Master Project

Comparison of UNet- and Transformer-based Architectures for Segmenting 3D Electron Tomography Volumes

Vasudha Venkatesan

Examiner: Prof. Dr. Thomas Brox
Advisers: Yassine Marrakchi, Joachim Greiner

University of Freiburg
Faculty of Engineering
Department of Computer Science

January 07$^{\text{th}}$, 2023

Abstract

Segmentation of tubules in cardiomyocyte cells is an important task in medical imaging. Tubules are units of the heart that are required to maintain the calcium levels for contraction and relaxation of the contractile elements, sarcomeres. Accurate segmentation of these structures is crucial for understanding the underlying mechanisms of cardiac diseases such as heart failure and arrhythmias. CNN-based approaches such as UNet have been widely used for semantic segmentation in the medical field and has provided promising results. Transformers, one of the recent deep learning architectures, has achieved state-of-the-art benchmarks on several natural language processing and vision tasks such as machine translation, object detection, segmentation, image classification, and action recognition. The UNETR (UNet + Transformers) method combines a vision transformer as an encoder to extract features or representations of the input image, with a CNN-based decoder via skip connections to predict output segmentation. Another CNN-based method, nnUNet (no-new-UNet), developed by Isensee et al, is specifically designed for medical images based on UNet architecture. It automatically configures the hyperparameters such as preprocessing methods, network architecture, training, and post-processing based on the image. In this project, we have compared the segmentation performance of these three different methods - UNet, UNETR and nnUNet, on cardiomyocte cell images obtained from electron tomography. Based on the experiments conducted, we observe that transformer based architecture UNETR has produced the highest dice score of 95.081 compared to standard UNet architecture (92.326) or nnUNet (94.833) with fewer computational resources.

Contents

1 Introduction

Convolutional neural networks (CNN) have provided a breakthrough in the field of computer vision with numerous applications such as semantic segmentation, object detection and recognition, reconstruction of depth from images[1][2][3][4]. In medical imaging, volumetric segmentation of organs or micro structures from 3D medical images is a challenging task since manual annotation of each 2D slice is laborious and time-consuming. It is also repetitive as the neighboring slice has an almost similar segmentation mask compared to the previous 2D slice. UNet utilizes an encoder-decoder architecture of CNN and has provided promising results for segmentation[5]. The encoder block in UNet downsamples the input image at different resolutions by extracting relevant features, and the decoder block reconstructs the image by upsampling the extracted features and concatenating them with skip connections from the encoder. The concatenation at each step helps preserve the spatial dependency lost during the downsampling. This results in a U-shaped architecture. Since biomedical image tasks have less training data, data augmentation techniques have helped in dealing with data invariance. UNet has won the ISBI cell tracking challenge with a large margin on two challenging 2D transmitted light datasets and is used extensively in biomedical image segmentation[6].

Self-attention-based architectures have gained a lot of attraction in recent years, and transformers have become the model of choice for natural language processing tasks. Transformers utilize self-attention mechanisms to draw global dependencies between the input and output. The idea behind self-attention mechanism is that they help the decoder utilize the most relevant parts of the input using a weighted

sequence of input vectors, where more relevant sequences have higher weights[7]. The transformer architecture can allow for parallelization, reducing the amount of training time and substantially improving performance at the same time. UNETR follows a similar architecture as UNet where the encoder part is replaced with a transformer encoder and is connected to a CNN based decoder via skip connections at different resolutions[8]. The transformer encoder learns global dependency from volumetric input image sequences while maintaining the encoder-decoder architecture. UNETR has provided excellent results on 3D CT and MRI tumor data and has achieved state-of-the-art performance on MSD dataset[8].

With an increasing number of publications in the field of medical image segmentation, algorithms were developed with similar architectures to UNet with minor changes based on their dataset. Researchers found it difficult to identify the best architecture, beyond the limited scenario in which they were trained. nnUNet was devised to address this issue by developing a segmentation algorithm that generalizes to 10 medical datasets corresponding to different parts of the human body. nnUNet, is an end-to-end self-configuring framework based on 2D and 3D UNet architecture and configures the entire training model including preprocessing, training loss, optimizer selection, or data augmentation most suitable for the corresponding dataset[9]. It produced state of the art results with highest dice scores across all classes in the medical decathlon challenge. In this project, we have implemented these three methods to determine the best architecture suitable for our dataset.

