.0228	0.6206 ± 0.0199	0.5840 ± 0.0296
16	38	0.5694 ± 0.0189	0.6304 ± 0.0194	0.5789 ± 0.0236

Data are expressed as mean $\pm$ standard deviation.

agreements between image slice predictions were required for a positive prediction, we could expect that as the threshold increased, the precision score would increase, and the recall score would decrease. We then chose the threshold which brought about the highest dice score.

The ensemble was done by summing up the predictions from the three models, the axial, coronal, and sagittal models, which made the greatest possible voxel value of $3D$ for a depth of D of the image slices. Thresholds ranging from 1 to $3D$ were used to calculate the dice, precision, and recall scores. We then selected the threshold which brought about the highest dice score.

3. RESULTS

3.1. **3D Approach.** When the stride used for data preprocessing was 128, the dice, precision, and recall scores of the model were 0.2327, 0.1836, and 0.5240, respectively. As for the performance of the model with a stride of 64, the dice, precision, and recall scores were 0.5071, 0.6050, and 0.4976. The model that had a data preprocessing stride of 32 had a dice score of 0.5683, a precision score of 0.6206, and a recall score of 0.5840. The model trained with data using a stride of 16 during preprocessing achieved a dice score of 0.5694, a precision score of 0.6304, and a recall score of 0.5789 (see Table 2).

In the 3D approach, reducing the stride used during data preprocessing generally led to improvements in dice score, precision, and recall. A significant performance boost was observed early in the experiment, particularly when the stride was initially halved from 128 to 64. The dice score more than doubled, increasing from 0.2327 with a stride of 128 to 0.5071 with a stride of 64. Precision saw an even more substantial increase, tripling from 0.1836 (stride 128) to 0.6050 (stride 64). However, recall slightly decreased from 0.5240 to 0.4976 during this initial stride reduction. Subsequently, when the stride was further reduced from 64 to 32, all three metrics improved: dice score rose to 0.5683, precision to 0.6206, and recall to 0.5840. Towards the end of the experiment, the rate of improvement plateaued, with smaller gains observed when the stride decreased from 32 to 16 (dice: 0.5683 to 0.5694; precision: 0.6206 to 0.6304; recall: 0.5840 to 0.5789). Overall, decreasing the stride yielded more significant gains in evaluation metrics in the initial stages of the experiment, with the rate of improvement diminishing as the stride became smaller.

3.2. **2D Approach.** The model trained with axial image sections had a dice score of 0.2110, a precision score of 0.1489, and a recall score of 0.5827. As for the model that used coronal images for training, the dice score was 0.1532, the precision score was 0.1031, and the recall score was 0.5965. Additionally, the model that used images of sagittal views as the dataset reached a dice score of 0.1678, a precision score of 0.1125, and a recall score of 0.5897. Finally, the ensemble model that took the average of the predictions of the axial, coronal, and sagittal models achieved a dice score of 0.4817, a precision score of 0.6339, and a recall score of 0.4367 (see Table 3).

Comparing axial, coronal, and sagittal models, the axial model achieved the highest dice score (0.2110), outperforming the coronal (0.1532) and sagittal (0.1678) models. Similarly, the axial model exhibited the

TABLE 3. Model performances of the 2D approach. The input size of the training set was (1, 320, 320)

Model	Threshold	Dice	Precision	Recall
Axial	1	0.2110 ± 0.0116	0.1489 ± 0.0102	0.5827 ± 0.0182
Coronal	1	0.1532 ± 0.0502	0.1031 ± 0.0395	0.5965 ± 0.0253
Sagittal	1	0.1678 ± 0.0171	0.1125 ± 0.0134	0.5897 ± 0.0235
Ensemble	3	0.4817 ± 0.0165	0.6339 ± 0.0129	0.4367 ± 0.0183

Data are expressed as mean $\pm$ standard deviation.

TABLE 4. Comparison of performances between models trained with axial image slices of different depths. The input size of the training set was (D, 320, 320) with D being the depth of the image slice

Depth	Threshold	Dice	Precision	Recall
1	1	0.2110 ± 0.0116	0.1489 ± 0.0102	0.5827 ± 0.0182
4	4	0.4712 ± 0.0208	0.5021 ± 0.0320	0.5156 ± 0.0109
8	7	0.5445 ± 0.0201	0.6031 ± 0.0341	0.5611 ± 0.0184
16	12	0.5931 ± 0.0174	0.6443 ± 0.0226	0.6071 ± 0.0171

Data are expressed as mean $\pm$ standard deviation.

best precision (0.1489) compared to the coronal (0.1031) and sagittal (0.1125) models. In contrast, the recall scores were relatively similar across all three models: axial (0.5827), coronal (0.5965), and sagittal (0.5897).

