Half-UNet: A Simplified U-Net Architecture for Medical Image Segmentation

Cs577 Deep Learning Project Report

Yash Khandelwal

Marut Pandya

11.15.2022
Fall 2022

Team Member Responsibilities

Yash Khandelwal

- Explored one of the three datasets mentioned in the Research Paper
- Worked on data preparation and partially contributed to data augmentation
- Implemented the Half-UNet model architecture
- Implemented the Nested Half-UNet model architecture
- Performed hyperparameter tuning on Nested Half-UNet model
- Model testing and evaluation for a couple of models

Marut Pandya

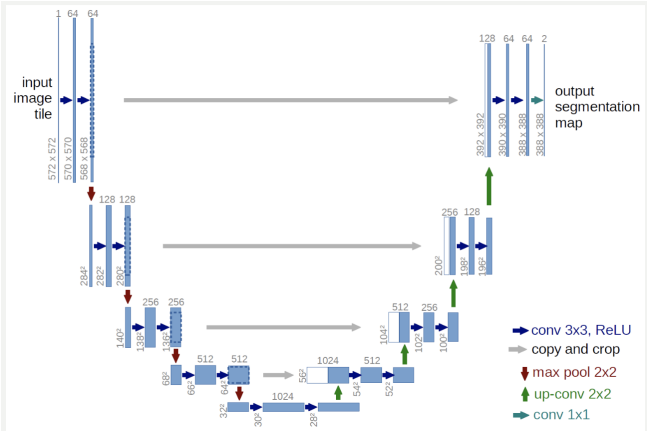
- Explored two of the three datasets mentioned in the Research Paper
- Worked on data augmentation
- Implemented the UNet model architecture
- Performed hyperparameter tuning on Half-UNet model as well as on UNet model
- Model testing and evaluation for a couple of models

Problem Statement

Medical image segmentation plays a vital role in computer-aided diagnosis procedures. The main goal of segmenting this data is to identify areas of the anatomy required for a particular study or medical diagnosis. U-Net is widely used in medical image segmentation. Many variants of UNet have been proposed, which attempt to improve the network performance while keeping the U-shaped structure unchanged. However, this U-shaped structure is not necessarily optimal. In this project, we attempt to analyze the segmentation ability of the existing UNet architecture and finally a more efficient architecture, Half-UNet and its variants are proposed. The proposed architectures are also encoder-decoder type architectures (similar to UNet) but in the Half-UNet architecture both the encoder block as well as the decoder block have been simplified. Furthermore we propose another architecture based on Half-UNet architecture as well as the U^2-Net Architecture which in our testing gave even better results. We compared all these models on a single left ventricular MRI dataset and all the models looked equally efficient with some having very few trainable parameters.

Proposed Solution

Over the years many UNet variants have been proposed that are capable of giving better results than UNet in slightly fewer parameters and FLOPs, but one thing that hasn't changed is the U-shaped structure. The image shown below is the image of the base UNet model which has a typical UNet structure and a few skip connections going from encoder to decoder and filling in the lost information using high level features.



Recently, [5] conducted comparative experiments on Multiple-in-Multiple-out (MiMo), Single-in-Multiple-out (SiMo), Multiple-in-Single-out (MiSo), and Single-in- Single-out (SiSo) encoders. The experiments show that the SiMo encoder can achieve almost the same performance as the MiMo encoder (such as [2]). This result suggests that the benefits of using multi-scale feature fusion are far less than those of the divide-and-conquer strategy. To verify the same In the research paper an extensive analysis has been done on the architecture to

understand the main contributing factors in the success of UNet. The analysis was performed by removing either encoding (**B in Figure 1 below**) or the decoder (**A in Figure 1 below**) and replacing them with a simple downscaling or upscaling operation. After training several models the conclusion was that the benefits of using multi-scale feature fusion (decoder) are far less than those of the divide-and-conquer strategy(encoder). This means that the feature fusion side (decoder) can be further simplified and still be able to achieve the same results. Moreover, there are several simple steps that can be performed to simplify each convolution operation as well, which means we can simplify both the encoder as well as the decoder and still be able to achieve the same results. In the **Figure 1** below we can see 3 models, where A, B are only encoder and only decoder and C is the more
complete architecture closer to the UNet. The results of the experimentations are in Table **1,** where A performed very close to the performance of C and UNet whereas B due to missing encoder region failed to do so. Keeping this in mind and some more changes we come up with this new simplified architecture called Half-Unet.