2 Related Work

UNet has proven to be a highly effective architecture for medical image segmentation tasks. It has been widely used in various applications, including organ, lesion, cell, and tissue segmentation, image registration, and data augmentation. For example, for liver and lesion segmentation, RMS-UNet or the Residual Multi-Scale UNet with dilated convolutions has provided the highest dice scores with best lesion segmentation for four publicly available datasets[10]. In the case of multi-organ segmentation, 3D attention-based UNet architecture is used where a similar CNN-based encoder-decoder is utilised[11]. Attention gates in the decoder part of the UNet are responsible for enhancing the relevant features and suppressing the features that are less relevant to the segmentation map. This model works very well for also 3D pancreas multi-class image segmentation[12]. Another interesting paper by Rui et al. performs multi-organ segmentation by co-training weight-averaged models trained from much fewer organ datasets[13]. This model achieves an average of 90.22 dice score for multi-organ segmentation with 30% lower computational cost.

Tubule UNet was particularly developed for segmenting tubule-like structures in breast cancer cells[14]. The authors of this paper, Eren et al. propose three architectures for the encoder - ResNet34, efficientNet-B3, and Dense-Net161 and a similar architecture as UNet using CNN, for the decoder. EfficientNet-B3 with reflection padding implementation performed the best among these three architectures with a 95.33 dice score.

Multiple architectures using vision transformers as encoders or decoders or both have also been implemented for volumetric segmentation. UNetFormer utilizes 3D

SwinTransformer as an encoder and CNN as a decoder in the UNet architecture. It is also pretrained on publicly available CT datasets and produces state-of-the-art results on liver and tumor datasets[15]. With so many architectures available for medical image segmentation, we chose to implement a vanilla UNet, UNETR that is specifically designed for volumetric segmentation of medical images, and nnUNet which is an auto-configuring deep learning framework developed for biomedical image segmentation and compare their performances.

3 Background

3.1 3D UNet

UNet consists of an encoder-decoder structure where the encoder downsamples the input image to extract the features and the decoder upsamples the encoded features to obtain the original resolution of the image. An encoder block in the contracting path consists of repeated $3 \times 3 \times 3$ convolutions each followed by batch normalization and ReLu, followed by a $2 \times 2 \times 2$ max pooling layer for downsampling. After each max pooling, the number of features is doubled and the dimension of image is halved. This block is repeated for four times and is then connected to the decoder block. Similarly, in the expanding path, each decoder block consists of $2 \times 2 \times 2$ upsampling followed by a repeated $3 \times 3 \times 3$ convolution each consisting of batch normalization and ReLu. The upsampling operation reduces the number of feature channels by half and upsamples the feature block. Finally, a $1 \times 1 \times 1$ convolution is performed which reduces the number of output channels to the required number of output labels. At each upsampling step, skip connections from encoder are concatenated to the upsampled tensor and help to restore spatial structures lost during max pooling.

We have used padded convolution to get the same output size as the input. In [6], zero-padded convolutions are used to apply the overlap tiling strategy. This strategy is used to predict pixels at the border where missing context around the border is extrapolated by mirroring the input image. This can be applicable to neuronal structure data structures, but in our dataset, the required t-tubules (label 1) are solid structures and is much less compared to the background (label 0).

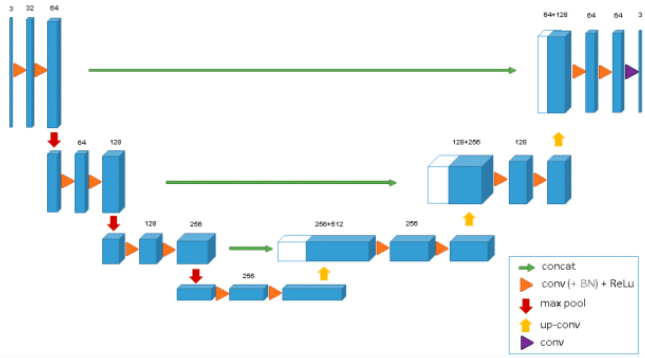


Figure 1: Model of 3D UNet architecture[5]. Here the number of features is represented in blue box and skip connections are represented using green arrows. The size of the image at various points are denoted above the box.

