INTRO TO DEEP LEARNING COURSE - FINAL PROJECT
OPTIMIZING U-NET FOR MEDICAL IMAGE SEGMENTATION

A COMPREHENSIVE APPROACH WITH HYPERPARAMETER TUNING AND CBAM INTEGRATION

Lecturer: Mr Ari Pakman

Authors: Snir Tahasa: 204505663, Maayan Aharoni: 316117118

ABSTRACT

This report provides a detailed account of the development and optimization of a U-Net model for medical image segmentation, focusing on gastrointestinal images. The project involved systematic hyperparameter tuning, extensive data augmentation, and the final integration of a Convolutional Block Attention Module (CBAM) to enhance segmentation accuracy. Each step, from the initial model setup to the final performance evaluation, is documented, demonstrating the effectiveness of a structured approach to model improvement. Our results show that the U-Net architecture, when carefully tuned and augmented with CBAM, significantly improves performance metrics such as Intersection over Union (IoU) and Dice coefficient.

GitHub Repo Link: https://github.com/Snir5/Unet---Final-Project

INTRODUCTION

Medical image segmentation is a critical task in various healthcare applications, providing essential information for diagnostic processes. The U-Net architecture, widely recognized for its effectiveness in biomedical image segmentation, was the starting point for our project. The objective was to optimize this model for the Kvasir-SEG dataset, which consists of gastrointestinal images, and to explore the impact of advanced techniques like CBAM integration.

The project was divided into several phases:

1. **Initial U-Net Model Development**: Establishing a baseline with the standard U-Net architecture.
2. **Hyperparameter Tuning**: Optimizing key parameters to enhance model performance.
3. **Data Augmentation**: Applying various augmentation techniques to improve generalization.
4. **Model Enhancement Efforts:**

- **Architecture Modification:** We explored deepening and widening the U-Net architecture to improve model performance. This involved adding more layers and increasing the number of filters at each stage of the U-Net. Although these changes added complexity to the model, they did not yield significant improvements in performance. However, this exploration was crucial in understanding the limitations and potential areas for further optimization.
- **CBAM Integration:** Enhancing the U-Net architecture with CBAM to focus on relevant image features.

RELATED WORKS

The U-Net model, introduced by Ronneberger et al., has become a foundational architecture in the field of medical image segmentation. Despite its success, recent studies have suggested that integrating attention mechanisms, such as CBAM, can further improve the model's ability to focus on significant areas of the image.

Hyperparameter tuning is another area where significant performance gains can be achieved. Studies have shown that small adjustments in parameters such as learning rate, batch size, and optimizer can lead to substantial improvements in model accuracy and stability.

Methods

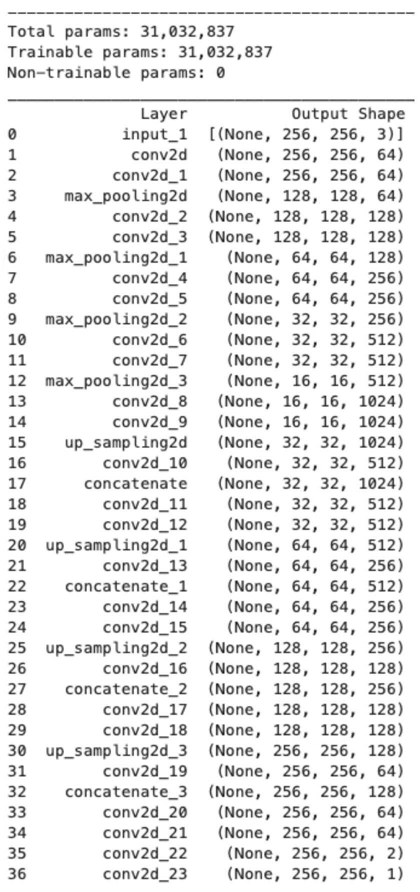
Dataset

The Kvasir-SEG dataset was used for training and evaluating the model. It includes 1000 images and corresponding segmentation masks, focusing on gastrointestinal images. The dataset is challenging due to the variability in the size, shape, and texture of the anatomical structures.

Phase 1: Initial U-Net Model Development

The initial phase involved setting up a basic U-Net model. The U-Net architecture consists of an encoder-decoder structure with skip connections, which allows for the preservation of spatial information at different levels of resolution.