Notably, the ensemble model, combining the axial, coronal, and sagittal models, demonstrated a significant increase in the dice score (0.4817). This was more than double the axial model's dice score (0.2110) and over three times that of the coronal model (0.1532). The ensemble model also showed a substantial improvement in precision (0.6339), exceeding the axial model's precision by more than four times and the coronal model's by over six times. However, the ensemble model's recall score (0.4367) was lower than that of each individual model (ranging from 0.5827 to 0.5965).

3.3. **2.5D Approach.** In the experiment using images of various depths, the model trained with images whose depth was 1 had a dice score of 0.2110, a precision score of 0.1489, and a recall score of 0.5827. As for the model that used images with a depth of 4 for training, the dice score was 0.4712, the precision score was 0.5021, and the recall score was 0.5156. Additionally, the model that used images whose depth was 8 as the dataset reached a dice score of 0.5445, a precision score of 0.6031, and a recall score of 0.5611. Finally, the model trained with the dataset whose depth was 16 achieved a dice score of 0.5931, a precision score of 0.6443, and a recall score of 0.6071 (see Table 4).

The evaluation metrics generally improved with increasing depth. The most significant improvement occurred between depths 1 and 4. Specifically, the dice score for depth 4 was more than double that of depth 1, and the precision for depth 4 was over three times higher than that of depth 1. Overall, dice scores showed a gradual increase with depth: 0.2110 (depth 1), 0.4712 (depth 4), 0.5445 (depth 8), and 0.5931 (depth 16). Precision followed a similar trend: 0.1489 (depth 1), 0.5021 (depth 4), 0.6031 (depth 8), and 0.6443 (depth 16). Recall scores also showed incremental increases from depth 4 (0.5156) to depth 8 (0.5611) and depth 16 (0.6071). However, the recall for depth 1 (0.5827) was higher than those for depths 4 and 8, only being surpassed by the recall of the depth 16 model. Consequently, a depth of 16 was chosen for the dataset images used to train and assemble the axial, coronal, and sagittal models.

The model trained with axial image sections had a dice score of 0.5931, a precision score of 0.6443, and a recall score of 0.6071. As for the model that used coronal images for training, the dice score was

TABLE 5. Model performances of the 2.5D approach. The input size of the training set was (16, 320, 320)

Model	Threshold	Dice	Precision	Recall
Axial	12	0.5931 ± 0.0174	0.6443 ± 0.0226	0.6071 ± 0.0171
Coronal	16	0.4865 ± 0.0477	0.5696 ± 0.0839	0.4947 ± 0.0234
Sagittal	16	0.5065 ± 0.0235	0.5828 ± 0.0323	0.5082 ± 0.0134
Ensemble	24	0.6011 ± 0.0189	0.6527 ± 0.0153	0.6162 ± 0.0228

Data are expressed as mean $\pm$ standard deviation.

0.4865, the precision score was 0.5696, and the recall score was 0.4947. Additionally, the model that used images of sagittal views as the dataset reached a dice score of 0.5065, a precision score of 0.5828, and a recall score of 0.5082. Finally, the ensemble model that took the average of the predictions of the axial, coronal, and sagittal models achieved a dice score of 0.6011, a precision score of 0.6527, and a recall score of 0.6162 (Table 5).

Among the axial, coronal, and sagittal models, the axial model demonstrated the highest performance in terms of segmentation accuracy, achieving a dice score of 0.5931. In comparison, the coronal and sagittal models yielded lower dice scores of 0.4865 and 0.5065, respectively. Furthermore, the axial model exhibited the best precision among the three models, with a score of 0.6443, outperforming the coronal model (0.5696) and the sagittal model (0.5828). Similarly, the axial model achieved the highest recall score (0.6071), indicating its superior ability to capture all relevant positive instances, while the coronal and sagittal models showed lower recall scores of 0.4947 and 0.5082, respectively.

When the axial, coronal, and sagittal models were combined into an ensemble model, the evaluation metrics showed a slight improvement across the board compared to the axial model. Specifically, the ensemble model achieved a dice score of 0.6011, a marginal increase from the axial model's score of 0.5931. Similarly, the ensemble model's precision score was 0.6527, slightly better than the axial model's 0.6443. For recall, the ensemble model reached 0.6162, a small gain over the axial model's 0.6071. Note that the 2D ensemble showed substantial improvements in dice and precision scores but had a decrease in recall. Conversely, the 2.5D ensemble model demonstrated small improvements across all evaluation metrics. Both the axial and ensemble models in the 2.5D approach exhibited superior performance compared to the best model in the 3D approach (trained with a preprocessing stride of 16).