3.2 nnUNet

To auto-configure the network model and hyperparameters based on the dataset, nnUNet classifies the design choices into three types of parameters as shown in Figure 2.

1. Fixed parameters
2. Rule based parameters
3. Empirical parameters

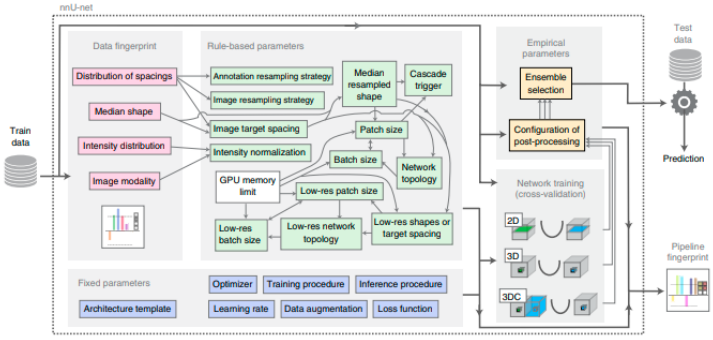


Figure 2: Model of nnUNet with various parameters differentiated[16]. The three types of parameters are fixed parameters, rule based parameters and empirical parameters. Fixed parameters are constant values set based on previously identified robust configurations. Rule based parameters are set based on the dataset attributes such as image size or class ratio. Empirical parameters are the remaining parameters such as post processing or ensemble architecture.

Fixed parameters do not change depending on the dataset and its values are set based on previously identified robust configurations. Some of them would include number of epochs, loss function, and inference procedure. Rule based parameters are derived from 'dataset fingerprint' and 'pipeline fingerprint' during the plan and preprocess stage. Dataset fingerprint is a standard representation of the dataset, typically consisting of image size, class ratios, and voxel spacing information. Pipeline fingerprint consists of the values that are based on interdependent configuration of fixed parameters and dataset fingerprints. For example, patch size, batch size, and network topology are interdependent configurations, and choosing values for these parameters are modeled in the form of heuristic rules that allow for almost-instant execution upon application. A list of all heuristic rules and general guiding principles is provided in Supplementary Note 2 of [16]. Empirical parameters constitute the remaining design choices such as post-processing based on training data. The architecture followed by nnUNet for this dataset is shown in Figure 3.

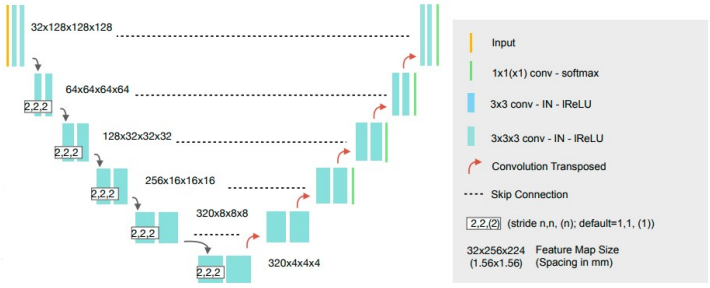


Figure 3: Architecture of the UNet created by the nnUNet algorithm. nnUNet creates a UNet of depth length 5 with the size of the feature map represented in the beginning.

nnUNet follows similar architecture compared to 2D or 3D UNet with certain modifications. Since large patch sizes produced better results, the authors Isensee et al., preferred using larger patch sizes from the image and reducing the batch size during training. Since batch normalization does not perform well on small batches, the authors also replaced batch normalization with instance normalization. They also replace the ReLu activations with leaky ReLu for negative slope values. nnUNet also attempts to create a deeper UNet compared to the original paper. The authors also suggest to perform downsampling until the feature maps are relatively small of the order $4 \times 4 \times 4$ to ensure sufficient context aggregation.

3.3 UNETR

In order to understand the architecture of UNETR, it is important for us to understand self attention architecture used by the transformers. The transformer utilizes multi head self attention layers in order to draw global dependencies between the input and the output. It also employs an encoder and decoder, but removing recurrence in favor of attention mechanisms allows for significantly more parallelization than methods like RNNs and CNNs[7].

