Medical Image Segmentation on the MSD Dataset: A Comparative Analysis of nnUNet and My Custom UNet2D Approach

Raghuram Cauligi Srinivas
rc5428@nyu.edu

***Abstract*—In this project, I address the task of medical image segmentation on the MSD dataset by comparing two deep learning approaches. I analyze the performance of the state-of-the-art, self-configuring nnUNet—trained for 50 epochs—against a custom-designed slice-wise UNet2D that I trained for 25 epochs using mixed-precision (FP16). This report discusses the differences in network architecture, my training techniques, performance metrics, and the impact of data preprocessing on segmentation accuracy.**

I. Methodology

I employed the nnUNet framework, which automatically adapts its network architecture, data augmentation, and training protocols to the dataset, enabling robust training over 50 epochs. In contrast, I developed my own custom UNet2D—a fixed 2D convolutional network with skip connections—and used FP16 training to improve efficiency. However, due to computational constraints, I was able to train my model for only 25 epochs.

Architectural Differences: I observed that the nnUNet is designed to be self-configuring and highly versatile. Its architecture is dynamically optimized based on the input data characteristics. It automatically selects between 2D and 3D convolutions, adapts patch sizes, and configures hyperparameters accordingly. The nnUNet's modular design includes sophisticated pre- and post-processing pipelines, dynamic patch-based training, and extensive data augmentation. In contrast, my custom UNet2D follows a traditional U-shaped design [1] with a fixed encoder-decoder structure. It consists of repeated blocks of two 3x3 convolutions followed by ReLU activations and max pooling in the encoder, with corresponding upsampling and concatenation in the decoder. This manually configured architecture lacks the adaptive optimization and advanced data handling mechanisms found in nnUNet [2], which I believe is a major factor behind the performance gap.

Advanced Training Techniques: In my training pipeline, I incorporated several advanced techniques to enhance model performance and training efficiency. I leveraged mixed-precision training using PyTorch's AMP framework with a gradient scaler, which not only accelerated computations but also reduced memory usage. This approach, combined with a learning rate scheduler (ReduceLROnPlateau) and robust checkpointing, allowed my model to adapt dynamically during training despite being limited to 25 epochs—significantly fewer than the 50 epochs used by state-of-the-art methods like nnUNet. Additionally, real-time logging through Weights & Biases enabled me to monitor training metrics continuously and preserve the best model state, ensuring that my training process was both efficient and resilient to fluctuations in loss and accuracy.

Data Preprocessing and Augmentation: Data preprocessing and augmentation were pivotal in boosting my model's generalization capability on the limited medical imaging data available. I implemented a custom data loader that extracts axial slices from 3D volumes, normalizes each slice consistently using a dedicated normalization function, and applies a series of on-the-fly random augmentations—including horizontal flipping, rotation, and scaling. These augmentation strategies increased the diversity of my training dataset, mitigated the challenges posed by limited annotated data, and provided the model with varied examples to learn from. Furthermore, by systematically splitting the dataset into training, validation, and testing sets, I ensured that my evaluation metrics were unbiased and reflective of real-world performance, ultimately contributing to a more robust segmentation framework.

II. Observations and Results

I assessed performance using the Dice coefficient (or pseudo Dice, as reported in the nnUNet logs), which quantifies the volumetric overlap between the predicted segmentation and the ground truth. Preprocessing steps such as intensity normalization and random geometric augmentations were critical for stabilizing training and enhancing model generalization.

Table I summarizes the average and median Dice scores computed from my training logs. My custom UNet2D achieved an average validation Dice of approximately 0.387 and a median of 0.439, whereas the nnUNet attained an average Dice of about 0.932 and a median of 0.933. This corresponds to an absolute performance gap of roughly 0.545 (or 54.5 percentage points) in Dice coefficient between the two models.

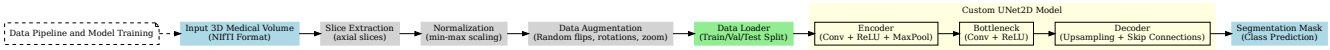


Fig. 1: Model Architecture and Data Preprocessing Pipeline.