TABLE 1 | Experimental results of different kinds of encoders.

Network model	Mammography	Lung nodule	Endocardium	Epicardium
Encoder (A)	0.8928	0.8867	0.8901	0.9328
Encoder (B)	0.8744	0.8803	0.7696	0.8158
Encoder (C)	0.8923	0.8878	0.8811	0.9280
U-Net	0.8939	0.8842	0.8797	0.9299

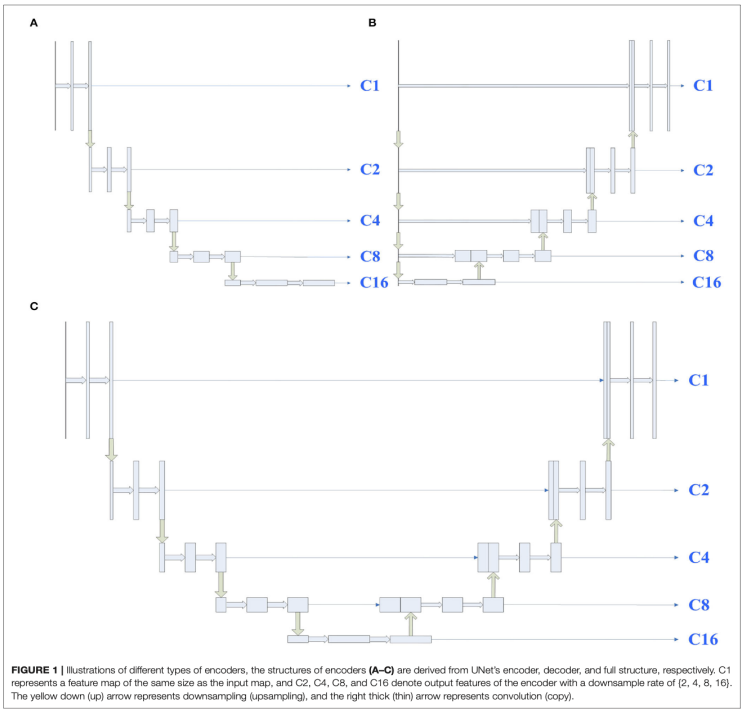


FIGURE 1 | Illustrations of different types of encoders, the structures of encoders **(A–C)** are derived from UNet's encoder, decoder, and full structure, respectively. C1 represents a feature map of the same size as the input map, and C2, C4, C8, and C16 denote output features of the encoder with a downsample rate of {2, 4, 8, 16}. The yellow down (up) arrow represents downsampling (upsampling), and the right thick (thin) arrow represents convolution (copy).

There are majorly three changes in the UNet model to get the Half-Unet. They are as follows :

- **Unification of channels :** In each downscaling step, we increase the number of channels in Unet which prevents information loss but at the same time increases the model complexity. Also, because the number of channels are different, additional convolutions are required to unify them which would increase the FLOPs as well as trainable parameters. To prevent this all channels have been unified, i.e. the same number of channels are used throughout all the layers.

- **Full Scale Feature Fusion :** In UNet architecture concatenation operations are used for feature fusion which again is done to let the model learn as much as possible but the problem is this increases the number FLOPs and in turn the model complexity. To cope up with this just like Resnet[6] architecture instead of concatenation, an additional operation is used which increases the amount of information under each channel without using any more parameters and also decreases the number of channels. Finally a full scale feature fusion is carried out with this addition operation.

- **Ghost Module :** Finally, the most important module which decreased the FLOPs by a significant amount is this ghost module which is a cheap operation being performed instead of a standard convolution. When we use normal convolution layers we are essentially multiplying the kernel with the image which includes its height, width and the depth (channels) which leads to a lot of floating point multiplications. Instead of that we only take half of the channels from convolutions and the other half by applying **depth wise separable convolutions** and they are finally concatenated. Because this operation is performed on each channel separately, we are able to save a lot of floating points operations. The image below shows exactly what happens inside the ghost module.

