Lung Tumor Segmentation using 3D U-Net and Ensemble Learning

Anonymous

***Abstract*—This study investigates the challenges and potential of lung tumor segmentation techniques, with a focus on optimizing hyperparameters, data augmentation, and ensembling methods. The optimized hyperparameters converged a Dice score of 0.36. Data augmentation techniques resulted in lower Dice scores, highlighting the need for further optimization or additional preprocessing steps. Sampling from erroneous regions improved the validation Dice score to 0.41. Weighted ensembling did not yield performance improvements due to the limited availability of additional models. Visualization of uncertainty maps revealed the model's struggles with tumor edges and small tumors, suggesting that future research should focus on edge detection and handling of small tumors to further enhance performance.**

***Index Terms*—Lung Tumor; U-Net; Ensemble Learning**

I. Introduction

Lung tumor segmentation is a crucial step in medical imaging, as it facilitates the accurate diagnosis, treatment planning, and monitoring of lung cancer. Despite its importance, lung tumor segmentation presents several challenges due to the small size of tumors and the large volume of lung data [1] To address these challenges, this coursework study introduces the use of a 3D U-Net, a convolutional network designed for segmentation [2]. Furthermore, ensemble learning, a technique that combines multiple learning algorithms to improve predictive performance is used in addition to various other preprocessing techniques [3]. The primary objective of this study are to investigate the efficacy of 3D U-Net and ensemble learning in lung tumor segmentation.

II. Material and Methods

A. Data

The Medical Segmentation Decathlon [1] (MSD) is an international challenge designed to assess the comparative performance of image analysis algorithms across multiple tasks and modalities. It focuses on the hypothesis that an algorithm performing well in multiple tasks will generalize well to previously unseen tasks, possibly outperforming custom-designed solutions. In this coursework study, I focus on the lung tumor task, utilizing a dataset consisting of preoperative thin-section CT scans from 96 patients with non-small cell lung cancer. The primary challenge lies in segmenting small tumor regions in images with a large field-of-view. The dataset, acquired from the Cancer Imaging Archive, was partitioned into 64 training samples, of which 58 were used for training and 6 for validation.

B. 3D U-Net Architecture

The 3D U-Net architecture [2], has emerged as a powerful approach for volumetric medical image segmentation. In this coursework study, I utilize the MONAI framework [4] to implement the 3D U-Net, with the final network structure adapted to suit the lung tumor segmentation task. The key advantage of employing a 3D network over its 2D counterpart lies in its ability to leverage volumetric context and spatial information, thus providing a more comprehensive understanding of the underlying structures. This is particularly crucial for medical imaging, where capturing the relationships between neighboring slices can lead to improved segmentation accuracy. By implementing the 3D U-Net architecture in this study, I aim to exploit the rich volumetric information present in the lung tumor dataset and enhance our segmentation performance accordingly. For detailed implementation of the 3D U-Net network, please refer to **Section 10** and **Step 1** in the attached Jupyter notebook.

C. Top-K Loss Function

In this coursework, I implement a variation of the Top-K Loss function proposed by [5], addressing the class imbalance problem due to the small size of lung tumors. The Top-K Loss function is particularly relevant for lung tumor segmentation tasks, as it can effectively balance the learning between the majority and minority classes, ensuring that the model learns from the hard-to-discriminate pixels and provides better segmentation results.

The Online Bootstrapping Loss class (see Section 6 in the notebook) combines Dice Loss with a modified Binary Cross Entropy (BCE) loss based on the top K hardest pixels. I first apply a sigmoid activation to the logits, calculate BCE loss, and threshold it based on the tumor fraction, which is the fraction of tumor pixels in the target. A mask is then applied to BCE loss values, and the mean of masked BCE loss is calculated.

During the first forward pass, I initialize the BCE and Dice loss values, and normalize the masked BCE loss. The combined loss is calculated using an alpha parameter, which determines the contribution of normalized masked BCE loss and Dice loss. This implementation, as detailed in the attached Jupyter notebook **Section 6**, enables the network to focus on more challenging pixels during training, improving the segmentation of small lung tumors.

