3D Medical Image Segmentation of the Heart: A Comparative Analysis of Augmentation Techniques

I. Introduction

3D medical image segmentation supports essential clinical tasks such as diagnosis, surgical planning, and treatment monitoring. Although 3D U-Net architectures have shown strong performance for volumetric segmentation, their generalization is often limited by the small size and uniform nature of medical imaging datasets. Data augmentation is therefore critical for increasing variability during training and reducing overfitting. To overcome this limitation, this project employs different augmentation strategies to investigate their impact. Both baseline and augmented models are trained and evaluated.

The main objectives of this project are as follows:

1 Implement two distinct 3D augmentation pipelines: a geometric–intensity pipeline applying flipping, 3D rotation, translation, and intensity jitter, and an elastic deformation method introducing non-linear anatomical variations.
2 Integrate both augmentation methods into a standardized 3D U-Net for fair and controlled experimentation.
3 Compare baseline and augmented models using Dice score, Hausdorff distance, sensitivity, and precision to quantify the effect of augmentation.
4 Visualize results, including augmented images, training curves, and predicted segmentations, to support qualitative analysis.

II. Material & Methods

This section provides a concise explanation of the dataset used, preprocessing, augmentation techniques applied, the 3D model and experimental settings, and the metrics used to evaluate the model.

A. Dataset

The dataset consists of 3D MRI heart images in `.nii.gz` format, along with their corresponding segmentation masks. It contains 17 training images and 3 testing images. The training images have shapes ranging from $[320, 320, 90]$ to $[320, 320, 130]$, while the testing images range from $[320, 320, 109]$ to $[320, 320, 120]$. The dataset provides high-resolution volumetric images suitable for 3D heart segmentation tasks.

B. Preprocessing

Preprocessing is an essential step in preparing the dataset for training. Although extracting patches is a common approach in 3D medical image segmentation, in this study, the entire 3D volumes were used due to the small dataset size, which is important to preserve the global anatomical context of the heart structures. Using whole volumes ensures that the spatial relationships between different cardiac regions are maintained, which can be critical for accurate segmentation. Since the images were not uniform in shape, they were first resized to a fixed target shape of $[224, 224, 96]$. This downsampling reduces memory and computational requirements during training, allowing the full 3D volumes to fit into GPU memory. All images and their corresponding segmentation masks were loaded using SimpleITK. Each image-mask pair was then resampled, applying linear interpolation for the images and nearest-neighbor interpolation for the masks to preserve label accuracy. Image intensities were normalized by computing the mean and standard deviation over non-zero voxels, followed by standardization and clipping to the range $[-6, 6]$. Segmentation masks were converted to binary masks, where all foreground labels were set to 1. Finally, a channel dimension was added to both images and masks to match the input requirements of the 3D U-Net model.

C. Data Augmentation

To increase variability and improve generalization, two types of augmentations were applied: affine transformations with intensity jitter, and elastic deformations. Affine transformations included random flips, small rotations, translations, and intensity variations. Elastic deformations applied smooth, random displacements to mimic realistic anatomical variations in the heart. The hyperparameters used for the augmentations are summarized in Table I.

TABLE I
Hyperparameters for data augmentation

Technique	Parameter	Value
Affine transformations	Maximum rotation angle	$\pm 10^\circ$
	Maximum translation	± 10 voxels
	Flip probability	0.5
	Intensity jitter	0.1
Elastic deformation	Scaling factor	15
	Smoothing	3

D. Model Architecture

For volumetric heart segmentation, a 3D U-Net architecture was employed. The network consists of an encoder-decoder structure with skip connections that preserve spatial information between corresponding levels. Each convolutional block contains two 3D convolution layers followed by layer normalization and ReLU activation. Downsampling is performed using 3D max-pooling, while upsampling uses 3D

transposed convolutions. The number of filters doubles at each downsampling step, starting from 16 filters, and the bottleneck contains 256 filters. The final output layer uses a $1 \times 1 \times 1$ convolution with sigmoid activation to generate voxel-wise probability maps. This architecture balances computational efficiency with sufficient capacity to capture complex 3D cardiac structures.

E. Performance Metrics

The performance of the model was evaluated using multiple metrics. The **Dice coefficient** measures the overlap between predicted and ground-truth masks, capturing both false positives and false negatives. **Sensitivity** quantifies the proportion of correctly identified foreground voxels, while **Precision** measures the proportion of predicted foreground voxels that are correct. Additionally, the **Hausdorff Distance** evaluates the maximum surface distance between predicted and true segmentations, highlighting boundary accuracy.