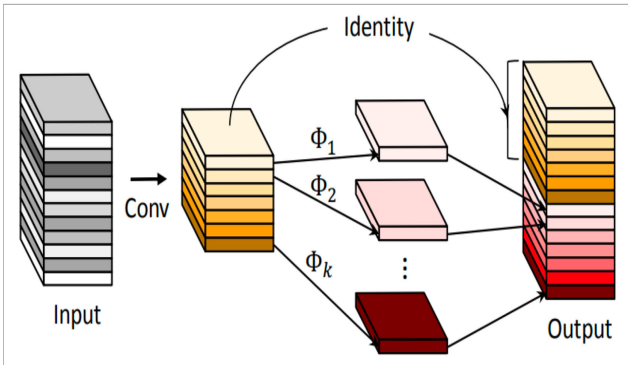


FIGURE 4 | An illustration of the convolutional layer and the introduced Ghost module for outputting the same number of feature maps. Φ represents the cheap operation.

All the above three changes combined give us the final architecture for the Half-Unet model which is shown below with unified channels, full scale feature fusion and use of a ghost module instead of a normal convolution.

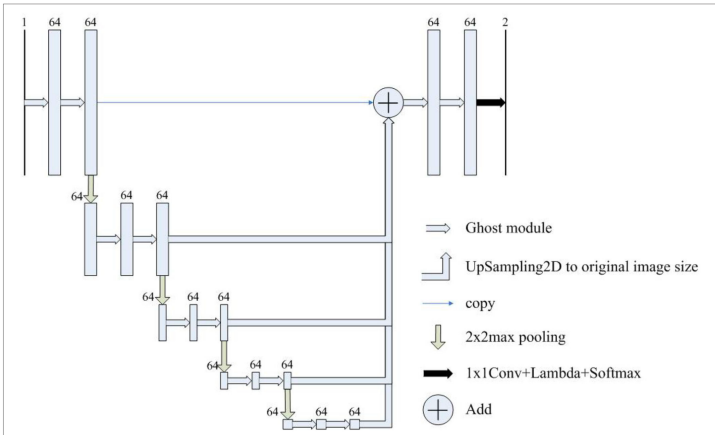
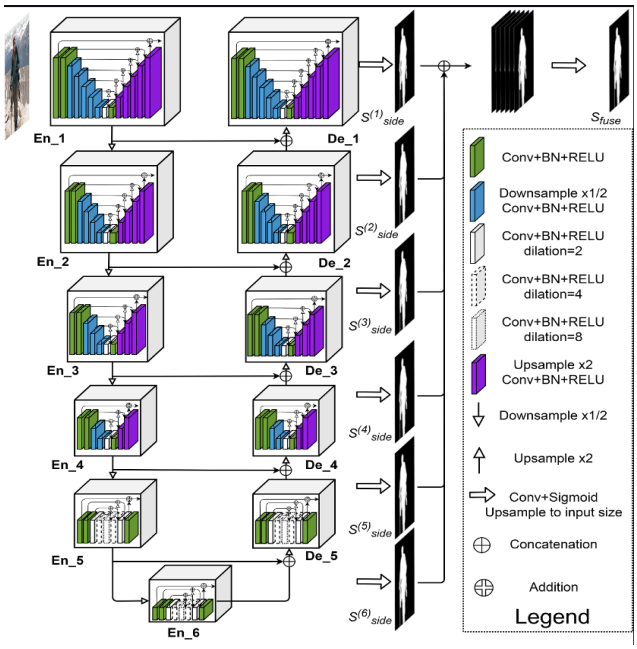