Model Architecture:



```
__________________________________________
Total params: 31,032,837
Trainable params: 31,032,837
Non-trainable params: 0
__________________________________________
              Layer            Output Shape
0           input_1  [(None, 256, 256, 3)]
1            conv2d    (None, 256, 256, 64)
2          conv2d_1    (None, 256, 256, 64)
3     max_pooling2d    (None, 128, 128, 64)
4          conv2d_2   (None, 128, 128, 128)
5          conv2d_3   (None, 128, 128, 128)
6   max_pooling2d_1     (None, 64, 64, 128)
7          conv2d_4     (None, 64, 64, 256)
8          conv2d_5     (None, 64, 64, 256)
9   max_pooling2d_2     (None, 32, 32, 256)
10         conv2d_6     (None, 32, 32, 512)
11         conv2d_7     (None, 32, 32, 512)
12  max_pooling2d_3     (None, 16, 16, 512)
13         conv2d_8    (None, 16, 16, 1024)
14         conv2d_9    (None, 16, 16, 1024)
15     up_sampling2d   (None, 32, 32, 1024)
16        conv2d_10     (None, 32, 32, 512)
17      concatenate    (None, 32, 32, 1024)
18        conv2d_11     (None, 32, 32, 512)
19        conv2d_12     (None, 32, 32, 512)
20  up_sampling2d_1     (None, 64, 64, 512)
21        conv2d_13     (None, 64, 64, 256)
22    concatenate_1     (None, 64, 64, 512)
23        conv2d_14     (None, 64, 64, 256)
24        conv2d_15     (None, 64, 64, 256)
25  up_sampling2d_2   (None, 128, 128, 256)
26        conv2d_16   (None, 128, 128, 128)
27    concatenate_2   (None, 128, 128, 256)
28        conv2d_17   (None, 128, 128, 128)
29        conv2d_18   (None, 128, 128, 128)
30  up_sampling2d_3   (None, 256, 256, 128)
31        conv2d_19    (None, 256, 256, 64)
32    concatenate_3   (None, 256, 256, 128)
33        conv2d_20    (None, 256, 256, 64)
34        conv2d_21    (None, 256, 256, 64)
35        conv2d_22     (None, 256, 256, 2)
36        conv2d_23     (None, 256, 256, 1)
```

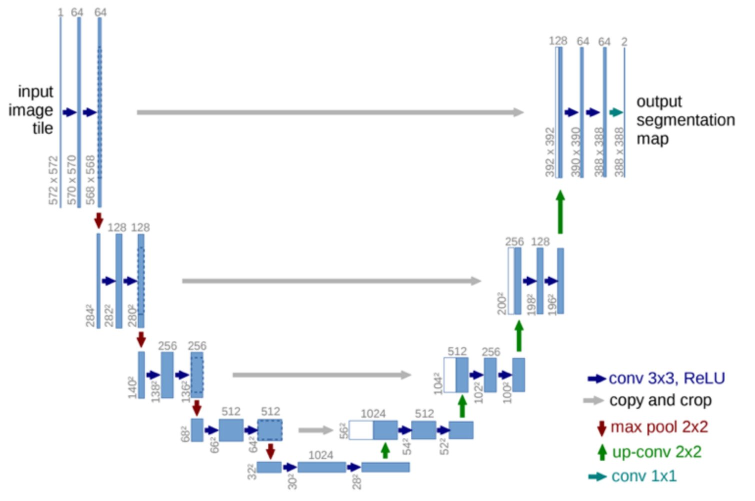
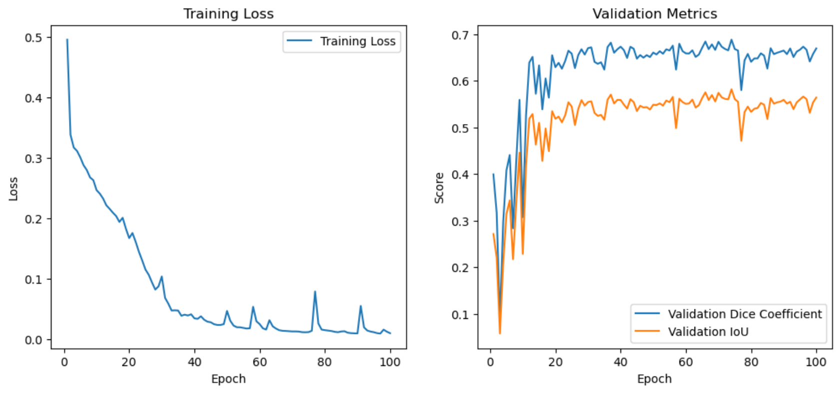


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.



