Comprehensive U-Net Model Analysis with Experimental Evaluations for Medical Image Segmentation

Candidate Numbers: 15, 18, 17, 33, 42

School of Economics, Innovation and Technology
Kristiania University College
Oslo, Norway

***Abstract*—This research paper explores the application of deep learning for medical image segmentation by using the Kvasir Instrument dataset. The main goal is to develop a U-Net Convolutional Neural Network model that can perform pixel-wise segmentation in order to identify and distinguish medical instruments from the background of the endoscopic images. In addition, five exploratory experiments: Image resolution, data augmentation techniques, training data size, optimization methods, and activation functions - were conducted to evaluate the model's overall performance and its ability to generalize. For the model evaluation, several metrics were used including Dice Coefficient (DSC), Intersection over Union (IoU), precision, recall, and overall accuracy. The results and the findings provided an insight of how the different factors influence segmentation accuracy of the model, while highlighting possible areas for improvement, in order to create a more robust and reliable model for medical image segmentation.**

***Index Terms*—Deep Learning, Medical Image Segmentation, U-Net, Image Resolution, Data Augmentation, Optimizers, Activation Functions**

I. Introduction

The increasing use of technology across various industries has brought amazing benefits, alongside some challenges. In healthcare, however, these technological advancements have revolutionized a number of areas, including diagnosis, treatment planning, and surgical navigation, particularly through the use of medical image analysis. Image segmentation, which involves partitioning an image into meaningful components for better interpretation, is a key component. Medical segmentation still remains a very complex challenge due to the variability of medical images, their complex anatomical structures, and the need for precise, pixel-level accuracy in predictions, despite carrying big potential.

The main goal for this research project was to develop a U-Net Convolutional Neural Network (CNN) model for segmenting medical instruments from the Kvasir Instrument dataset. The U-Net architecture was chosen due to its effective encoder-decoder structure, which allowed for pixel-level segmentation by capturing both spatial and contextual information [1].

The report is divided into multiple sections. The main section provides a comprehensive explanation of the U-Net model, which is the main foundation for all experiments conducted in this study. Thereafter, the following sections detail the methodology, experimental setup, and results of all the experiments conducted to determine their influence on the U-Net model's performance and generalization. These experiments include image resolution (Section 1), data augmentation techniques (Section 2), training data size (Section 3), optimization methods(Section 4), and activation functions (Section 5).

Notebook	Purpose	Experiment	Parameter
Notebook 1	U-Net workflow template	Baseline setup	Batch size: 8
Notebook 2	Image Resolution	64 x64, 128x128, 256x256	Epochs: 30
Notebook 3	Data augmentation	RandomRotation, Color Jit, Affine	Early stopping (patience = 5)
Notebook 4	Training data size	20%, 50%, 100%	Optimizer: Adam
Notebook 5	Optimizer comparison	Adam, SGD, RMSprop	Loss Function: BCELoss
Notebook 6	Activation functions	ReLU, Leaky ReLU, Tanh	Scheduler: CosineAnnealingLR

Fig. 1. Overview of Notebooks Structure and Experiments

Note: *The master Notebook 1 serves as a baseline template for all the experiments, implementing the standardized U-Net workflow outlined in the methodology. The team members have duplicated and modified this template for each experiment, in order to ensure code consistency as well as code structure and tools across all tests. Only experimental variables (e.g., resolution, augmentation, optimizer, activation function) are adjusted in subsequent notebooks. The notebook was created and assisted by each team member.*

II. Methodology

In this report, the methodology follows the pipeline workflow depicted in Figure 1.

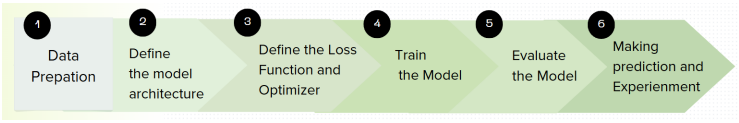

Fig. 2. Data Pipeline

A. *Dataset Preparation*

*1) **Loading the data**:* Kvasir-Instrument dataset was used for the binary segmentation of medical instruments in endoscopic images. The dataset was obtained from https://datasets.simula.no/kvasir-instrument/ and consisted of four files, training/testing splits alongside images and masks that were loaded into the Jupyter Notebook. The dataset contained 590 annotated images that had resolutions ranging from 720x576 to

1280x1024, alongside corresponding segmentation masks and bounding boxes for gastrointestinal (GI) tools such as snares and balloons. The dataset was predefined with 478 training images and 118 test images [2].

*2) **Data Exploration**:* After loading the dataset, an exploratory analysis was conducted in order to understand the dataset's structure and characteristics.

- **Class Distribution:** Examined by plotting the training images alongside corresponding masks. While the training images were scaled to the range [0,1], the mask values ranged from [0, 255], which fall outside of the expected model range. To address this disparity, normalization was implemented in later steps.

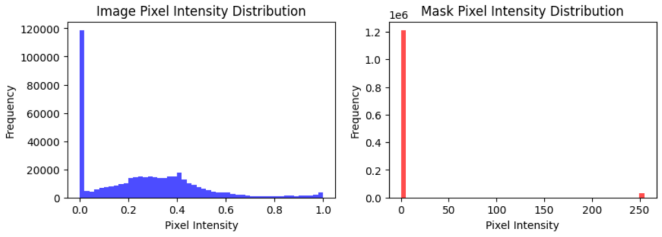

Fig. 3. Pixel Intensity Distribution Before Verify Binary (0, 1) Values

- **Image-Mask Pairing:** Displayed several samples of image-mask pairs to ensure correct pairing.