3.3.1 Self attention

At its core, self-attention is a weighted sum of all its input vectors. The weight for each input is a derived value computed from all the inputs. A softmax operation is applied to these weights to ensure they are positive and sum to 1.

$$W = softmax(X^T X) \tag{1}$$

Every input in a self-attention layer occurs in three different positions : value - the input vector that is used to compute the weighted sum Y, query - the input vector matched against which every other input vector, key - the vector against which each query is matched. Self-attention is an attention layer where all the keys, queries, and values are obtained from the same set. In order to make self-attention more flexible, linear transformations can be applied to these vectors to differentiate each role as shown in equation 2. Multi-head attention is used to model different relationships among different inputs in a network[17]. This is done by applying self attention operation in parallel. There are multiple query, key and value matrices which are initialised randomly. After training, each head is used to represent input in a different representation space.

$$\begin{aligned} k_i &= Kx_i + b_k \\ q_i &= Qx_i + b_q \\ v_i &= Vx_i + b_v \end{aligned} \tag{2}$$

The **K,Q** and **V** in the equation (2) represent key, query and value matrices that represent the linear transformation to differentiate each role. b_k, b_q, b_v are the corresponding bias vectors for key, query and value. k_i, q_i, v_i is the corresponding key, query and value for the input x_i.

3.3.2 Vision transformers

In order to keep the implementation simple and efficient, the authors of vision transformers, Alexey et al. decided to maintain the architecture of vision transformers

similar to NLP transformers. The input image x $\in R^{H\times W\times D\times C}$ is divided into 1D sequence of non overlapping patches of $x_v \in R^{N\times(P^3\cdot C)}$ with resolution $(P \times P \times P)$ and number of patches, $N = (H \times W \times D)/P^3$.

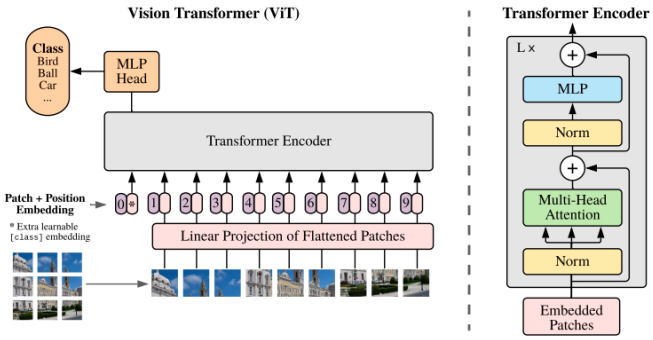


Figure 4: Model overview of Vision Transformer (ViT)[18]. Here, the input is converted into non over lapping patches and is projected using linear transformer into an embedding space. The sequence is added with a position embedding and given to a transformer model. The transformer block consists of multi head self attention that learns input features in different embedding spaces.

The output from encoder can be utlized in various ways based on the application.

A linear projection is applied to these patches to project the input in a K-dimensional space for the transformer architecture. In order to maintain the spatial information, a learnable positional encoding is applied to the linear projection. The output from the embedding layer is given to a stack of transformer blocks and is used as an encoder. Each transformer block consists of MSA (multi-head self attention) and MLP (multi-layer perceptron) sublayer. MSA is responsible for the model to learn how each input patch is related to the other input patches. The output from MSA is fed into the same MLP layer individually. Since these feed forward operations are independent, they can be executed in parallel. Each sublayer is followed by a residual connection and layer normalization to prevent vanishing gradient problems.

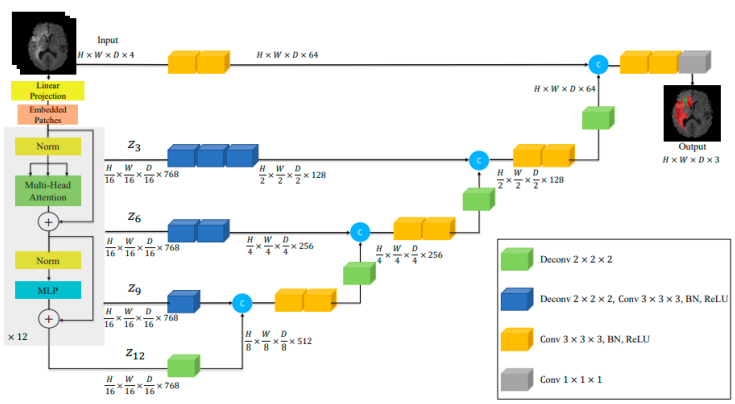


Figure 5: Model of 3D UNETR architecture[8]. Our model consists of a transformer encoder that directly utilizes 3D patches and is connected to a CNN-based decoder via skip connection.

