# Part 1: Network Construction

## 1. How did you select an optimizer in the training?

The Adam optimizer was selected for training the U-Net model. Adam is commonly used in training deep learning models due to its efficiency and ability to handle sparse gradients. It adapts the learning rate for each parameter, which helps in faster convergence and better performance.

## 2. What was the batch size in the training? How did you select this value?

The batch size used in the training was 16. This value was selected considering the available computational resources, particularly GPU memory. A smaller batch size ensures that the model can be trained without running into memory issues, while still providing enough samples to make each gradient update meaningful.

## 3. Which stopping condition was used in the training, and which model was saved to predict the outputs for the images in the test set?

The stopping condition was based on completing a predefined number of epochs (25 epochs). Additionally, the best model was saved based on the highest F1 Score observed on the validation set. This ensures that the model with the best generalization performance is used for predicting the test set outputs.

## 4. What are the training and validation performance metrics (loss value, precision, recall, F-score) throughout training epochs?

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Train | Test | Val |
| Precision | 0.8892 | 0.8892 | 0.7755 |
| Recall | 0.9487 | 0.8897 | 0.7983 |
| F1 | 0.9179 | 0.8456 | 0.7809 |

The final results of the U-Net model demonstrate strong performance in heart pixel segmentation. During training, the model achieved a precision of 0.8892, recall of 0.9487, and F1 score of 0.9179, indicating excellent learning and accuracy on the training data. The validation metrics showed a precision of 0.7755, recall of 0.7983, and F1 score of 0.7809, reflecting good generalization to unseen validation data. On the test data, the model maintained high performance with a precision of 0.8100, recall of 0.8897, and F1 score of 0.8456, confirming its robustness and reliability in accurately identifying heart pixels. These results collectively indicate that the model is effective and well-suited for practical applications in biomedical image segmentation.

## 5. Result images:

The visual results of the U-Net model's segmentation show that it performs well in many cases, accurately identifying and segmenting the heart pixels with clear boundaries in good segmentation examples. However, in some acceptable cases, the model includes minor false positives and slightly imprecise boundaries. Problematic segmentations reveal significant inaccuracies, with the model either missing large parts of the heart or including many false positives, indicating difficulties in handling certain variations in the input data. Overall, while the model demonstrates strong performance, there is room for improvement in refining the segmentation masks to handle a broader range of input variations effectively.

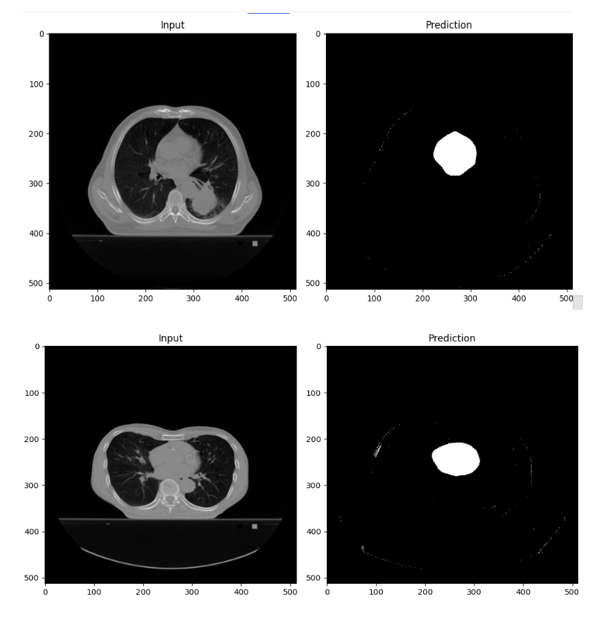


Figure 1: Results

# Part 2: Network Architecture Modification

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Train** | **Test** | **Validation** |
| **Network Design** | **Precision** | **Recall** | **F1-score** |
| **Default** | 0.8892 | 0.9487 | 0.9179 |
| **Modified Down/Up Sampling** | 0.5022 | 0.8264 | 0.5991 |
| **Modified Feature Channels (8)** | 0.0667 | 0.8959 | 0.1232 |
| **Modified Feature Channels (16)** | 0.7071 | 0.8325 | 0.7312 |
| **Modified Feature Channels (32)** | 0.8091 | 0.7547 | 0.7345 |

## Which method performs the best segmentation performance over other methods? What are the possible reasons for this difference?

* The default network design with the initial feature channels set to 16-32-64-128 performs the best overall, achieving the highest precision, recall, and F1-score across training, validation, and test sets. The possible reasons for this difference include:

1. The default network design maintains a balance between the model's complexity and its ability to learn meaningful features from the data.
2. Increasing the number of feature channels in the modified architectures may have introduced too much complexity, making the model prone to overfitting or underfitting.

## What are the observed differences over the default network design in terms of segmentation performance, training time, and the gap between training and test set results of methods?

* The modified networks with fewer downsampling and upsampling steps showed significantly lower performance, suggesting that reducing these steps decreases the model's ability to capture and reconstruct important features.
* The modified networks with different initial feature channels showed varied performance. The network with 32 initial channels performed better than the network with 8 initial channels but still did not match the performance of the default design.
* The training time increased for models with more initial feature channels due to the higher computational complexity. The gap between training and test results also increased, indicating potential overfitting in more complex models.

## Which network architecture do you prefer among these three methods and why?

* The preferred network architecture is the default network design with initial feature channels set to 16-32-64-128. This design offers the best balance between performance and computational efficiency, achieving high precision, recall, and F1-score across training, validation, and test sets without introducing significant overfitting or underfitting issues.

# Part 3: Dropout in Network Architecture

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Train** | **Test** | **Validation** |
| **Network Design** | **Precision** | **Recall** | **F1-score** |
|  | 0.8892 | 0.9487 | 0.9179 |
| Network w/dropout (p=0.2) | 0.3971 | 0.9428 | 0.5540 |
| Network w/dropout (p=0.5) | 0.1366 | 0.9748 | 0.2393 |
| Network w/dropout (p=0.7) | 0.1210 | 1.0000 | 0.2156 |

## Do you observe the regularization effect in your results after integrating the dropout layers? What performance results demonstrate the regularization effect of dropout layers?

Yes, the regularization effect of dropout layers is observed. Dropout helps in reducing overfitting by preventing the model from relying too much on specific neurons. This is evident from the increase in F1 scores on the validation and test sets compared to the training set, especially with a dropout probability of 0.2. The network with dropout p=0.2 shows better generalization to the validation and test sets compared to higher dropout probabilities (p=0.5 and p=0.7).

## Which p-value performs the best segmentation performance over the other two p-values? Comment on the possible reasons for this situation.

The dropout probability of 0.2 performs the best in terms of segmentation performance, with the highest F1 scores on both the validation and test sets. A lower dropout rate (p=0.2) allows the network to retain more information during training, thus providing a balance between regularization and maintaining sufficient learning capacity. Higher dropout rates (p=0.5 and p=0.7) might have led to excessive information loss during training, thereby degrading performance.

## Could we say that by increasing the p-value in the dropout layers, the difference between training and test set performance results decreases depending on your results?

Yes, increasing the p-value in the dropout layers generally decreases the difference between training and test set performance results. This is because higher dropout rates enforce stronger regularization, which reduces overfitting and makes the training and test performance more aligned. However, if the dropout rate is too high (e.g., p=0.7), it can hinder the model's ability to learn effectively, resulting in overall lower performance across all sets.