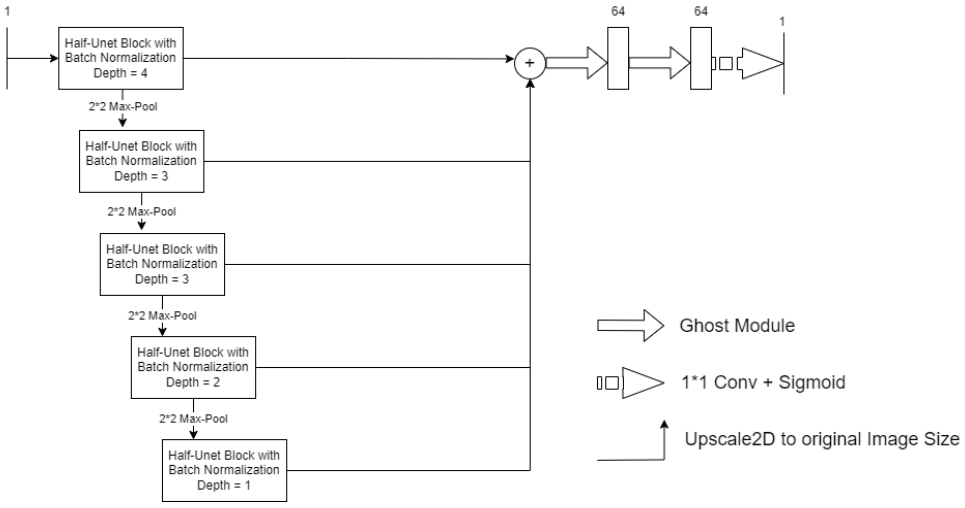
FIGURE 3 | The architecture of Half-UNet. The input image size is detailed in **Table 2**. The numbers above the rectangles represent the number of feature map channels.

We also experimented with this Half-UNet model further to improve its performance as well as versatility. The problem was this model might be too sophisticated for some complicated segmentation tasks so to utilize its benefits for more complex task as well, we tried to increase the model complexity by keeping the structure same and adding a multi scale feature extraction at each level of the Half-UNet model using multiple (nested) Half-UNet models with varying depths at each level. This idea was directly influenced by another UNet based research paper, i.e. the U^2-Net [4] which uses the concept of nested U-structure. Its design allows it to capture more contextual information from different scales thanks to the mixture of receptive fields of different sizes and it also increases the depth of the whole architecture without significantly increasing the computational cost. Using the same concept we tried the nested Half-UNet model for which the architecture can be seen in the below image.

U^2-Net



Nested Half-UNet

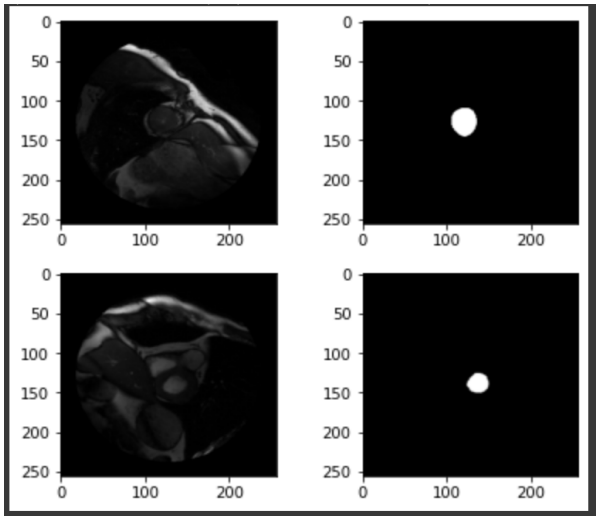


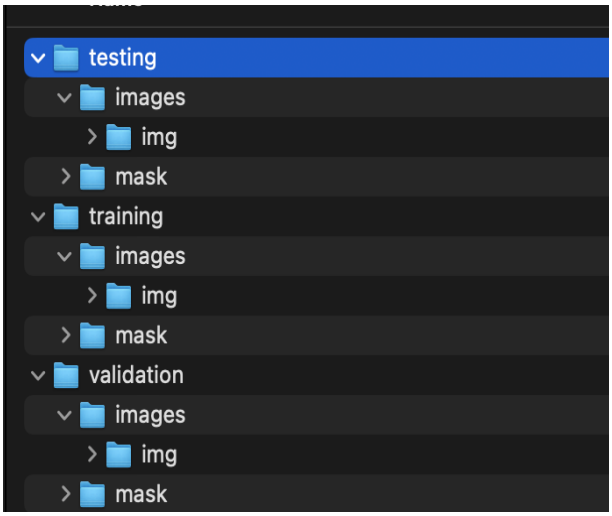
Implementation Details

Dataset Used : MICCAI Left Ventricular MRI Dataset [9]

Dataset Description : This dataset contains short-axis images of cardiac MRI scans from multiple cases. There are 45 cases in MICCAI 2009, divided into three groups, and each group contains 15 more cases belonging to different types of failures and the normal scenario. All of the left ventricular MRI cases have endocardium and some have epicardium, so because endocardium is in all of them in this project we only focus on **endocardium segmentation**. Among them, 30 cases are used as the training set and 15 cases are used as the test set. The training data came out to be around 550 while the testing was around 260. The data has two divisions, one with all the images in **dicom** format and the other is a file containing the **list of all the contours** for creating segmentation masks.