In UNETR, encoder extracts features or sequence representation from multiple resolutions of the size $\frac{H \times W \times D}{P^3} \times K$ from the input image, where K is the dimension of embedding space. Since the output from the transformers is in embedding space, it is reshaped into input space by passing it through a block consisting of a $2 \times 2 \times 2$ deconvolutional layer followed by a $3 \times 3 \times 3$ convolutional layer, batch normalization, and ReLu activation. The number of blocks required for reshaping the output from the transformer depends on the image resolution at that part of the UNet architecture. At the bottleneck of the encoder, a $2 \times 2 \times 2$ deconvolution is applied and is concatenated with the output from the previous layer, similar to the skip connections in UNet architecture. The output is passed through consecutive $3 \times 3 \times 3$ convolutional layers and upsampled using a deconvolutional layer. This is repeated until the output reaches the original input resolution and a final $1 \times 1 \times 1$ convolutional layer with softmax activation to generate pixelwise segmentation result.

4 Methods

4.1 Dataset

The input images are in HDF5 format with an approximate size of $800 \times 800 \times 140$ voxels. The total training dataset consists of 201 images and validation dataset consists of 50 images. These images were chosen from separate image slices such that no two images from similar slice were present in training, validation set and test set, to prevent data leakage. The test data set is separate and consists of 70 images that are obtained directly from electron tomography images of the cardiomyocyte cells.

4.2 Data Loading and Augmentation

Since the input images are fairly large, we implemented lazy loading in the data loader. HDF5 files also support chunkwise loading of the image. As we use only patches of the image to train the model, loading the particular chunks instead of the entire image greatly improved the computational efficiency of GPUs. Random 3D cropping and normalization were the only data augmentation methods applied to input data. The above mentioned techniques were implemented for UNet and UNETR. For nnUNet, a number of data augmentation techniques are applied stochastically based on a predefined probability obtained from a uniform distribution. The set of data augmentations available in nnUNet are: scaling, rotation, addition of gaussian noise, gaussian blur, brightness, contrast, gamma augmentation, mirroring and simulation of low resolution where each patch is downsampled to half its size using

nearest neighbour interpolation and then sampled back to original resolution [9].

4.3 Loss function

We used the same loss function to train both UNet and UNETR as used by the original [8] paper. The loss function is a weighted sum of dice loss and cross-entropy loss, implemented by Monai. It is given by the equation 3.

$$L(G,Y) = \lambda_{dice} \sum_{j=1}^{J} \frac{\sum_{i=1}^{I} G_{i,j} Y_{i,j}}{\sum_{i=1}^{I} G_{i,j}^2 + \sum_{i=1}^{I} Y_{i,j}^2} + \lambda_{ce} \sum_{i=1}^{I} \sum_{j=1}^{J} G_{i,j} log Y_{i,j} \tag{3}$$

G signifies the one hot encoded ground truth value and **Y** is the probability output of the predictions made by the model. **J** is the total number of labels in the dataset and **I** is the total number of voxels in the input image. $Y_{i,j}$ denotes the probability of label **j** at position **i** in the image. λ_{dice} and λ_{ce} are the weights for corresponding dice score and cross entropy loss. The loss for nnUNet is similar but uses soft dice loss and so the loss is maximised during train. It is the simple average of soft dice loss and cross entropy loss without weights.

4.4 Training Scheme

Batch size was consistently maintained as 2 for all the three models. Other major hyperparameters chosen during training for these three different methods are given in Table 1.