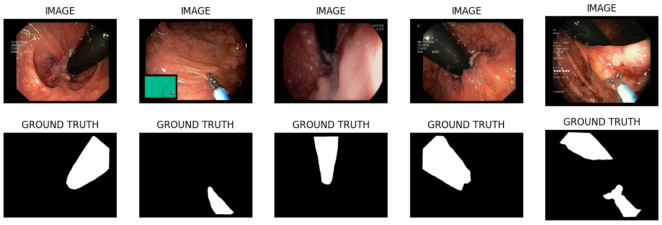

Fig. 4. displays multiple images from the Kvasir-Instrument dataset along with their corresponding masks.

*3) **Defining Transformation and Augmentation**:*

- **Resizing:** Images were resized to 256 x 256 pixels for uniformity.
- **Image Normalization:** The RGB images were normalized using a mean and standard deviation of [0.5, 0.5, 0.5].
- **Mask Normalization:** Masks were scaled to the range of [0,1] to match the model's expected input.

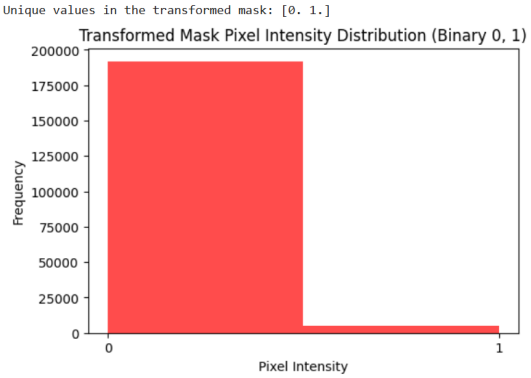

Fig. 5. Mask pixel Intensity Distribution after Verify Binary (0, 1) Values

*4) **Define Custom Dataset**:* A custom dataset class, KvasirDataset, was implemented to handle the non-standard format of the Kvasir-Instrument dataset. This class ensured compatibility with PyTorch's data loading mechanisms and facilitated integration with data loaders. It has three main functions [3]:

- **init:** Initializes the dataset paths and parameters.
- **len:** Returns the number of samples.
- **getitem:** Loads an image and its corresponding mask.

*5) **Data Loader**:* Using the KvasirDataset class, train_loader and test_loader were created to facilitate the batch processing during model's training and evaluation. Each batch of size 8, consisted of image-mask pairs, while the training set consisted of 472 images, divided into 59 training batches (472 / 8) and the testing set contained 118 images, divided into 15 testing batches (118 / 8). Both the images and masks were resized, transformed and batched into 4 dimensional tensors of the shape [bach_size, channels, height, width]. The images have a Torch tensor size of 8, 3, 256, 256 (8 batches, 3 RGB channels, 256x256 pixels), and the masks have a Torch tensor size of 8, 1, 256, 256 (8 batches, 1 grayscale channel, 256x256 pixels) [3].

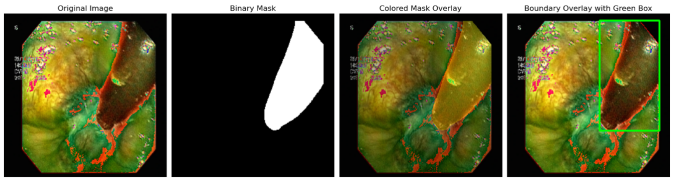

Fig. 6. Images and masks after added in train and test loader iteration

B. Model Architecture

Several model architectures, including FCN, Mask R-CNN, and DeepLab, are commonly used for image segmentation tasks. However, for medical image segmentation, the U-Net architecture has proven to be particularly effective due to its encoder-decoder structure with skip connections [1] [4]. This is the main reason for selecting and implementing the U-Net model from scratch for the task. The model converts the RGB

input image with a batch size of 3, 256, 256 into a binary segmentation mask with a batch size of 1, 256, 256.

Model's Architecture Overview:

- **Input:** Each image that is input into the network is an RGB image with a batch size of 3, 256, 256.
- **Encoder-Decoder with Skip Connections: The encoder** uses convolutional blocks with ReLU and dropout, slowly increasing channels (64, 128, 256, 512, 1024) while reducing spatial dimensions due to pooling operations. At **the bottom neck** of the network, an upsampling (deconvolution) layer reduces the channel count from 1024 to 512. **The decoder**, with skip connections, upscales the concentrated features from the corresponding encoder layers, finally outputting a segmentation mask of shape [batch_size, 1, 256, 256] (binary Kvasir mask with a single channel, where each pixel is classified as 0 or 1.
- **Binary Mask Output:** The final layer consists of a 1x1 convolution which reduces the channel count to 1, followed by a sigmoid activation function that results in a binary segmentation mask, where each pixel is classified as either belonging to the object of interest such as a medical instrument or to the background [1].

For all experiments, Binary Cross-Entropy Loss and Adam optimizer were applied with ReLU and Sigmoid activations, ensuring stable training and effective model performance.

C. Define the Loss Function, Optimizer, and Learning Rate Scheduler

- **Loss Function:** Binary Cross Entropy (BCE) loss, also known as log loss, was used to quantify the difference between predicted and true segmentation masks. This method is suitable for binary tasks like those in the Kvasir dataset.
- **Activation Functions:** ReLU was applied in hidden layers, while Sigmoid was used in the final layer of the U-Net model for binary classification.
- **Optimizer:** The Adam optimizer was selected as a baseline due to its adaptive learning rate capabilities and proven effectiveness across various tasks, making it a reliable choice for this model.
- **Learning Rate Scheduler:** CosineAnnealingLR was used to adjust the learning rate during training, improving the model's ability to converge by reducing the learning rate smoothly as training progresses.
- **Epochs and Batches:** Defines the number of times that the learning algorithm will learn through the entire training dataset. Training was conducted over 30 epochs with a batch size of 8. The dataset consisted of 412 images, with 59 iterations per epoch.