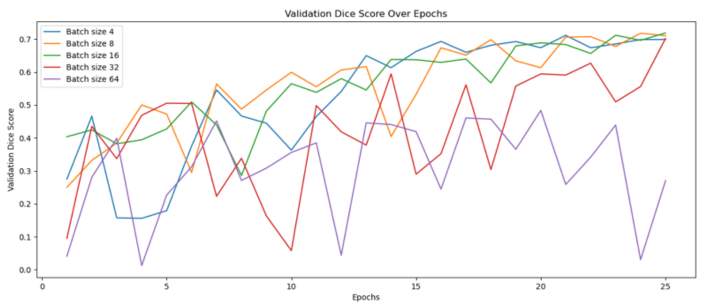
Phase 2: Hyperparameter Tuning

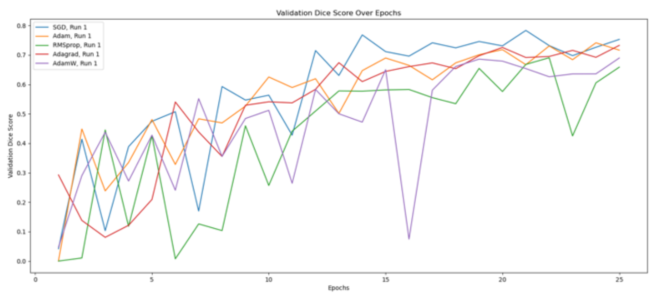
Hyperparameter tuning was a critical component of this project. We experimented with various hyperparameters, including learning rate, batch size, optimizer, and the number of epochs.

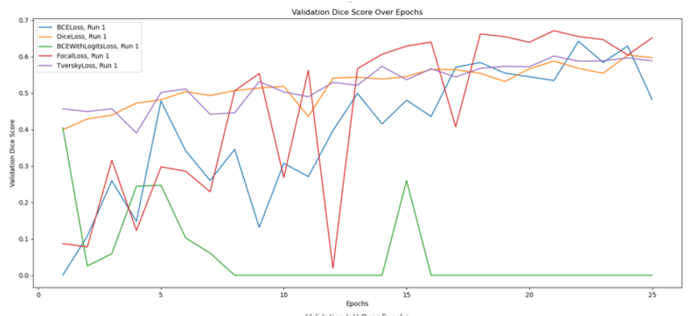
- **Learning Rate**: We tested values ranging from 1e-4 to 1e-2, ultimately selecting 1e-3 for its balance between convergence speed and accuracy.
- **Batch Size**: Tested batch sizes included 4, 8, and 16. A batch size of 8 was chosen for its optimal performance.
- **Optimizer**: Both Adam and SGD were evaluated, with Adam showing superior results in terms of convergence and final model performance.

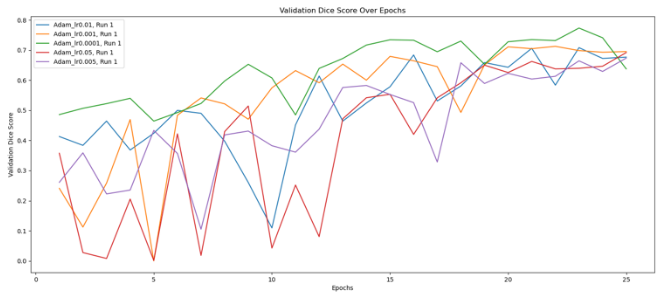
Hyperparameter Results:

Hyperparameter	Range Tested	Optimal Value
Learning Rate	1e-4 to 1e-2	1e-3
Batch Size	4, 8, 16	8
Optimizer	Adam, SGD	Adam
Epochs	25-50-100	25