	Optimizer	Epochs	Learning rate
UNet	Adam	400	$1e^{-5}$
UNETR	Adam	400	$1e^{-5}$
nnUNet	SGD	1000	0.01 (initial LR)*

Table 1: Values of hyperparameters used by three methods during training. nnUNet utilizes a polyLR scheme during training for learning rate.

nnUNet assigns learning rate using polyLR schedule where the learning rate is

multiplied by $(1 - \frac{iter}{maxiter})^{power}$. PolyLR has proven to be more effective than step learning rate when training. The power term controls the shape of learning rate decay and the value set to 0.9[19]. With lower learning rates such as $1e^{-2}$ or $1e^{-3}$ for UNet and UNETR, loss values after certain epochs produces NAN values. Thus a lower learning rate of $1e^{-5}$ was chosen for the learning rate. Medical images suffer from class imbalance, which lead to certain rare classes being ignored during training. All the three methods perform oversampling during training to ensure atleast one of the foreground classes is available during training. For training UNet and UNETR, mixed precision also was implemented in order to prevent out of memory issues. Mixed precision helps in increasing training speed up to three folds as it reduces memory access time by using only half the bytes compared to single precision. This enables us to train larger models or with larger batch sizes.

4.4.1 Dice score

To evaluate the three methods, dice score is used to compare their segmentation performance. The formula of dice score is given by equation 4. Here TP refers the number of true positives, FP refers to number of false positives and FN refers to number of false negatives. True positives refers to the same label in both ground truth and prediction, false positive refers to labels predicted positive in prediction but not present in ground truth labels and false negative are the missed out predictions from the ground truth.

$$dicescore = \frac{2 * TP}{2 * TP + FP + FN} \tag{4}$$

4.5 Inference

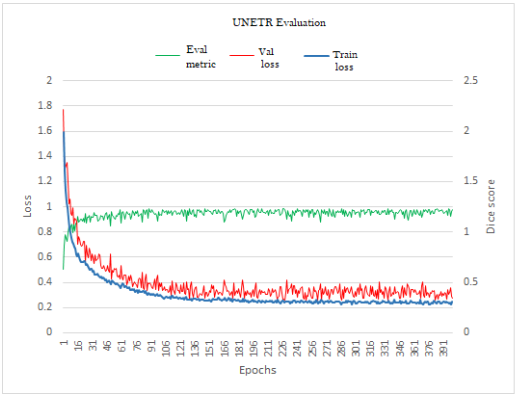
Inference procedure was implemented using sliding window inference by Monai. Patch based inference method is preferred to fully convolutional inference due to the memory constraints. Window size is set to the patch size during training with

an overlap of 0.5. This is consistent for all the three deep learning models. In nnUNet, cross validation of five sets was trained and in the end, for inference an ensemble of all the five models were used.

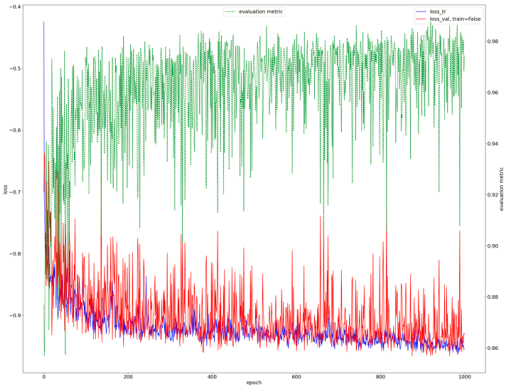
5 Analysis

5.1 Quantitative Analysis

The comparison of dice scores for three models is given in Table 2. The patch size in which UNet and UNETR was trained is $64 \times 256 \times 256$ voxels (depth x height x width). Since a larger patch size led to memory issues in UNet, we trained UNETR with a larger patch size of $64 \times 320 \times 320$ voxels. Increasing the patch size along the height and width dimension produces higher dice score, because it provides more context of the classes than along the depth dimension. The dice score obtained from UNETR with a patch size of $64 \times 320 \times 320$ voxels yielded the best dice score of 0.9508 comparable to the value obtained from nnUNet of 0.963 with a patch size of $40 \times 256 \times 240$ voxels. The training loss, validation loss and dice curves obtained during training of UNETR are shown in Figure 6. The training converges after 300 epochs with the highest validation dice of 0.983 for UNETR. Similarly, the training loss, validation loss and evaluation metric for nnUNet for cross validation set four is shown in the Figure 6.



(a) UNETR evaluation scores.



(b) nnUNet evaluation metrics.