III. Results and Discussion

The hyperparameters used to train the model are summarized as follows: batch size = 2, number of epochs = 150, random seed = 42, optimizer = Adam with a learning rate of 1×10^{-4}, and loss function = binary cross-entropy combined with Dice loss.

A. Performance Evaluation

Experiments were conducted both with and without data augmentation. First, the model was trained without augmentation, and the corresponding training loss and Dice coefficient are shown in Fig. 1. Next, the model was trained using affine transformations with intensity jitter, and the results are shown in Fig. 2. Finally, the model was trained with elastic deformations, and the results are shown in Fig. 3.

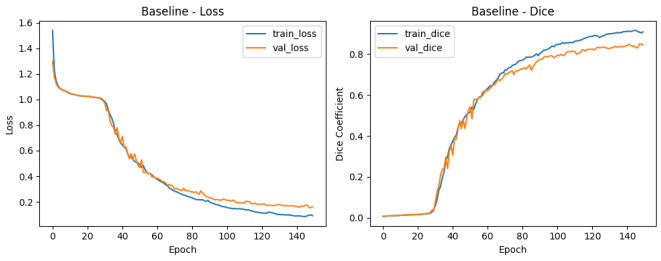

Fig. 1. Training loss and Dice coefficient of the base model without augmentation.

In Table II, the results of each experiment are presented for individual test samples as well as the mean across the test set. For individual patients, some experiments performed best for certain metrics, while others performed better on different patients. When considering the mean results across all test samples, the model trained with affine transformations and intensity jitter achieved the highest performance.

Table III shows the percentage improvement of the segmentation metrics achieved by the augmentation techniques

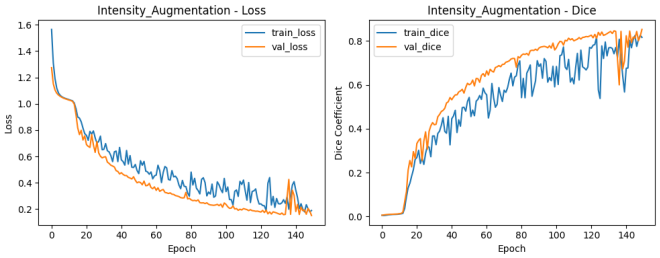

Fig. 2. Training loss and Dice coefficient with affine transformations and intensity jitter.

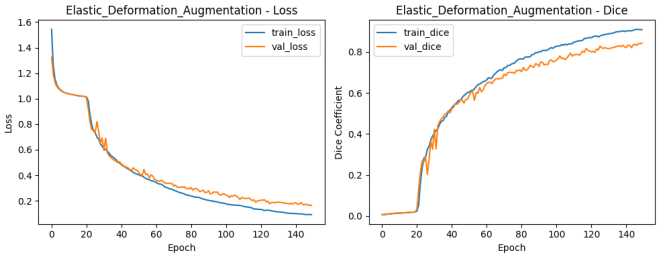

Fig. 3. Training loss and Dice coefficient with elastic deformations.

TABLE II
Comparison of Metrics Across Augmentation Experiments

Case	Experiment	Dice ↑	Precision ↑	Sensitivity ↑	Hausdorff ↓
Individual Test Patient Results					
018	Baseline	0.9064	0.8408	**0.9831**	2.8284
019	Baseline	0.8740	0.8266	0.9271	2.4495
020	Baseline	0.8964	0.8831	0.9102	**2.0000**
018	Affine-Intensity	0.9202	**0.9041**	0.9370	**2.0000**
019	Affine-Intensity	**0.9154**	**0.8827**	**0.9507**	**1.7321**
020	Affine-Intensity	**0.9342**	0.9045	**0.9659**	**1.4142**
018	Elastic	**0.9261**	0.8930	**0.9619**	2.2361
019	Elastic	0.8788	0.8650	0.8931	2.2361
020	Elastic	0.8977	**0.9219**	0.8747	2.2361
Mean Results Across Test Set					
Mean	Baseline	0.8923	0.8502	0.9401	2.4260
Mean	Affine-Intensity	**0.9233**	**0.8971**	**0.9512**	**1.7154**
Mean	Elastic	0.9009	0.8933	0.9099	2.2361

compared to the baseline. Among the augmentations, Affine-Intensity demonstrates the greatest overall improvement. Elastic augmentation also improves most metrics compared to the baseline, however, it shows a slight decrease of -3.21% in Sensitivity. Overall, the results indicate that intensity-based affine augmentation provides the most consistent and substantial improvement across all metrics.