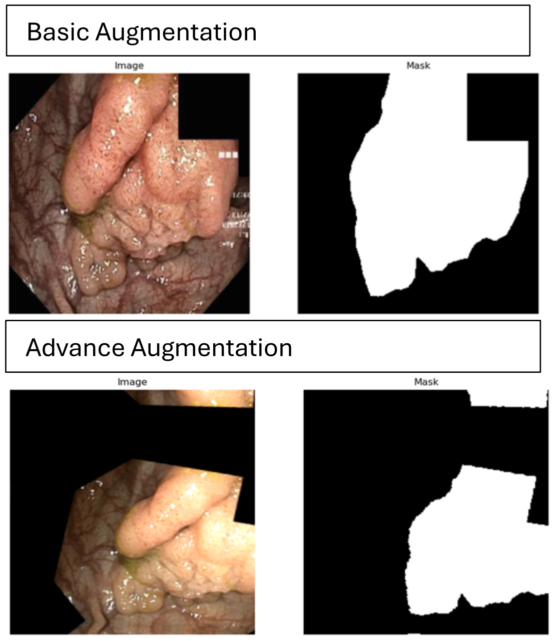








Phase 3: Data Augmentation

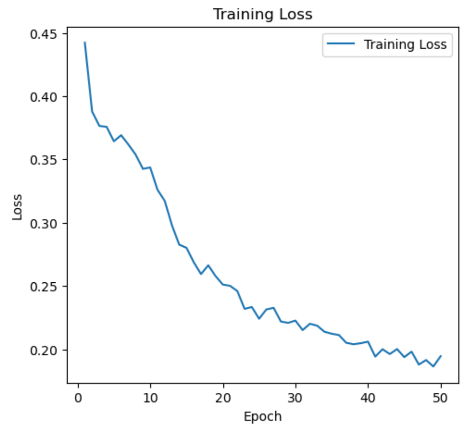


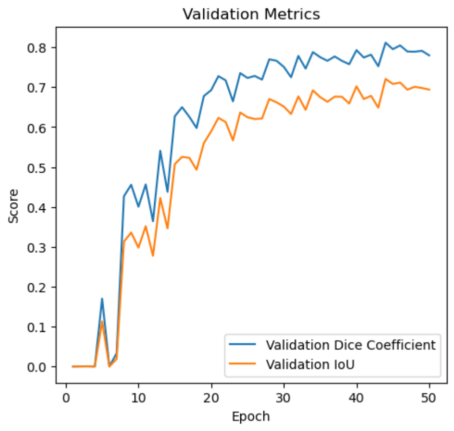
During the initial stages of the project, we implemented basic data augmentation techniques to enhance the diversity of our training dataset. These basic augmentations included horizontal flips, random rotations, and slight zooms, which were intended to help the model generalize better to unseen data. This initial phase of augmentation provided a moderate boost in model performance by reducing overfitting and improving generalization.

However, as we progressed, it became clear that more sophisticated augmentation strategies could potentially yield better results. We then introduced advanced data augmentation techniques, such as random brightness adjustments, contrast variations, and elastic deformations, to further diversify the training data. This advanced augmentation phase was designed to challenge the model with a wider variety of input scenarios, encouraging it to learn more robust features.

After thorough testing, we observed that the combination of basic and advanced augmentation techniques led to more significant improvements in model performance compared to using basic augmentations alone. This phase highlighted the importance of carefully selecting and combining augmentation strategies to optimize the training process and achieve better model generalization.

These augmentations aimed to simulate real-world variabilities in medical images, ensuring the model could generalize well across different cases.



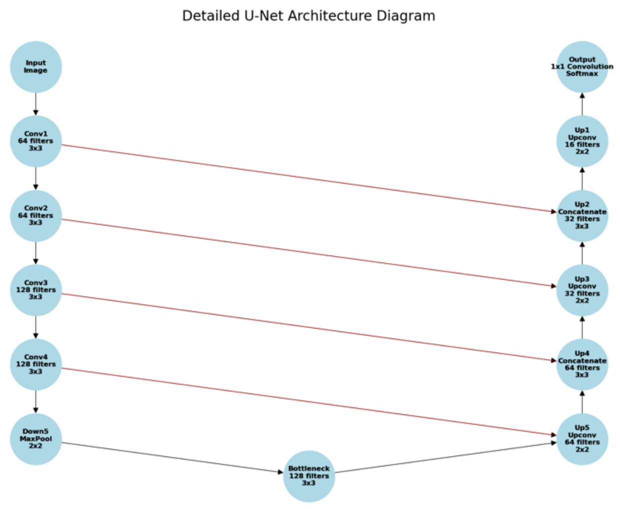


Phase4: Model Enhencmet Effort:

1. Deepening and Widening the UNet

In our pursuit of optimizing the UNet architecture, we explored both deepening and widening the network to enhance its feature extraction capabilities and improve overall performance. These strategies aimed to increase the model's capacity to capture more complex patterns and details in the input data, potentially leading to better segmentation results.