Figure 6: These images denote the training metrics of models UNETR and nnUNet. The blue curve represents training loss, red curve represents validation loss and green curve represents the dice score

	Validation loss	Validation dice score	Test dice score
UNet	0.25871	0.98253	0.92326
UNETR	0.27615	0.97203	0.93199
UNETR (Larger patch)	0.27830	0.98465	0.95081
nnUNet		0.98010	0.94833

Table 2: Comparison of validation loss, validation dice score and average test dice score for three segmentation methods. Here the validation loss for nnUNet is not applicable since soft dice loss function is used for training, where the loss is maximised during training. Larger patch size is 64x320x320 voxels.

While comparing the five lowest dice scores obtained from UNETR and nnUNet model, three images were common with poor segmentation results for both models. The common error in these images was oversesgmentation.

5.2 Qualitative Analysis

On analysing individual dice score of the segmented images for model UNETR, we obtain a good understanding on the performance of segmentation on different input

images. First, we analyse the images with a dice score of 0.98 or above. Out of 56 test images, 8 images had a dice of 0.98 or above which resulted in almost perfect segmentation as shown in Figure 7. Again out of the 56 images, 44 images has a dice score of 0.93 or above. A few images with less than 0.85 dice score reduced the average dice score on test images.

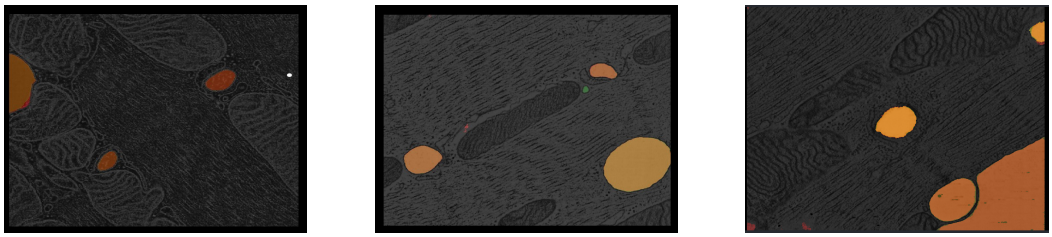


Figure 7: Best segmented images using UNETR with dice score above 0.99. Green color was used for ground truth values and red color was used for predictions. An overlap of green and red produced brown color, indicating almost accurate segmentation predictions.

The lowest was 0.7477 whose segmentation is shown in Figure 8 with other poorly segmented images. Figure 8 shows the segmentation of the same image at different slices. In the first image, the model has completely missed the segmentation of t-tubule and predicted the blob which looked similar to t-tubule but with a smoother surface and lesser texture such as small thread like structures inside the surface. After a few slices, it predicted the t-tubule, but not with proper boundary and further more slices, the image quality reduced and the model oversegmented at certain positions towards the corners. Oversegmentation at the end of the slices occurred in few more images where the quality of the image was poor at the end or beginning. Some examples of it is shown in Figure 9. Oversegmentation can be reduced by masking low quality regions with ignore mask label.

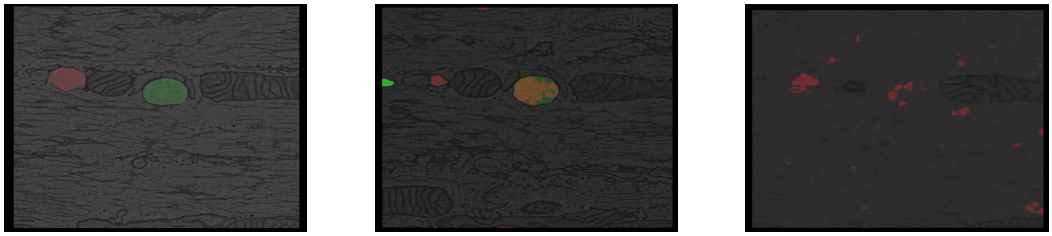


Figure 8: Poor segmentation of same image at different slices using UNETR with dice score of 0.74. Green color denotes ground truth and red denotes prediction, brown denotes the overlap of both the slices.

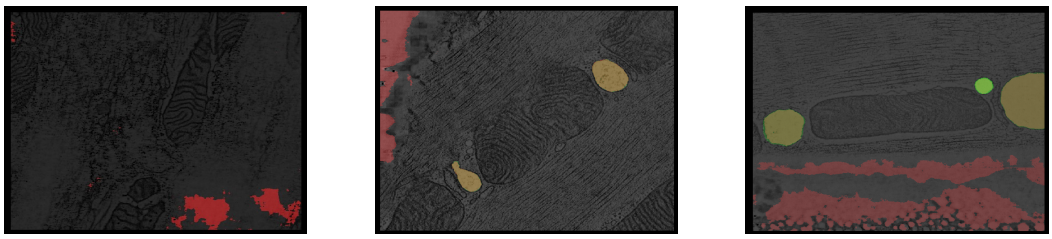


Figure 9: Oversegmentation at the borders where image quality is low with a dice score of 0.83.