For all experiments, Binary Cross-Entropy Loss and Adam optimizer were applied with ReLU and Sigmoid activations, ensuring stable training and effective model performance.

D. Training Setup

After initializing the model, optimizer and the loss function, the model is set to training mode using `model.train()`. The function `train_model()` trains the segmentation model for one epoch by processing each batch of images and masks from the Kvasir dataset.

- **Forward Pass:** Each batch of images and masks passes through the model, generating predicted masks that highlight important regions in the images, such as medical instruments.
- **Loss Calculation:** Binary Cross-Entropy (BCE) loss measures the discrepancy between the predicted and actual masks. A higher loss indicates a less accurate prediction.
- **Backward Propagation:** Gradients are calculated based on the loss for each batch to update model weights.
- **Weight Adjustment:** The model's weights are updated using the `optimizer.step()` in the direction that minimizes the loss.
- **Learning Rate Scheduler:** The learning rate is adjusted according to the cosine annealing scheduler, which improves the training convergence.
- **Optimizer Step:** The model's parameters are updated with the optimizer, which measures how well the loss is minimized and how the segmentation accuracy is improved.
- **Logging Metrics:** Metrics like average training loss, accuracy, Dice Coefficient (DSC), and Intersection over Union (IoU) are tracked to monitor the model's performance.
- **Early Stopping:** Early stopping is employed to stop the training whenever the model does not show any more improvement after a set number of epochs (`patience = 5` in this study). This ensures that learning is stopped when no meaningful improvements are made.

E. Evaluation Setup

The evaluation function follows a workflow similar to training but uses the test set to assess the model's performance on unseen endoscopic images. This ensures model generalization and accurately segments key regions. This accuracy is important as it should improve the same as training progresses.

- **Set Model to Evaluation Mode:** Use `model.eval()` to disable dropout and batch normalization, ensuring consistent predictions.
- **Track Test Loss:** Initialize variables like `test_loss` to accumulate the loss over the test set.

- **Iterate Over Test Data:** Process each batch of test images and masks from the Kvasir test loader to generate predictions and compare them to actual labels.
- **Concatenate Results:** Gather all true and predicted masks across batches to calculate metrics for the entire test dataset.
- **Return Metrics:** Compute and return metrics such as Dice Coefficient (DSC), Intersection over Union (IoU), and test loss, summarizing the model's segmentation accuracy for medical instruments and regions in the Kvasir dataset.

F. Performance Evaluation Workflow

*1) **Evaluation Metrics**:* The choice of evaluation metrics reflects the report's focus on segmentation. Recommended metrics for the task include Dice Similarity Coefficient (DSC), F1-score, and Jaccard Index of Intersection over Union (IoU), along with other standard segmentation metrics such as precision, recall, overall accuracy, and loss, as follows [5]:

- **Dice Coefficient (DSC):** A metric that measures the overlap between the predicted and ground truth masks, ranging from 0 to 1. It is especially useful for imbalanced datasets as it considers both false positives and negatives. **Formula:** $DSC = \frac{2\times|A\cap B|}{|A|+|B|}$ [6]
- **Intersection over Union (IoU):** Also known as the Jaccard Index, IoU measures the ratio between predicted and ground truth masks, with a range of 0 to 1. It is a useful metric for segmentation tasks as it accounts for both false negatives and false positives. **Formula:** $IoU = \frac{|A\cap B|}{|A\cup B|}$ [7]
- **Precision:** Indicates the proportion of correctly identified pixels with a positive identification.
- **Recall:** Indicates how well the model can recognize every true positive pixel.
- **F1-Score:** A performance metric that strikes a balance between recall and precision.
- **Accuracy:** Represents the overall correctness of pixel classification.
- **BCE Loss:** Used to evaluate the model's loss.

*2) **Segmentation Mask vs. Ground Truth Mask from the Test Loader**:* Performance beyond the numerical metrics requires visualizing the segmentation mask alongside the ground truth mask. The predicted mask is the model's attempt to recreate the areas of interest (such as surgical instruments) highlighted by the manually annotated ground truth [8].

- **Overlap Assessment:** High segmentation accuracy is indicated when the predicted and ground truth masks overlap almost perfectly, as seen with high Dice Similarity Coefficient (DSC) values (e.g., 0.81). Low DSC values (e.g., 0.46) highlight missed segmented regions.
- **IoU Visualization:** High Intersection over Union (IoU) values indicate precise segmentation, while low values reveal discrepancies, helping to identify areas where the model is struggling.

*3) **Visualization**:* Tools like Weights and Biases (WandB), were used to track and display model performance in real time, and visualize the metrics obtained.

III. Experiments and Results

Main Section: The master U-Net workflow

The baseline performance of the U-Net model on the Kvasir dataset is summarized in Figures 7 and 8. The results show high accuracy for both training and test sets (94.84%) with a low loss of (0.12) across 8 epochs, indicating the model learned effectively with minimal overfitting. The DSC and IoU metrics show balanced accuracy improvements across training and testing, Although precision and recall show slight fluctuations, the model is able to maintain reliability in detecting regions of interest. Overall, the model is robust, generalizes well to test data and is suitable for pixel-wise segmentation tasks, making it a strong baseline for all the experiments.