Dataset Preparation : Data preparation essentially has 3 steps, the first is to read and convert each image from dicom to png format. Second, we created masks from the contour list where in a numpy array all the background pixels are set to 0 while all the foreground pixels are set to 255. Which can be seen in the image below. Also the original structure of this dataset is too complicated and not apt for model training, so we created a directory structure from where we can read data using generators with ease. The image for the directory structure is also attached below.





Dataset Augmentation : To further diversify the data and get more samples to train on, we **augmented data** exactly as it was mentioned in the research paper. For each endocardium image we rotate it seven times 45 degrees iteratively and do a horizontal and vertical flip which takes the count to 10 images per origin image. Because we had to control the type of augmentation happening on each image, we used a custom function to do so and not the keras inbuilt one as that does all the augmentation on the go randomly. **Finally we end up with around 4900 training images, 500 validation images and 260 testing images.**

Data Preparation and Augmentation code file : **data_preparation.ipynb**

Implemented Model Architectures :These are the following models which have been implemented in **Keras** and further trained on the same data as well as with the same configuration to be able to do a fair comparison between them. The models are as follows :

- Half-UNet : Same as mentioned in [1]
- UNet : Same as mentioned in [3]
- Nested Half-UNet : Model inspired by U^2-Net [4]
- Half-UNet with batch normalization and another with regularization

Training Configuration and Details :
Epochs : 60
Input image size = (256, 256, 3)
Batch Size : 2
Learning rate : Initially set to 0.001 and switched to 0.0005 at epoch 30 and 0.0001 at epoch 50.
Optimizer : Adaptive moment estimation (Adam)
Loss Function : Binary cross entropy (also tried Dice Loss)
Convolution weight initializer (not for UNet): Kaiming initialization

Evaluation Metrics : Used multiple metrics during training to visualize the performance of the model. We used **dice coefficient, iou, precision, recall, f1 score, as well as accuracy**. The most reliable of them is the dice coefficient which punishes the wrong detections exactly how it's supposed to be done and the worst is the accuracy because of very high data imbalance, we have way more background pixels than the foreground pixels and therefore it gives high accuracy even though the model is barely performing.

Testing Instructions : Steps to be able to test all the saved models on the test images is as follows :

1) In the "DL Project Final.ipynb" file run the first cell which is just the imports and make sure to change the "FOLDER_PATH" to the appropriate base directory containing training, testing and validations dataset and also the training models for testing. The link for the trained models as well as the prepared dataset has been shared in the Data directory.
2) Run the data generator cell with the heading "Data Generator Creation" (second cell)
3) At the end of the python notebook we have a cell with heading "Model loading, evaluation and result visualization" which contains all the required functions to test the model, we have to run this cell as well.
4) Running the following lines will load the desired model as well as initialize the testing generator. The names of the models are kept based on the architecture to avoid confusion. "Half_unet_reg" means half_UNet with regularization, similarly "half_unet_norm" means Half_UNet with batch normalization.

```
all_models = load_models() # pass any of the following as a string to load the specific model : 'half_unet', 'half_unet_reg', 'half_unet_norm', 'unet', 'nested_half_unet'
gen, num_tests = load_test_data(1) # pass any number higher than 1 which represents the batch_size for the data generator
```

5) Finally running the following function will run all the models and plot the outputs, the function can be customized to view runs on different images from the test set by passing **start** which is the start index and can be between 0-265 (we have 266 test images) and **test_img_count** which is the number of images to run the models on.