D. Training and Validation

The training loop processes the dataset in batches, with each batch containing multiple patches to accommodate the large size of the volumetric data. I used the Adam optimizer for its adaptive learning rate capabilities and efficient performance. The training process involves a forward pass through the model, followed by the calculation of the loss using the Top-K Loss function. Subsequently, gradients are computed, and the optimizer updates the model weights. The code for training and validation process and be found at Section 9 in the attached Jupyter Notebook.

To monitor the performance of the model, I logged the training loss and Dice Score for every 10 steps. The Dice Score was chosen as the evaluation metric because it provides a balance between precision and recall, making it suitable for segmentation tasks. In addition to the training and validation process described, at every 10 iterations of each epoch, a visual validation step is incorporated to help assess the model's performance on the validation dataset. This visual validation consists of extracting a 2D slice from the volumetric data, specifically selecting the slice with the largest area of tumor size. The logits and the model's predictions (obtained after applying the sigmoid function and applying a threshold) at the corresponding location in the 3D image are also visualized alongside the 2D slice. This visual validation step provides a helpful qualitative assessment of the model's performance, as it allows to visually inspect the model's predictions in comparison to the ground truth.

During the validation phase, the model is set to evaluation mode, and the gradients are not calculated. I validate the model on an entire batch and compute the validation loss and Dice Score. To evaluate the model on entire images, I compute the mean Dice Score across all the images in the validation set. The model checkpoints are saved during the training process, and if the mean Dice Score on entire images surpasses 0.65, the training is terminated. This approach allows us to obtain a model that generalizes well to new data while mitigating the risk of overfitting.

E. Data Augmentation

In this study, I employed data augmentation techniques for training, specifically Affine transformations and Elastic Deformations, to improve the generalizability of the model. These techniques were applied using custom functions as detailed in the attached Jupyter Notebook (see **Section 2**).

Affine transformations, such as rotations, shearing, translations, and scaling, were applied with a probability of 0.2. The chosen parameters for these transformations ensured a reasonable level of augmentation without causing extreme deformations, which could negatively affect model performance.

Elastic Deformations were also introduced with a probability of 0.2, to simulate non-linear transformations in the data. I specified the sigma and magnitude ranges to control the smoothness and intensity of the deformation, respectively. Additionally, Gaussian noise was added to the images with a probability of 0.2, using a mean of 0.0 and standard deviation of 0.1 to simulate potential variations in image quality.

F. Hyperparameter Optimization

Hyperparameter optimization was conducted using a random search approach and the Weights & Biases logger [6] to search across various parameter combinations using a random search. The code implementation can be found in **Step 3** of the attached Jupyter Notebook. Configuration included parameters such as

- Channels
- Batch size
- Number of residual units
- Number of patches
- Patch size
- Learning rate
- Drop out rate
- Activation function
- Alpha value for the loss function

Considering the memory constraints of our 12GB GPU, the optimized configuration was selected based on its superior performance.

G. Sampling from Erroneous Regions

Sampling from erroneous regions is a technique to address class imbalance and improve training efficiency on sparse datasets [7]. This method, also known as the isample algorithm, focuses on selecting patches with high error rates for training. The algorithm initializes error maps for each image in the training dataset and adapts the sampling process based on the errors produced by the current UNet weights. By concentrating on areas where the network struggles, it efficiently trains the model and potentially improves generalization.

Due to computational constraints, error maps are updated every three epochs and saved as *.npy* array files on the local drive, avoiding memory errors. However, the process of updating error maps significantly increases the training time (estimated 30 hours to reach 8,000 steps), leading to the project being conducted without using it for the rest of the experiments. Early-stage observations showed no significant performance improvement when the model is trained from scratch.

However, one interesting observation is that during random sampling, the amount of patches containing any tumor pixel is not controlled, usually accounting for only about 10 percent of the patches (5/48) where the batch size is 4 and the patch size is 12. However, after applying the isample method to trained models (trained for 8,000 steps), the number increases to around 10/48 and the validation dice score has increased significantly. This indicates that the isample method is more effective at focusing on regions near the tumor pixels, which are areas where the model is likely to have higher error rates due to the class imbalance problem. By targeting these challenging regions more frequently during training, the isample method has the potential to better address class imbalance and improve the model's ability to identify tumor pixels. However, as mentioned, the increased training time and lack of observed performance improvement in this particular study led to the exclusion of the isample method from further experiments when training a model from scratch.