Metric	Values
Epoch	8
Test Accuracy	94.84756
Test DSC	0.72887
Test F1-Score	0.72887
Test IoU	0.57340
Test Loss	0.12741
Test Precision	0.71532
Test Recall	0.74293
Train Accuracy	93.96435
Train DSC	0.65342
Train IoU	0.49673
Train Loss	0.15250

Fig. 7. Performance Metrics for U-Net on the Kvasir Dataset

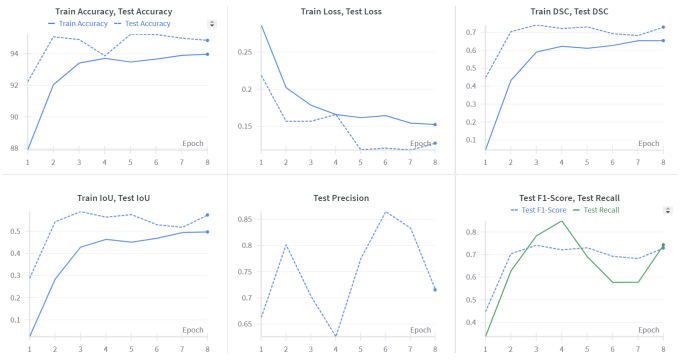

Fig. 8. Metrics Visualization, click to view the live version **Click here**

EXPERIMENT 1: IMAGE RESOLUTION

This study examines how different image resolutions affect the performance of a U-Net model in binary segmentation tasks. The evaluation focused on image resolutions: 64x64, 128x128, and 256x256. Over multiple training epochs, the performance was evaluated using metrics like Dice Similarity Coefficient (DSC), Intersection over Union (IoU), accuracy, and loss. Prior research indicates that low-resolution images have the potential to bias deep learning models and diminish segmentation accuracy [9]. This emphasises the significance of properly examining how resolution impacts segmentation performance.

A. *Methodology*

The dataset consisted of 472 training and 118 testing samples. Images and masks were resized to the target resolutions and standardized to 256x256 to ensure consistent input dimensions. We evaluate the effect of resolution changes by implementing a custom transformation function. A custom PyTorch Dataset class, kvasirDataset was implemented to handle dynamic resolution-specific transformations and preprocessed the data into tensors.This function resizes images and masks by downscaling to target resolutions of 64 x 64, 128 x 128, and then upscaling them back to 256 x 256. For each resolution, the image dimensions were transformed to torch.Size([8,3,256,256]) for images and torch.Size([8,1,256,256]) for masks, ensuring everything was set up correctly before training the U-Net model. The same configuration, optimizer, loss function, and performance metrics as the baseline setup were used for training and evaluation. Metrics like IoU, Dice Coefficient (DSC), and pixel accuracy were averaged to compare the results for all three resolutions. Weights & Biases (W&B) helped log and visualize the metrics. Predicted masks were checked against ground truth masks, and IoU scores were calculated for each sample.

B. *Result*

Metric	256 x 256	128 x 128	64 x 64
Epochs	12	12	25
Test Accuracy	94.82165	95.65472	97.33898
Test DSC	0.75518	0.76694	0.79296
Test F1-Score	0.75518	0.76694	0.79296
Test IoU	0.60666	0.62198	0.65695
Test Loss	0.12543	0.13231	0.13690
Test Precision	0.64801	0.65791	0.78738
Test Recall	0.90482	0.91928	0.79863
Train Accuracy	94.51223	92.38212	89.59167
Train DSC	0.76389	0.73070	0.67696
Train IoU	0.62525	0.58910	0.52608
Train Loss	0.12788	0.14240	0.17306

Fig. 9. Comparison of Resolution scores

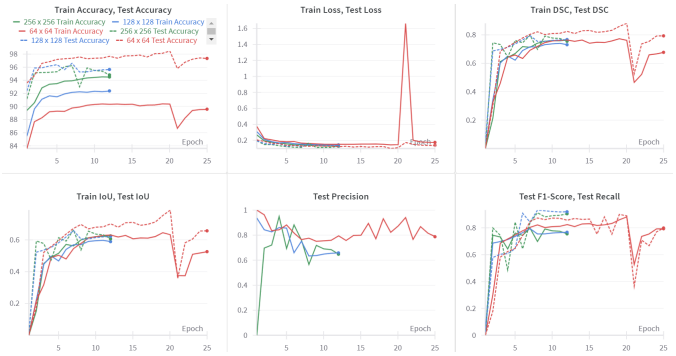

Fig. 10. Impact of Resolutions, click to view the live visualization: **Click here**

*1) **Performance Using 256x256 Resolution (Green Line)**:* The results for the 256x256 resolution was efficient and demonstrated a good balanced segmentation performance across all metrics. The outcomes were as follows:

- **Test Accuracy:** 94.82%, indicating strong generalization despite higher computational requirements.
- **Dice Similarity Coefficient (DSC):** 0.75518, showing consistent performance in pixel-wise accuracy.
- **Intersection over Union (IoU):** 0.60666, reflecting good overlap between predicted and ground truth masks.
- **Test Loss:** 0.12543, the lowest among all resolutions, indicating effective learning and strong generalization.
- **Test Precision:** 0.64801, suggesting a slight tendency toward over-segmentation with more false positives.
- **Test Recall:** 0.90482, ensuring thorough coverage and strong segmentation reliability.

The resolution stabilized in 12 epochs, showcasing quick convergence due to its ability to capture fine details. The resolution's ability to capture fine details makes it well-suited for tasks requiring high segmentation accuracy.

*2) **Performance Using 128x128 Resolution (Blue Line)**:* The 128x128 resolution delivered strong segmentation performance, balancing accuracy and efficiency. The results were as follows:

- **Test Accuracy:** 95.65%, slightly higher than the 256x256 resolution.
- **Dice Similarity Coefficient (DSC):** 0.76694, the highest among all resolutions.
- **Intersection over Union (IoU):** 0.62198, competitive with the 256x256 resolution.
- **Test Loss:** 0.13231, slightly higher than the 256x256 resolution, reflecting occasional training instability.
- **Test Precision:** 0.65791, indicating moderate segmentation accuracy with fewer false positives than 256x256.
- **Test Recall**: 0.91928, the highest recall across all resolutions, demonstrating the resolution's ability to identify most true positives.