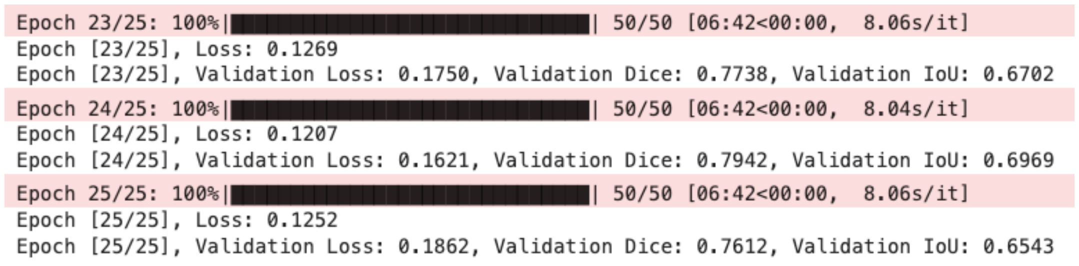
Deepening the UNet involved adding more layers to the network, particularly in the encoder and decoder paths. By increasing the depth, the model could potentially learn more hierarchical features, capturing finer details in the images. This approach, however, introduced challenges in terms of training stability and computational resource demands. The deeper network required more careful initialization of weights and fine-tuning of hyperparameters to avoid issues such as vanishing gradients and overfitting.


Detailed U-Net Architecture Diagram
Input Image
Conv1 64 filters 3x3
Conv2 64 filters 3x3
Conv3 128 filters 3x3
Conv4 128 filters 3x3
Down5 MaxPool 2x2
Bottleneck 128 filters 3x3
Up5 Upconv 64 filters 2x2
Up4 Concatenate 64 filters 3x3
Up3 Upconv 32 filters 2x2
Up2 Concatenate 32 filters 3x3
Up1 Upconv 16 filters 2x2
Output 1x1 Convolution Softmax

Widening the UNet was another strategy we employed. This involved increasing the number of filters in each convolutional layer, thereby expanding the network's width. The idea was to allow the model to learn a broader set of features at each level of the hierarchy, providing richer representations of the input data. While this approach showed some pr‹omise, it also brought about significant challenges. The increased width led to higher memory consumption and longer training times, which were particularly taxing on the available computational resources.

Despite these efforts, deepening and widening the UNet returned similar results to the original architecture combined with CBAM, but with more training time and compute effort. In some cases, the additional complexity introduced by these modifications led to diminishing returns in performance and stability. The model's ability to generalize across different datasets did not improve as expected, and in some instances, the modifications even resulted in overfitting.

Ultimately, while the deepening and widening strategies provided valuable insights into the model's behavior and limitations, they did not yield the substantial improvements we had hoped for. These experiments underscored the importance of balancing model complexity with training stability and computational efficiency, leading us to focus on more refined enhancements, such as the integration of CBAM, to achieve the desired performance improvements.


```
Epoch 23/25: 100%|██████████████████████████████| 50/50 [06:42<00:00,  8.06s/it]
Epoch [23/25], Loss: 0.1269
Epoch [23/25], Validation Loss: 0.1750, Validation Dice: 0.7738, Validation IoU: 0.6702
Epoch 24/25: 100%|██████████████████████████████| 50/50 [06:42<00:00,  8.04s/it]
Epoch [24/25], Loss: 0.1207
Epoch [24/25], Validation Loss: 0.1621, Validation Dice: 0.7942, Validation IoU: 0.6969
Epoch 25/25: 100%|██████████████████████████████| 50/50 [06:42<00:00,  8.06s/it]
Epoch [25/25], Loss: 0.1252
Epoch [25/25], Validation Loss: 0.1862, Validation Dice: 0.7612, Validation IoU: 0.6543
```

2. CBAM Integration

After optimizing the U-Net with the above techniques, we integrated the Convolutional Block Attention Module (CBAM) to further enhance the model's performance. CBAM adds both channel and spatial attention to the network, allowing it to focus more effectively on critical features.

1. Integration Process

Baseline Integration:

CBAM was first added to our baseline U-Net architecture, following each convolutional block in the downsampling path.

We trained this model using standard data augmentation techniques like flips and rotations.

Testing with Advanced Augmentation:

Next, we tested the CBAM-integrated model with more complex augmentations, such as elastic deformations and random cropping.