In this coursework study, the two components of the isample algorithm are implemented in the dataset (reading and use error map) and training loop (updating error map), with details available in **Section 3 and 9** of the attached Jupyter Notebook.

H. Uncertainty Estimation

The Augmentation-based Aleatoric method, proposed by [8], for uncertainty estimation consists of two main parts. The first part focuses on a mathematical representation of ensembles of predictions using multiple transformed versions of the input, and the second part calculates the diversity of these prediction results to estimate aleatoric uncertainty.

The Monte Carlo simulation with augmented versions of the input image was used to obtain pixel-wise uncertainty using entropy. This was achieved by running the model multiple times with slightly different versions of the input image using transformations such as:

- Translation
- Scale
- Rotations
- Flips

Each transformation is implemented using MONAI's transforms library, which allows inversing the predictions logits to the pixel-wise location of the original input image. The resulting probability maps were combined to generate a final prediction and an entropy map for each pixel, representing the model's uncertainty in its prediction.

To visualize the uncertainty estimates, a validation function was used to take 2D slices of the 3D image, label, output, and entropy map and concatenate them horizontally (see **Section 4** of the Jupyter Notebook). The image slices were normalized for better visualization, and the uncertainty tended to be at the edge of the tumors (see **Step 5** in the Notebook).

III. Ensemble Learning

A. Model Averaging

Model ensemble is a technique when multiple models are trained independently and their predictions are combined during testing [3]. In this approach, each model segments an unseen image and produces class-confidence maps. These maps are then integrated into an ensemble model, by averaging the confidence scores for each class across all models. The final segmentation is determined by assigning each voxel the class with the highest confidence.

The main objective for averaging is to leverage the diverse knowledge of individual models within the ensemble, thereby improving overall performance. The ensemble's true posterior is approximated by marginalizing the effects of model-specific parameters. This approach can be seen as a relaxation of a pre-existing neglected strong prior.

In this project, different models developed by group members are ensembled by averaging their logits. It is crucial to analyze the impact of each individual model on the ensemble's performance and consider excluding models that might negatively affect the results. For a details of the implementation, refer to the attached Jupyter Notebook (see **Step 6 and 7**).

B. Weighted Ensembling

Weighted ensembling assigns different weights to individual models in the ensemble, reflecting their contribution to the overall prediction. In this section, a loop function is used to test various relative weights for the two models, U-Net and Deep Medic. The weights range from 0 to 1, incremented by 0.1, representing the proportion of each model's contribution to the ensemble. This approach allows for fine-tuning the weights to achieve better performance, as opposed to simple averaging, which gives equal importance to each model.

The choice of weights plays a crucial role in determining the ensemble's performance. It is essential to consider the individual model's performance when assigning weights, as a poorly performing model can negatively affect the overall result. By carefully selecting the weights, it is possible to optimize the ensemble and achieve better performance than with simple averaging. For a detailed understanding of the implementation, refer to the attached Jupyter Notebook (see **Step 6** in the Notebook).

IV. Results

The performance improvements achieved in each step are as follows:

- Hyperparameter optimization: The optimized hyperparameters reached a Dice score of 0.25 after 4,000 iterations, eventually converging at 0.36 (see **Step 3** of Jupyter Notebook; see Figure 1 for the randomly selected parameters and the result validation loss at iteration 4,000).
- Data augmentation: The applied techniques led to convergence at a lower Dice score of around 8,500 iterations, suggesting a need for further optimization of augmentation parameters or additional preprocessing steps. However, the augmentation techniques resulted in both the training and validation performance converging to a lower Dice score at around 8,500 iterations, indicating a decrease in model performance. This outcome suggests that the chosen parameters for augmentation might require further optimization, or that additional data preprocessing steps may be necessary to maximize the benefits of data augmentation for the lung tumor segmentation task.
- Sampling from erroneous regions: The model, initially trained on identified hyperparameters using random sampling, was further trained for 1000 epochs, converging at a validation Dice score of 0.41.
- Weighted ensembling: No performance improvement was observed due to Deep Medic's (the only external model) lack of contribution (see **Step 6** for implementation details). In addition, the initial results by average show a drop in performance when compared to a single U-Net model, with a validation Dice score decreasing from 0.41 to 0.29. This could be attributed to the inclusion of a Deep Medic model, which has a lower validation Dice score of 0. Interestingly, as the weights of the U-Net increase from 0 to 1, and the Deep Medic's weights decrease from 1 to 0, the performance experiences a steady increase,

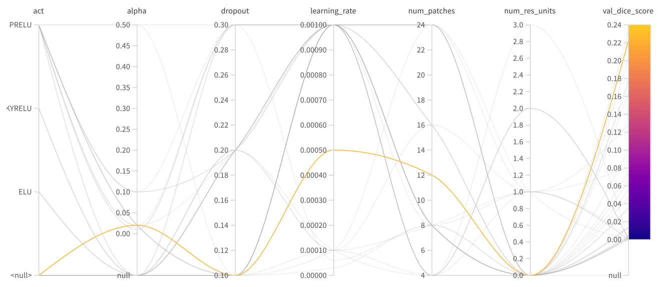

Fig. 1. The figure shows the hyperparameters randomly selected for the hyperparameter optimization step and their corresponding validation loss at iteration 4,000 during training of the lung tumor segmentation model.

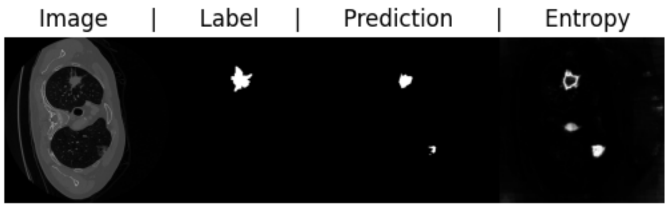

Fig. 2. Visualization of uncertainty maps: The figure shows an example of uncertainty maps generated by the model, highlighting areas of increased certainty in clear tumor regions and higher uncertainty in challenging areas such as tumor edges or small tumors. These insights suggest a need for improvements in edge detection and handling of small tumors

highlighting the superiority of the U-Net in this specific case.

Visualization of uncertainty maps revealed the model's increased certainty in clear tumor regions and uncertainty in challenging areas like tumor edges or small tumors (see Figure 2). These insights suggest that the model's uncertainty is higher in areas where tumor identification is difficult, indicating a potential need for improvements in edge detection and handling of small tumors.

V. Discussion

The results of this study provide valuable insights into the challenges of lung tumor segmentation and the potential of various techniques to improve performance. However, several limitations were identified, which could be addressed in future research to further enhance the performance of segmentation models.

Despite the attempts to optimize data augmentation techniques, a decrease in model performance was observed, highlighting the need for further optimization of augmentation parameters or additional preprocessing steps.

Additionally, the study was limited by the availability of only two models, U-Net and Deep Medic, for ensembling. As demonstrated, the performance improves when the U-Net model is assigned a higher weight, eventually stopping when the weight reaches 1. This suggests that the U-Net model contributes significantly to the ensemble's performance, while the Deep Medic model may not have a substantial impact. Therefore, introducing a wider variety of models could lead to a more robust ensemble performance.

Although some improvements were achieved with the current hyperparameter optimization, there may still be room for further fine-tuning to yield better performance. Moreover, exploring different model architectures with a particular focus on edge detection and handling small tumors could result in performance enhancements.

To address these limitations and continue building on the insights gained in this study, future research directions may include:

- Thorough hyperparameter tuning: The potential for performance improvement through manual layer additions suggests that more extensive hyperparameter optimization, given sufficient computational power, could identify better configurations for the segmentation task.
- Advanced ensembling techniques: Investigating different ensemble learning techniques, such as stacking, bagging, or boosting, and incorporating a diverse set of models could lead to improved performance in lung tumor segmentation tasks.