This resolution achieved a balance between computational efficiency and segmentation performance, taking only 12 epochs to stabilize. While its precision was moderate, its high recall

and competitive IoU make it a dependable choice for activities that need accuracy.

*3) **Performance Using 64x64 Resolution(Red Line)**:* This resolution shows strong performance but experiences instability around the 22nd epoch, recovering well to demonstrate good segmentation capability. The results were as follows:

- **Test Accuracy:** 97.34%, the highest among all resolutions.
- Dice Similarity Coefficient (DSC): 0.79296, the best overall performance for DSC.
- **Intersection over Union (IoU):** 0.65695, higher than the 256x256 and 128x128 resolutions.
- **Test Loss:** 0.13690, slightly higher than both 128x128 and 256x256, indicating less effective generalization.
- **Test Precision:** 0.78738, the highest among all resolutions, showing minimal false positives.
- **Test Recall:** 0.79863, the lowest across resolutions, reflecting challenges in identifying true positives.

Although there was a spike in training loss that was observed around the 22nd epoch, this has little significance on impact of performance, as seen by the consistent test loss. This resolution stabilized after 25 training epochs. While it achieves high precision with minimal false positives, its lower recall and higher test loss limit its ability to capture finer details and broader segmentation coverage.

C. *Discussion*

The **256x256** resolution showed reliable segmentation quality, with strong DSC and IoU scores that highlight its ability to capture fine details. It stabilized after 12 epochs, making it efficient in convergence. It achieved the lowest test loss, proving to be dependable. In contrast, the 64x64 resolution required 25 epochs to stabilize, indicating slower convergence and didn't capture details as well, which was reflected in its lower IoU and recall scores. The 128x128 resolution works well as a middle ground. It balances efficiency and segmentation quality, with solid performance and reasonable resource use, although it sometimes showed instability during training. Higher resolutions were more efficient and delivered better segmentation reliability. In contrast, lower resolutions needed more resources and produced lower-quality results. The 128x128 resolution remains a practical middle ground.

D. *Conclusion*

This experiment confirms that resolution significantly influences both computing efficiency and segmentation performance. The **256x256** resolution stabilizes quickly and is excellent for critical applications like in this experiment, where the medical images require high segmentation accuracy and fine-grained detail capture. The **64x64** resolution with its high precision does still not suit tasks for medical image segmentations due to its low efficiency and low segmentation reliability. The **128x128** resolution offers a good balance between performance and efficiency. It performs better than 64x64 but doesn't quite match the results of 256x256. Future work could be to explore advanced techniques, such as the relation-based upsampling modules described in RCNet [10], to further optimize segmentation performance at lower resolutions.

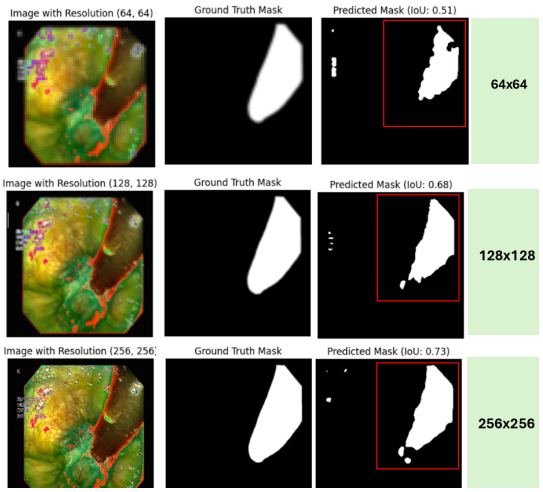

Fig. 11. Prediction Comparison - ground truth, and predicted masks of 64, 128 and 256

EXPERIMENT 2: DATA AUGMENTATION

This experiment involves a comparative analysis of how three different data augmentation methods (RandomRotation, ColorJitter, and Affine transformations) affect the performance of the U-net model. The primary goal is to evaluate the model's ability to generalize and accurately identify patterns by using Albumentations, a widely used augmentation library.

Data augmentations often play an important role in improving the performances of deep learning models specially when working with medical image analysis [11]. This experiment aims to determine whether these three augmentation techniques influence the model's segmentation accuracy positively or negatively.

A. *Methodology*

Albumentations is a fast and flexible library that offers a wide range of transformations that give better control over the augmentations compared to PyTorch's built-in library [12]. The key advantage of Albumentations is, it's Cython-based implementation that accelerates the augmentation process compared to PyTorch's slower Python-based approach. Making it ideal for large datasets where speed is important.

Augmentations chosen for the experiment [13]:

- Random Rotation: Randomly rotates the image by 90 degrees to improve model robustness to orientation variations.
- Color Jitter: Modifies the brightness, contrast, saturation, and hue of the images in order to simulate different lighting conditions and colour shifts that are common in real-world imaging scenarios.
- Affine Transformation: Improves the model's capacity to generalize through a variety of input types that simulate complex image distortions and translations, by using scaling, translation, rotation, and shearing.