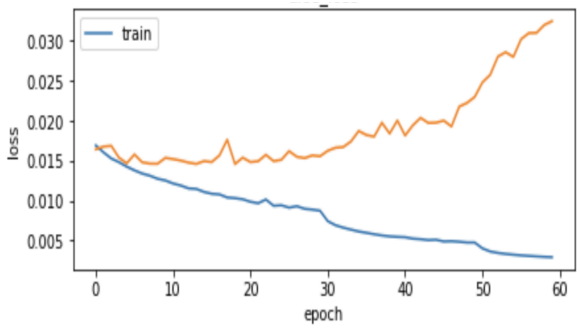
```
#to check the output on different data run the below model with start which is the start index and test_img_count which is the total images to test on
#we have total 266 images in the testing set so the start can be anything between 0 and 265 (inclusive)
visualize_outputs_of__all_models(gen, all_models, start = 3, test_img_count = 4)
```

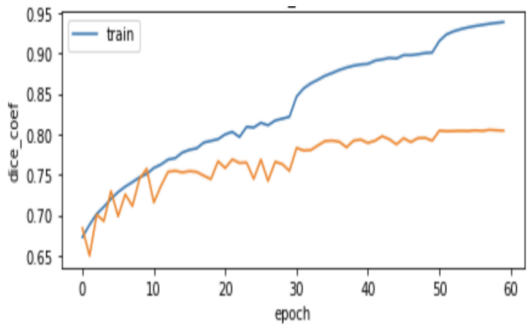
Note : All the code apart from data preparation and augmentation is inside the **DL Project Final.ipynb** file. Appropriate headings and comments have been added for the ease of use

Results and Discussion

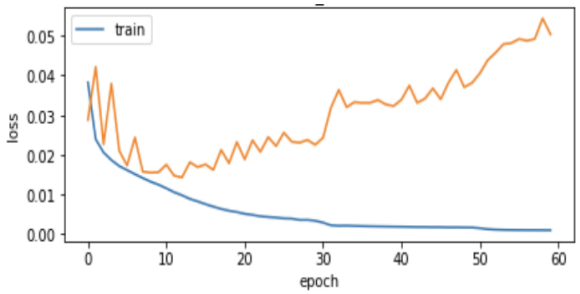
The training was performed on several models as mentioned above. Following are the loss and dice coefficient plot for all of them :