6 Conclusion

In this project, we have compared the performance of CNN based approach and transformer based approach for volumetric segmentation of t-tubules in cardiomyocyte cells. We observe that, using a transformer based encoder and CNN based decoder, produces better quality segmentations compared to fully CNN based approach. We trained both the models until the losses converge and observe a higher dice score of 0.95081 for a single UNETR model and 0.94833 for an ensemble of five cross validated sets UNet models. A combination of transformer and CNN based models can used as a foundation for segmentation of other kinds biomedical images.

Bibliography

[1] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[2] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *International Conference on Learning Representations*, 2014.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[4] E. Ilg, T. Saikia, M. Keuper, and T. Brox, "Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation," *CoRR*, vol. abs/1808.01838, 2018.

[5] Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," *Medical Image Computing and Computer-Assisted Intervention – MICCAI. Lecture Notes in Computer Science(), vol 9901. Springer, Cham.*, 2016.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, 2015.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[8] A. Hatamizadeh, Y. Tang, V. Nath, D. Yang, A. Myronenko, B. Landman, H. Roth, and D. Xu, "Unetr: Transformers for 3d medical image segmentation," *arXiv*, 2021.

[9] F. Isensee, J. Petersen, A. Klein, D. Zimmerer, P. F. Jaeger, S. Kohl, J. Wasserthal, G. Koehler, T. Norajitra, S. Wirkert, and K. H. Maier-Hein, "nnu-net: Self-adapting framework for u-net-based medical image segmentation," *arXiv*, 2018.

[10] R. A. Khan, Y. Luo, and F.-X. Wu, "Rms-unet: Residual multi-scale unet for liver and lesion segmentation," *Artificial Intelligence in Medicine*, vol. 124, p. 102231, 2022.

[11] Liu Y, Lei Y, Fu Y, Wang T, Tang X, Jiang X, Curran WJ, Liu T, Patel P, and Yang X, "Ct-based multi-organ segmentation using a 3d self-attention u-net network for pancreatic radiotherapy," *Med Phys. 2020 Sep;47(9):4316-4324. doi: 10.1002/mp.14386. Epub 2020 Aug 2. PMID: 32654153; PMCID: PMC8278307*, 2020.

[12] O. Oktay, J. Schlemper, L. L. Folgoc, M. C. H. Lee, M. P. Heinrich, K. Misawa, K. Mori, S. G. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention u-net: Learning where to look for the pancreas," *CoRR*, vol. abs/1804.03999, 2018.

[13] R. Huang, Y. Zheng, Z. Hu, S. Zhang, and H. Li, "Multi-organ segmentation via co-training weight-averaged models from few-organ datasets," *CoRR*, vol. abs/2008.07149, 2020.

[14] E. Tekin, Çisem Yazıcı, H. Kusetogullari, F. Tokat, A. Yavariabdi, L. O. Iheme, S. Çayır, E. Bozaba, G. Solmaz, B. Darbaz, G. Özsoy, S. Ayaltı, C. K. Kayhan,

Ümit İnce, and B. Uzel, "Tubule-u-net: a novel dataset and deep learning-based tubule segmentation framework in whole slide images of breast cancer," *Nature methods*, 2023.

[15] A. Hatamizadeh, Z. Xu, D. Yang, W. Li, H. Roth, and D. Xu, "Unetformer: A unified vision transformer model and pre-training framework for 3d medical image segmentation," 2022.

[16] F. Isensee, P. F. Jaeger, S. A. A. Kohl, J. Petersen, and K. H. Maier-Hein, "nnu-net: a self-configuring method for deep learning-based biomedical image segmentation," *Nature Methods*, 2020.

[17] B. Peter, *Lecture 12.1 Self-attention from DLVU*. DLVU, 2020.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020.

[19] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *CoRR*, vol. abs/1606.00915, 2016.