B. Results

Metric	RandomRotation (Green)	ColorJitter (Blue)	Affine (Red)
Epoch	18	18	17
Test Accuracy	95.33174	93.33107	95.877
Test DSC	0.76171	0.51143	0.73995
Test F1-Score	0.76171	0.51143	0.73995
Test IoU	0.61513	0.34357	0.58724
Test Loss	0.12052	0.1626	0.10605
Test Precision	0.72658	0.80646	0.83179
Test Recall	0.80041	0.37445	0.66638
Train Accuracy	94.46914	93.72192	94.59015
Train DSC	0.68455	0.61943	0.67519
Train IoU	0.53307	0.46762	0.51958
Train Loss	0.1472	0.17112	0.13724

Fig. 12. Comparison of Augmentation Methods

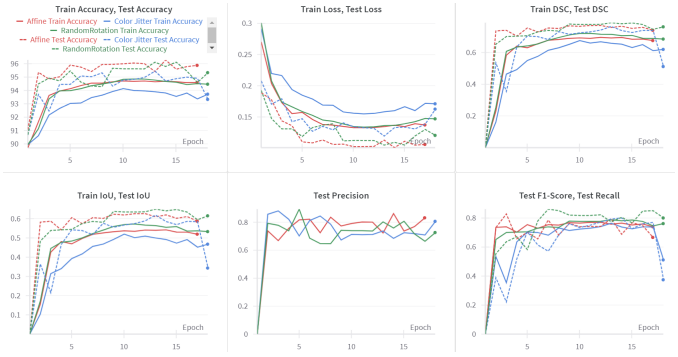

Fig. 13. Comparison of Performance Metrics Across Augmentation Techniques, click to view the live visualization **Click here**

Test and Train Accuracy:

- **RandomRotation(Green):** Achieved the highest test and train accuracy of 95.33% and 94.47%, proving that the rotation technique has helped the model to generalize across unseen data while continuing to maintain consistency during training.
- **AffineTransformations(Red):** Came in second place with results similar to RandomRotation, with a test accuracy of 95.88% and a train accuracy of 94.59%. These results indicate a strong generalization effect overall despite the added complexity due to scaling and translation.
- **ColorJitter(Blue):** Showed the lowest results of 93.33% and 93.72%. This is most likely due to the colour variations that introduced a high degree of noise to the data.

Dice Similarity Coefficient (DSC):

- **RandomRotation(Green):** Obtained the highest results of 0.7617, indicating that the model successfully recognized boundaries and distinguished between the foreground and the background.
- **AffineTransformations(Red):** Scored 0.7399, indicating that the model struggled slightly more with some complex variations.
- **ColorJitter(Blue):** Achieved the lowest results of 0.5114, clearly showing the model's difficulty in capturing boundary clarity due to increased colour variability.

Intersection over Union (IoU):

- **RandomRotation(Green):** Achieved the highest score of 0.6151, indicating strong overlap and resistance between the predicted and actual images.
- **AffineTransformations(Red):** Scored slightly lower at 0.5872, demonstrating the model's struggle with scaling and translation.
- **ColorJitter(Blue):** Scored the lowest at 0.3436, highlighting it's difficulty in alignment and overlap.

Precision and Recall:

- **AffineTransformations(Red):** Achieved the highest scores of 83.18% in precision and 66.64% in recall, showcasing it's strength in reducing false positives.
- **RandomRotation(Green):** Followed with scores of 72.65% (Precision) and 80.04% (Recall), showing it's ability to reduce false positives and segment boundaries effectively.
- **ColorJitter(Blue):** Scored the lowest with 80.64% (Precision) and 37.45% (Recall), showing some generalization across shapes but poor performance in maintaining a balance between true and false positives.

Loss:

- **AffineTransformations(Red):** Achieved the lowest test loss of 0.1061, indicating strong performance during predictions.
- **RandomRotation(Green):** Scored a slightly higher test loss of 0.1205.
- **ColorJitter(Blue):** Achieved the highest test loss at 0.1626, showing its difficulty in handling segmentation tasks effectively due to added colour variability.

C. Discussion

RandomRotation(Green) consistently showed the best segmentation performance with the highest Test DSC and IoU, which indicates that it has improved the model's ability to generalize across different orientations and aligns well with ground truth, offering accurate segmentation, as show in Fig.14. ColorJitter(Blue) provided balanced but also slightly reduced model's accuracy due to added variability, which introduced noise in predicted images and led to less precise predictions, especially in complex areas. Affine transformation(Red) scored the highest precision but low test loss, which indicates challenges in generalization due to overfitting. Despite the predicted image showing reasonable segmentation, there are still some struggles with finer details.

D. Conclusion

Overall, the findings show that choosing the right augmentation technique for the task and it's priorities is essential. This is because each of the techniques has distinct strengths and limitations. The best option to further enhance model performance and overall segmentation quality would be to combine different augmentation methods.

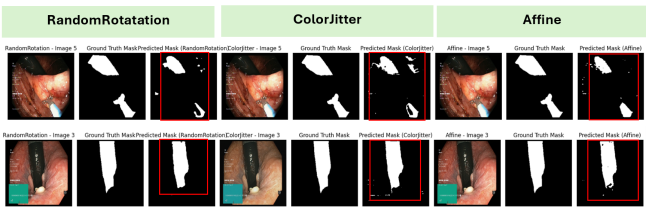

Fig. 14. Prediction Comparison - original, ground truth, and predicted masks

Experiment 3: Training Data Size

Assessing the impact of different training data sizes on a deep learning model's ability to segment surgical instruments in endoscopic images was the goal of the experiment. The model was trained and evaluated using the Kvasir-Instrument dataset, which comprises 472 annotated images. Three subsets were created from the dataset:

Dataset	Number of Images	Percentage of Total
Complete Data	472	100%
Fifty Percent of Data	236	50%
Twenty Percent of Data	94	20%

Fig. 15. Comparison of Dataset Sizes Used for Training and Evaluation

To guarantee consistent assessment, each dataset was divided into 80% training and 20% testing periods. Standard metrics were used to assess the model's performance after it had been trained on each subset.