By addressing these limitations and exploring the suggested research directions, the performance of lung tumor segmentation models can be further improved, ultimately contributing to more accurate diagnoses and improved patient outcomes.

VI. Conclusion

In conclusion, this study has shed light on the challenges and potential of lung tumor segmentation techniques, yet limitations remain. Future research should focus on more extensive hyperparameter optimization, advanced ensembling methods, and additional model architectures to enhance performance. By addressing these limitations and exploring new research avenues, the accuracy of lung tumor segmentation models can be significantly improved, ultimately leading to better diagnoses and patient outcomes in the fight against lung cancer.

References

[1] M. Antonelli, A. Reinke, S. Bakas, K. Farahani, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. Menze, O. Ronneberger, R. M. Summers, B. van Ginneken, M. Bilello, P. Bilic, P. F. Christ, R. K. G. Do, M. J. Gollub, S. H. Heckers, H. Huisman, W. R. Jarnagin, M. K. McHugo, S. Napel, J. S. G. Pernicka, K. Rhode, C. Tobon-Gomez, E. Vorontsov, J. A. Meakin, S. Ourselin, M. Wiesenfarth, P. Arbeláez, B. Bae, S. Chen, L. Daza, J. Feng, B. He, F. Isensee, Y. Ji, F. Jia, I. Kim, K. Maier-Hein, D. Merhof, A. Pai, B. Park, M. Perslev, R. Rezaiifar, O. Rippel, I. Sarasua, W. Shen, J. Son, C. Wachinger, L. Wang, Y. Wang, Y. Xia, D. Xu, Z. Xu, Y. Zheng, A. L. Simpson, L. Maier-Hein, and M. J. Cardoso, "The medical segmentation decathlon," *Nature Communications*, vol. 13, no. 1, Jul. 2022. [Online]. Available: https://doi.org/10.1038/s41467-022-30695-9

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[3] K. Kamnitsas, W. Bai, E. Ferrante, S. McDonagh, M. Sinclair, N. Pawlowski, M. Rajchl, M. Lee, B. Kainz, D. Rueckert, and B. Glocker, "Ensembles of multiple models and architectures for robust brain tumour segmentation," 2017. [Online]. Available: https://arxiv.org/abs/1711.01468

[4] M. J. Cardoso, W. Li, R. Brown, N. Ma, E. Kerfoot, Y. Wang, B. Murrey, A. Myronenko, C. Zhao, D. Yang, V. Nath, Y. He, Z. Xu, A. Hatamizadeh, A. Myronenko, W. Zhu, Y. Liu, M. Zheng, Y. Tang, I. Yang, M. Zephyr, B. Hashemian, S. Alle, M. Z. Darestani, C. Budd, M. Modat, T. Vercauteren, G. Wang, Y. Li, Y. Hu, Y. Fu, B. Gorman, H. Johnson, B. Genereaux, B. S. Erdal, V. Gupta, A. Diaz-Pinto, A. Dourson, L. Maier-Hein, P. F. Jaeger, M. Baumgartner, J. Kalpathy-Cramer, M. Flores, J. Kirby, L. A. D. Cooper, H. R. Roth, D. Xu, D. Bericat, R. Floca, S. K. Zhou, H. Shuaib, K. Farahani, K. H. Maier-Hein, S. Aylward, P. Dogra, S. Ourselin, and A. Feng, "Monai: An open-source framework for deep learning in healthcare," 2022. [Online]. Available: https://arxiv.org/abs/2211.02701

[5] Z. Wu, C. Shen, and A. v. d. Hengel, "Bridging category-level and instance-level semantic image segmentation," 2016. [Online]. Available: https://arxiv.org/abs/1605.06885

[6] "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/,

[7] L. Berger, E. Hyde, M. J. Cardoso, and S. Ourselin, "An adaptive sampling scheme to efficiently train fully convolutional networks for semantic segmentation," 2017. [Online]. Available: https://arxiv.org/abs/1709.02764

[8] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren, "Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1807.07356