TABLE I: Comparison of Dice Performance between My UNet2D and nnUNet

Metric	My UNet2D	nnUNet	Difference
Average Dice	0.387	0.932	0.545
Median Dice	0.439	0.933	0.494

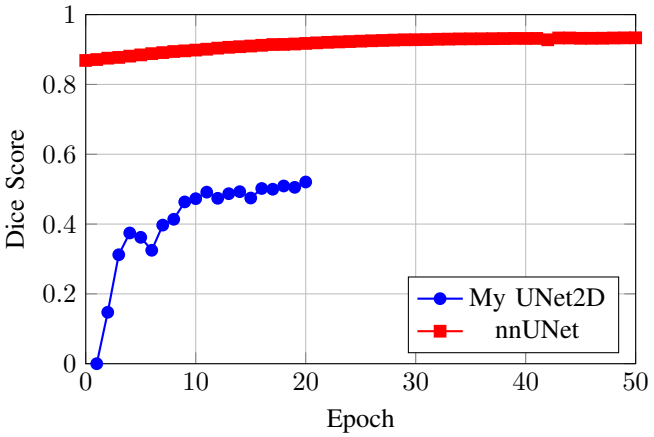


Fig. 2: Dice score progression over epochs for my UNet2D (blue, epochs 1–20) and nnUNet (red, epochs 0–50).

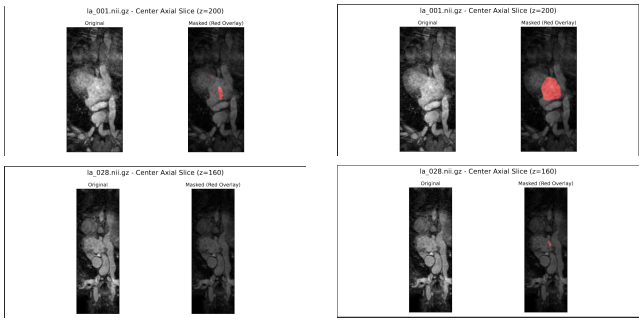
Figure 2 shows the progression of the Dice score over epochs for both models. The blue curve represents my custom UNet2D (epochs 1 to 20), while the red curve shows the nnUNet performance over epochs 0 to 50.

III. Comparison of Segmentation Results

To further evaluate the differences between the two models, I compared the segmentation masks produced by my UNet2D and the nnUNet. As shown in Figure 3, the nnUNet's segmentation masks are significantly better; the masks align accurately with the underlying anatomical structures and delineate the regions of interest with high precision. In contrast, the masks produced by my UNet2D exhibit noticeable misalignments and less precise boundaries. This qualitative comparison corroborates the quantitative results, highlighting the advantages of the self-configuring nnUNet framework over my custom approach.

IV. Conclusion

In summary, while my custom UNet2D model demonstrates the feasibility of a manually engineered 2D segmentation network, the adaptive nnUNet framework achieves superior performance due to its dynamic configuration, robust augmentation, and longer training regime. The Dice coefficient—a critical performance metric for segmentation tasks—underscores the importance of these advanced strategies. In future work,



(a) My UNet2D (b) nnUNet

Fig. 3: Comparison of segmentation results: (a) shows the output from my UNet2D with noticeable misalignment of the segmentation mask, while (b) displays the nnUNet result where the segmentation mask fits accurately over the image.

I plan to integrate adaptive preprocessing techniques [4] to further narrow this performance gap.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Proc. MICCAI*, 2015, pp. 234–241.

[2] F. Isensee, P. F. Jaeger, S. A. Kohl, J. Petersen, and K.-H. Maier-Hein, "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation," *Nat. Methods*, vol. 18, no. 2, pp. 203–211, 2021.

[3] A. L. Simpson *et al.*, "Medical Segmentation Decathlon," *arXiv preprint arXiv:1904.09096*, 2019.

[4] F. Garcea, A. Serra, F. Lamberti, and L. Morra, "Data augmentation for medical imaging: A systematic literature review," *Computers in Biology and Medicine*, vol. 152, p. 106391, 2023.