A. Results

*1) **Performance Using 100% of Training Data:*** The model performed well on all evaluation metrics when it was trained on the entire dataset (472 images). The results were as follows:

- **Test Accuracy:** 95.76%, indicating that the model correctly classified the majority of pixels.
- **Dice Similarity Coefficient (DSC):** 0.7385, showing significant overlap between the predicted and true masks.
- **Intersection over Union (IoU):** 0.5855, reflecting strong pixel-level agreement.
- **Test Loss:** 0.1166, suggesting that the model minimized prediction errors effectively.
- **Test Precision:** 0.8678, showing high accuracy in identifying relevant pixels but not the highest among all datasets.
- **Test Recall:** 0.6428, highlighting the model's ability to identify true positives but with room for improvement in detecting smaller or subtler features.

Qualitative analysis of the predicted masks against the ground truth masks reveals a high degree of alignment, particularly for larger tools (see images for the 100% dataset). The IoU values for each image ranged from 0.08 to 0.91, demonstrating the model's capacity to produce precise and reliable segmentations.

*2) **Performance Using 50% of the Training Data:*** All metrics showed a decline in performance when the model was trained using only half of the data (236 images). The results were as follows:

- **Test Accuracy:** 94.75%, indicating a minor decline in classification performance.
- **Dice Similarity Coefficient (DSC):** 0.6364, indicating a decrease in segmentation quality.
- **Intersection over Union (IoU):** 0.4667, showing less overlap between the true and predicted masks.
- **Test Loss:** 0.1376, suggesting a higher difficulty in generalizing from the smaller dataset.
- **Test Precision:** 0.8995, the highest among all datasets, reflecting fewer false positives but at the cost of lower recall.
- **Test Recall:** 0.4923, indicating that the model missed more true positive regions, particularly for smaller or less noticeable tools.

In contrast to the 100% dataset, visual comparisons of the predicted masks and ground truth masks (see images for the 50% dataset) reveal more disjointed predictions and less accurate boundaries. Numerous predicted masks were unable to fully capture the tool regions, and the IoU values varied between 0.00 to 0.90, indicating that the model's performance was less consistent as the quantity of training data dropped.

*3) **Performance Using 20% of the Training Data:*** The outcomes were surprisingly good even when the model was only trained on 94 images or 20% of the total data. The results were as follows:

- **Test Accuracy:** 95.25%, close to the full dataset's accuracy.
- **Dice Similarity Coefficient (DSC):** 0.6981, slightly lower than the 100% dataset's result but higher than the 50% dataset's.
- **Intersection over Union (IoU):** 0.5362, indicating better overlap than in the 50% experiment.
- **Test Loss:** 0.1190, suggesting better generalization with the smaller dataset.
- **Test Precision:** 0.8572, lower than the 50% dataset but close to the 100% dataset.
- **Test Recall:** 0.5888, higher than the 50% dataset but lower than the 100% dataset.

Even though the boundaries of the predicted masks were less accurate than those of the full dataset, the model still produced good segmentation performance according to qualitative results for the 20% training data (see images for the 20% dataset). The IoU values, which varied from 0.07 to 0.92, showed that even in more difficult situations, the model could identify pertinent tool areas. The experiment's higher recall indicates that the model was able to concentrate more on

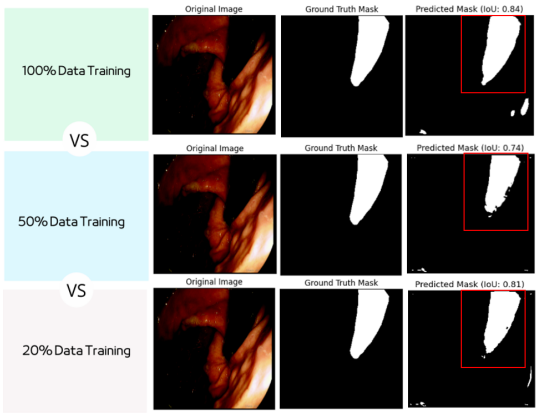

Fig. 16. Data Picture Impact Comparison

identifying true positives despite having less training data, though at the expense of slightly less precise boundaries.

B. *Discussion*

The outcomes unequivocally show that the quantity of training data has a major impact on the model's performance. The model achieved the highest Dice Similarity Coefficient (DSC) and IoU on the 100% dataset, as was to be expected. However, the model still performed well even with 50% and 20% less data, particularly in terms of accuracy and recall. The results for the 20% dataset showed relatively high Test Accuracy and Test Recall, surpassing the 50% dataset in Dice Similarity Coefficient (DSC) and Intersection over Union (IoU). This suggests that with less training data, the model generalized better in some respects, possibly due to overfitting tendencies in the 50% dataset.

Despite a moderate decline in IoU and DSC performance in the 50% training data experiment, the model was still able to identify the majority of the pertinent features. The higher Precision with 50% data (0.8995) shows a bias toward avoiding false positives, though this came at the cost of lower recall.

The 20% dataset experiment's outcome shows that deep learning models, especially those employing the U-Net architecture, can still function well with small datasets. This illustrates the robustness of the U-Net model when techniques for data augmentation are applied to increase the variability of the training data.

C. *Conclusion*

Despite having a small amount of training data, this experiment shows how much the U-Net architecture can segment medical images. The model trained on 20% of the dataset performed competitively compared to the models trained on 50% and 100% of the data. While the 100% dataset model showed the best results for Dice Similarity Coefficient (DSC), IoU, and Recall, the 50% dataset achieved the highest Test Precision, indicating that the model confidently avoided false positives. The 20% dataset model demonstrated competitive generalization and recall, suggesting that with smaller datasets, the model was able to focus more effectively on identifying true positive regions.

The results underscore the potential of deep learning models to perform well with small datasets, particularly when leveraging architectures like U-Net that are capable of strong generalization. Furthermore, this suggests that U-Net can be a valuable tool in clinical settings where labeled data is scarce, as it performs remarkably well even with significantly reduced data.