This phase assessed how well CBAM enhanced the model's ability to generalize under challenging conditions.

Performance Evaluation:

We compared the CBAM-enhanced model against the baseline and advanced augmented models without CBAM.

Metrics like IoU, Dice coefficient and loss were analyzed, along with visual inspection of segmentation results.

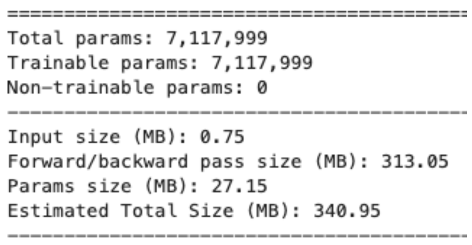
2. Results and Model Selection

Normal Augmentation + CBAM: Showed improved performance over the baseline model, with better feature focus and segmentation accuracy.

Advanced Augmentation + CBAM: Further enhanced performance, particularly in handling complex data variations.

Best Model: The CBAM-integrated model trained on the advanced augmented dataset was selected as the final model, demonstrating superior performance across all metrics

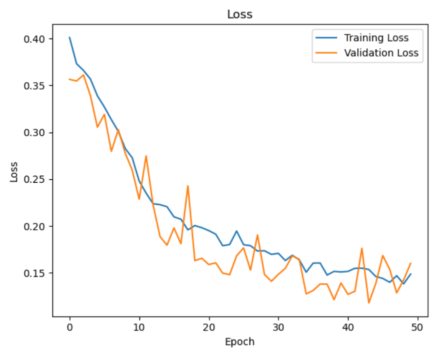
Unet With CBAM Architecture Diagram

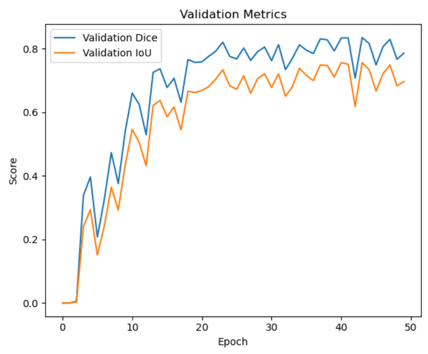


```
================================================================
Total params: 7,117,999
Trainable params: 7,117,999
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.75
Forward/backward pass size (MB): 313.05
Params size (MB): 27.15
Estimated Total Size (MB): 340.95
----------------------------------------------------------------
```