TABLE III
Improvement of Metrics Achieved by Augmentation Techniques Compared to Baseline

Experiment	**Dice ↑**	**Precision ↑**	**Sensitivity ↑**	**Hausdorff ↓**
Affine-Intensity	**+3.47%**	**+5.52%**	**+1.18%**	**+29.29%**
Elastic	+0.96%	+5.07%	-3.21%	+7.83%

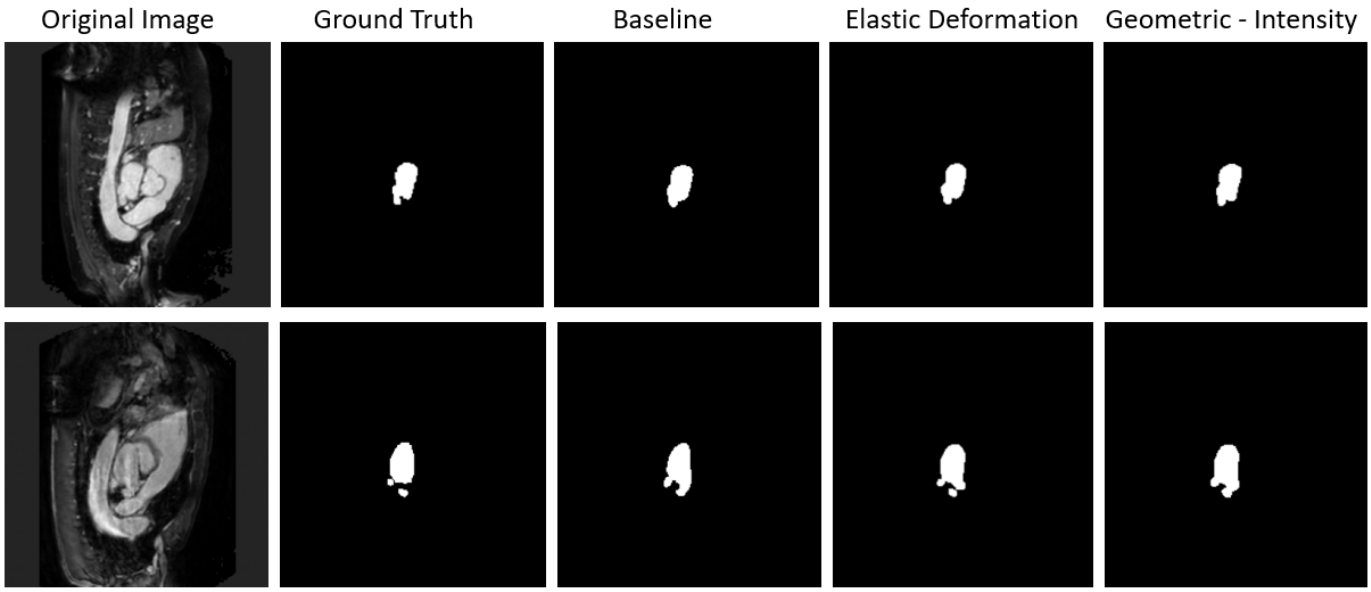

Fig. 4. Comparison of segmentation predictions: original images, ground truth, and predictions from baseline, affine-intensity, and elastic deformation experiments.

B. Visual Explanation

To visualize the predictions made by each model, the figure 4 shows the original images, ground truth masks, and predictions from each experiment. It can be clearly seen that in the first image, the model trained with affine-intensity augmentation produces predictions closer to the ground truth, while in the second image, elastic deformation augmentation performs best. In contrast, the baseline model provides poorer predictions. These visualizations highlight the effectiveness of training with augmentation in improving segmentation accuracy.

IV. Conclusion

This study presents a comparative analysis of the impact of augmentation techniques on 3D MRI heart segmentation. The dataset consisted of 17 training and 3 testing images. Two augmentation strategies—affine transformations with intensity jitter and elastic deformations—were applied, and the 3D U-Net model was trained using the Adam optimizer for 150 epochs. The best model was selected based on validation performance. Results indicate that affine-intensity augmentation achieved the greatest overall improvement, with Dice = **+3.47%**, Precision = **+5.52%**, Sensitivity = **+1.18%**, and a 29.29% reduction in Hausdorff distance compared to the baseline. Although these augmentation methods improved segmentation performance, there are limitations. Using full 3D volumes instead of patch extraction may not be suitable for larger datasets due to memory constraints. Future work could explore combining multiple augmentation techniques to further enhance performance. Additionally, employing transformer-based encoders or other advanced architectures may improve segmentation results and generalization on larger datasets.

Code Availability

The complete code for this project is available on GitHub at: https://github.com/daniyalbinasif/3D_segmentation. The repository contains all scripts for data loading, preprocessing, augmentation, model training, and evaluation. Additional results generated during the experiments can be found in the `Results` folder within the repository.