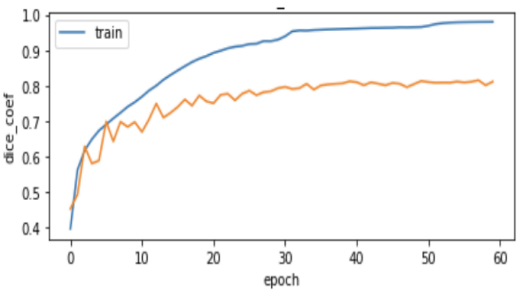
- **Half-UNet**



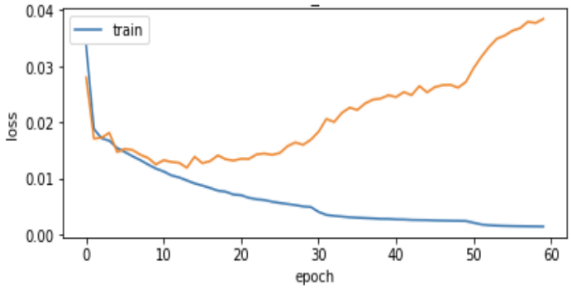


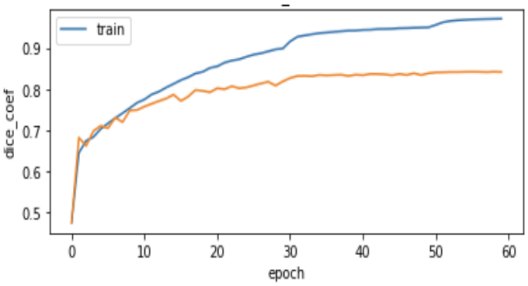
- **Unet**



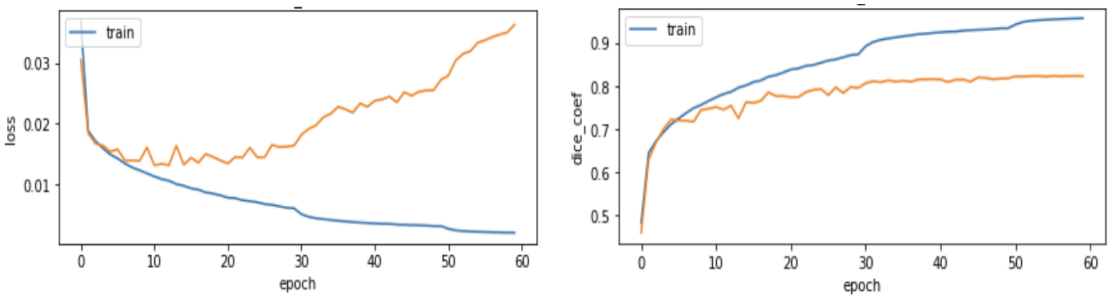


- **Nested Half-UNet**





- **Half-UNet with Batch Normalization**



For the endocardium MRI segmentation dataset, the base Half-Unet model gave a validation dice coefficient of around 0.7711 and there was a lot of overfitting in the model. To deal with that batch normalization was introduced, which helped in reducing the overfitting a bit and gave slightly better results with dice coefficient of around 0.7899. With the same settings when tried on UNet, the results we got were slightly better than the base model of Half-UNet but not better than the version with Batch normalization, its dice coefficient came out to be 0.7789. Finally, the fourth model we tried was the Nested Half-UNet architecture, which outperformed all the other models and gave the dice coefficient of around 0.8121 which was significantly higher than the previous models. When we talk about the trainable parameters there is a huge gap between Unet and the others. Unet has around 140 times more parameters than the base Half-UNet model and around 30 times more than the Nested Half-UNet. The table below summarizes all the results we got in all models we trained and also includes the number of trainable parameters per model. Also, at 30 and 50 epochs we see a spike in the performance, especially for training that is because we are reducing the learning rate by a factor of 2 and 10 at 30 and 50 respectively leading to lower changes and hence taking smaller steps as we reach closer to the global minima.

Models	Dice Coefficient	Trainable Parameters
Half-UNet	0.7711	220,193
UNet	0.7789	31,043,521
Half-UNet with Batch Normalization	0.7899	316,577
Nested Half-UNet	0.8131	984,737

Conclusion and Future Scope

The results shown in the above table as well as in the graphs shows that the Nested Half-UNet model performed better than the rest, but there is still a lot of overfitting in all the models as we can clearly see in the training plots. with extensive data augmentation and better regularization methods would definitely help minimize that and in turn give better dice coefficient and segmentation results. Moreover, the architectures explored here are not properly fine tuned which I believe leaves us with a lot of scope for improvement and far better results.
Moreover, this same concept and the architecture can also be extended to multiclass general image segmentation tasks and not just medical image segmentation. The idea here is to find the least contributing module of the architecture and replace it with a less expensive and slightly less accurate module in order to get a faster model with almost the same performance which can be very useful when it comes to model inference and deployment.

REFERENCES

[1] Half-UNet: A Simplified U-Net Architecture for Medical Image Segmentation by Haoran Lu, Yifei She, Jun Tie and Shengzhou Xu published on 9th June 2022.
https://www.frontiersin.org/articles/10.3389/fninf.2022.911679/pdf

[2] Feature Pyramid Networks for Object Detection by Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie
https://arxiv.org/pdf/1612.03144

[3] U-Net : Convolutional Networks for Biomedical Image Segmentation by Olaf Ronneberger, Philipp Fischer, Thomas Brox

https://arxiv.org/pdf/1505.04597

[4] U^2 -Net: Going Deeper with Nested U-Structure for Salient Object Detection by Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane and Martin Jagersand
https://arxiv.org/pdf/2005.09007.pdf

[5] UNet++ : A Nested U-net Architecture for Medical Image Segmentation by Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang
https://arxiv.org/pdf/1807.10165.pdf

[6] Deep Residual Learning for Image Recognition by Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun and Microsoft Research
https://arxiv.org/pdf/1512.03385.pdf

[7] Data: http://www.eng.usf.edu/cvprg/Mammography/Database.html

[8] Data: https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI

[9] Data: https://sourceforge.net/projects/cardiac-mr/