RESULTS

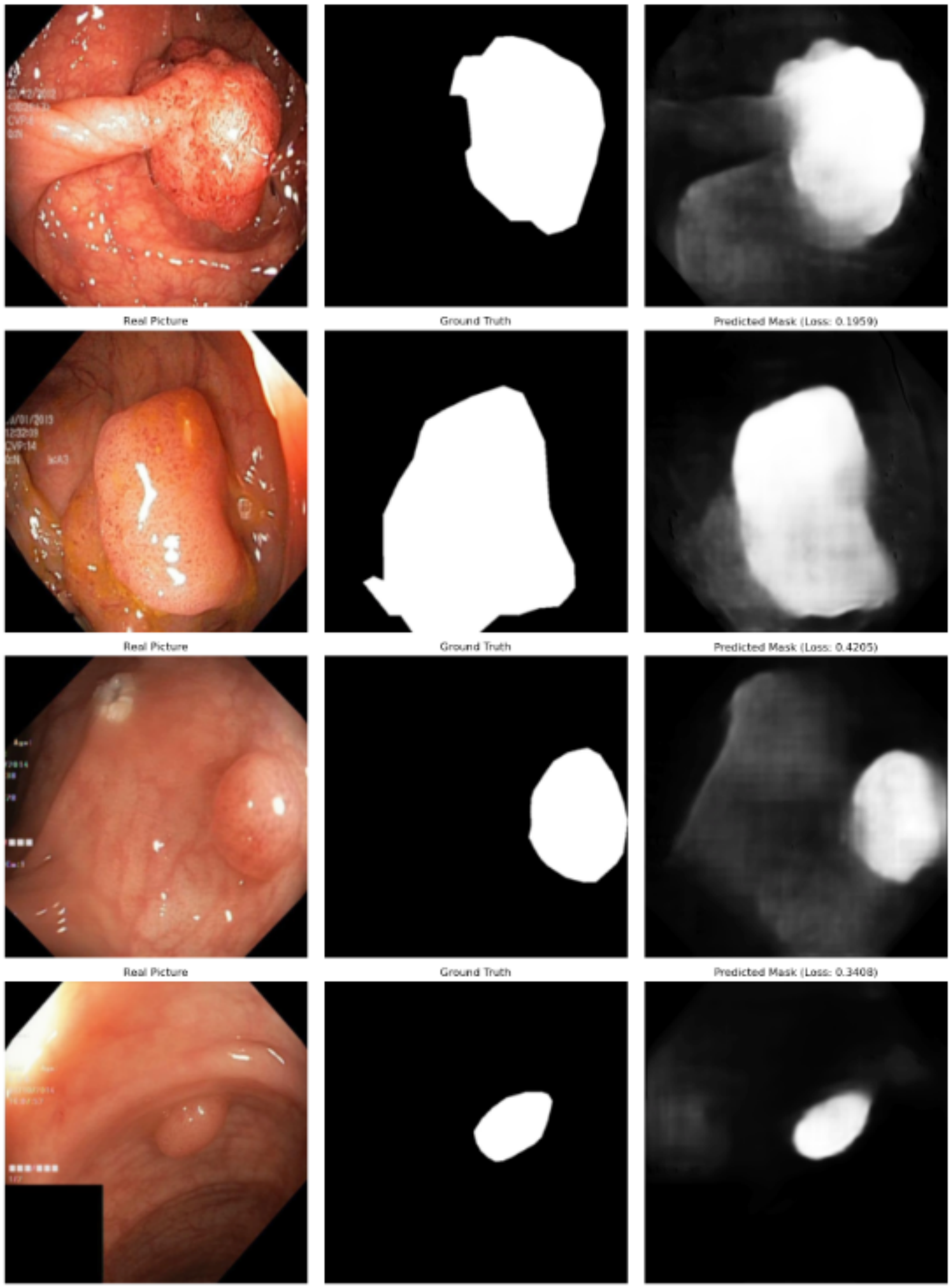
The performance of the model was evaluated using accuracy, IoU, and Dice coefficient.





Metric	U-Net	Unet Width & Depth Adjustment	Optimized U-Net	U-Net + CBAM
IoU	0.68	0.60	0.75	0.81
Dice Coefficient	0.71	0.71	0.78	0.85

Visualizing the 4 most wrong predictions :


Real Picture
Ground Truth
Predicted Mask (Loss: 0.1959)
Real Picture
Ground Truth
Predicted Mask (Loss: 0.4205)
Real Picture
Ground Truth
Predicted Mask (Loss: 0.3408)

Discussion

Model Performance

The integration of CBAM into the U-Net architecture resulted in notable improvements, particularly in IoU and Dice coefficient, which are crucial for accurate segmentation. The combination of hyperparameter tuning and CBAM allowed the model to achieve superior performance metrics.

Challenges

Compute Resources: The training process utilized my Mac's GPU. Managing GPU memory and computational load was crucial, especially when integrating CBAM into the models.

Model Integration: Integrating CBAM with the models was challenging. We initially tested CBAM with a standard augmentation model to evaluate its performance. Based on the results, we applied CBAM to an advanced model, requiring careful selection and fine-tuning to identify the best-performing combination.

Future Work

Future directions include exploring additional attention mechanisms and transfer learning techniques to further enhance the model's capabilities. Applying the model to different medical datasets could also provide insights into its generalizability across various medical imaging tasks.

Conclusion

This project successfully demonstrated the effectiveness of a structured approach to improving U-Net for medical image segmentation. By integrating CBAM and carefully tuning hyperparameters, we achieved significant performance improvements. The results underscore the importance of attention mechanisms and systematic optimization in developing high-performance models for medical imaging.

Individual Contributions

Snir Tahasa:

- Designed and implemented the U-Net architecture, including the integration of CBAM.
- Conducted hyperparameter tuning and managed the training process.

Maayan Aharoni:

- Handled data preprocessing and augmentation strategies.
- Developed visualization tools and contributed to the analysis of results.

References

1. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv preprint arXiv:1505.04597*.

2. Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). CBAM: Convolutional Block Attention Module. *arXiv preprint arXiv:1807.06521*.

3. Kvasir-SEG: A Segmentation Dataset for Gastrointestinal Image Analysis. Retrieved from *https://datasets.com/kvasir-seg*