4. DISCUSSION

Generally, smaller strides in data preprocessing led to improved model evaluation metrics. This improvement can be attributed to two main effects. First, reducing the stride exponentially increased the amount of training data. This was particularly beneficial for training neural networks, which typically require large datasets to achieve optimal performance. While techniques like transfer learning can mitigate the need for extensive data by leveraging models pre-trained on other tasks (e.g., ImageNet), the limited availability of suitable pre-trained models in the medical imaging domain prevented its use in this research. Nevertheless, the increased data from a smaller stride in data preprocessing effectively acted as a form of data augmentation, exposing the model to more diverse cubic segments from the original MRI images. Second, a smaller stride during preprocessing also positively impacted inference. The model output had the same (128,128,128) shape as the input cubes. To obtain a final prediction mask matching the original MRI image size, we aggregated the predictions of individual data cubes at their corresponding locations. A smaller stride resulted in greater overlap between these cubes, with a maximum overlap factor of $(128/s)^3$ for a stride of s. This increased overlap meant that more individual cube predictions contributed to the final prediction of each voxel.

Concerning the effects of ensemble learning, in the 2D approach, assembling axial, coronal, and sagittal models significantly enhanced performance, notably boosting the precision score fivefold compared to the average. This improvement likely stemmed from the requirement of greater agreement among the models for a positive prediction, effectively correcting many false negatives present in individual prediction masks. While this came with a trade-off of a quarter decrease in the average recall score, the dice score saw a substantial increase to 2.7 times the average. This observation implied that the prediction robustness for specific voxels varied across different views. For example, a voxel may be a part of an extremely small tumor area in an axial section located at the upper or lower periphery of the tumor. Predicting the tumor mask in an axial view would be particularly difficult due to the more severe imbalance between the normal and tumor voxels. However, the coronal or sagittal section of the mentioned voxel may contain a larger tumor area which would make segmentation relatively easier. Thus, ensembling models trained on different views improved prediction accuracy. Conversely, the 2.5D approach yielded only slight improvements in evaluation metrics through ensembling compared to the axial model's results. The less pronounced benefit in the 2.5D model indicates that increasing the image depth already provided sufficient 3D spatial information to the models, thereby limiting the potential for further enhancement from model ensembling across different views.

As a result, the 2.5D ensemble learning framework outperforms the 3D approach for training models on 3D intracranial tumor segmentation. Furthermore, in situations with constrained computational power where the 3D approach is impractical, the 2.5D ensemble learning framework offers a suitable alternative, allowing for a reduction in input data depth to a manageable level.

Statements and Declarations

The authors declare that they have no conflict of interest. This work is adapted from the author C.-F. Kuo's master's thesis.

Acknowledgments

This research was based on data collected from 2020 The 3rd Asia Cup Brain Tumor Segmentation Challenge hosted by the National Taiwan University (NTU) Medical Genie Project team. This work was supported in part by the National Science and Technology Council, Taiwan, and in part by the China Medical University.

References

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Venice, Italy, 2017. IEEE.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas, 2016. IEEE.

[3] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maier-Hein. Brain tumor segmentation and radiomics survival prediction: Contribution to the brats 2017 challenge. In A. Crimi, S. Bakas, H. Kuijf, B. Menze, and M. Reyes, editors, *International MICCAI Brainlesion Workshop*, volume 10670, pages 287–297, BrainLes 2017. Lecture Notes in Computer Science, 2017. Springer, Cham.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, New York, 2012.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] J. Liu, H. Xiao, J. Fan, W. Hu, Y. Yang, P. Dong, L. Xing, and J. Cai. An overview of artificial intelligence in medical physics and radiation oncology. *Journal of the National Cancer Center*, 3(3):211–221, 2023.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, Las Vegas, USA, 2016. IEEE.

[8] S. H. Revell. Relationship between chromosome damage and cell death. In *Radiation-Induced Chromosome Damage in Man*, pages 215–233. Liss AR, New York, 1983.

[9] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. Wells, and A. Frangi, editors, *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, MICCAI 2015, Munich, Germany, 2015. Springer, Cham.

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 1409.1556, 2014.

[11] L. N. Smith and N. Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pages 369–386. SPIE, 2019.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, Boston, USA, 2015. IEEE.

[13] S. A. Taghanaki, Y. Zheng, S. K. Zhou, B. Georgescu, P. Sharma, D. Xu, D. Comaniciu, and G. Hamarneh. Combo loss: Handling input and output imbalance in multi-organ segmentation. *Computerized Medical Imaging and Graphics*, 75:24–33, 2019.

[14] Y. Zhong, Y. Yang, Y. Fang, J. Wang, and W. Hu. A preliminary experience of implementing deep-learning based auto-segmentation in head and neck cancer: A study on real-world clinical cases. *Frontiers in Oncology*, 11:e638197, 2021.

[15] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang. UNet++: A nested U-net architecture for medical image segmentation. In D. Stoyanov, editor, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, volume 11045, pages 3–11, DLMIA ML-CDS, 2018, Spain, 2018.