Metric	100% Training Data (Green)	50% Training Data (Blue)	20% Training Data (Red)
Epoch	19	22	9
Test Accuracy	95.75733	94.75483	95.25207
Test DSC	0.73853	0.63636	0.69806
Test F1-Score	0.73853	0.63636	0.69806
Test IoU	0.58545	0.46666	0.53617
Test Loss	0.11663	0.13756	0.11904
Test Precision	0.86782	0.89948	0.85715
Test Recall	0.64276	0.49234	0.58878
Train Accuracy	94.78857	93.95469	94.02424
Train DSC	0.71112	0.65694	0.67919
Train IoU	0.56393	0.50305	0.52158
Train Loss	0.13681	0.15759	0.15085

Fig. 17. Impact of Training Data Size Score

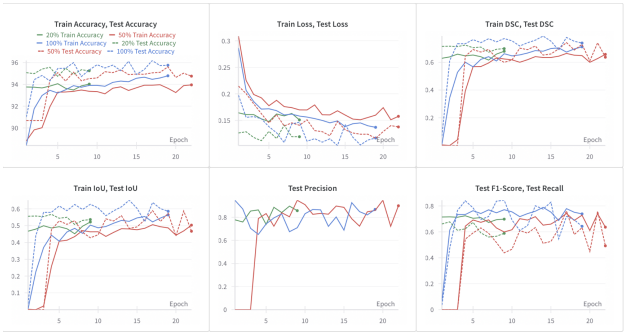

Fig. 18. Impact of Training Data Size, click to view the live visualization: **Click here**

Experiment 4: Optimizer Comparison

The goal of this experiment is to explore, compare the performances of different optimizers for the medical image segmentation and identify the best performing optimizer in the U-Net model. For this three optimizers have been selected Adam, SGD, and RMSprop on U-Net model performance for medical image segmentation using Kvasir-Instrument dataset

A. *Methodology*

All optimizers will be initiated with identical model parameters, trained with a consistent learning rate of 0.001, and a CosineAnnealingLR scheduler is employed for learning rate adjustments. Each optimizer's train and test metrics score is measured by how well it minimizes the loss and improves segmentation accuracy [14]:

Adam (Adam Gradient Descent): Adam computes the gradient and maintains moving averages of both gradients and squared gradients. It adjusts the learning rate and updates the model weights accordingly.

SGD (Stochastic Gradient Descent): SGD calculates the gradient of the loss for each training example and updates the model weights based on this gradient.

RMSprop (Root Mean Squared Propagation): RMSprop tracks the moving average of squared gradients, scales them, and updates the model parameters using these scaled gradients and the learning rate.

Finally, we summarize each optimizer's performance based on stability and segmentation accuracy to determine the most effective optimizer for this task [15].

B. Discussion

We looked into the three different optimizers to evaluate its performance: Adam, SGD and RMSp under the same configuration. The result shows that choosing the right optimizer has a high impact on the model's performance. Below is a summary of each optimizer experiment:

Metric	Adam (Green)	SGD (Blue)	RMSprop (Red)
Epoch	15	6	6
Test Accuracy	95.56749	90.67824	90.67824
Test DSC	0.74986	0.0	0.0
Test F1-Score	0.74986	0.0	0.0
Test IoU	0.59982	0.0	0.0
Test Loss	0.10917	0.29081	9.31106
Test Precision	0.79108	0.0	0.0
Test Recall	0.71273	0.0	0.0
Train Accuracy	92.84353	89.94629	89.94032
Train DSC	0.52079	0.0	0.00015
Train IoU	0.38155	0.0	8e-05
Train Loss	0.16619	0.31453	10.05688

Fig. 19. Optimizer Scores

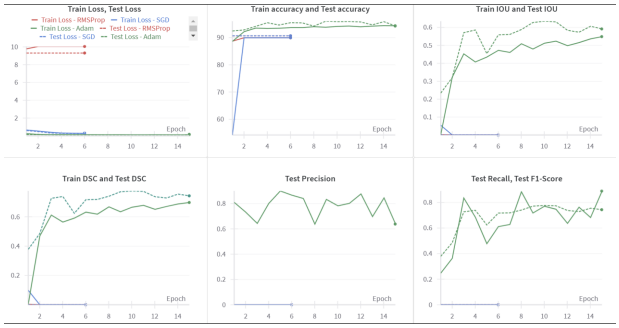

Fig. 20. Visualization of the optimizer experiment, click to view the live visualization **Click here**

Adam Optimizer (Green line): Reach the highest performance across most metrics, with test accuracy of 95.57%, DSC of 0.74986 and IoU of 0.59982 with the lowest test loss of 0.10917 compared to other optimizers, with precision at 0.79108 and recall at 0.71273 show a good maintain in terms of balance performance, likely because Adam's ability to adapt its learning speed helps the model learn accuracy and finding a good balance by avoiding the mistake and finding all correct answer.

SGD and RMSp: Both resulted in lower scores across all metrics:

- **SGD (Blue Line):** Given the result, we assume that the static learning rate in SGD made it difficult for the model to adapt to the complex segmentation task. This resulted in test accuracy around 90.68% with a higher test loss at 0.29, and failure to get a high score in DSC and IoU, suggesting underfitting [14].
- **RMSprop (Red Line):** While RMSprop is well-known for handling non-stationary data by adjusting the learning rate based on each parameter, it fails to learn on unseen test data, with the highest loss at 9.31%. However, train and test scores remained around 90.68%. This suggests that RMSprop made the model adjustments too large, causing it to miss important parts of the unseen data, so the model couldn't learn effectively. [15].

C. Random validation: Ground truth and predicted images on Kvasir Dataset

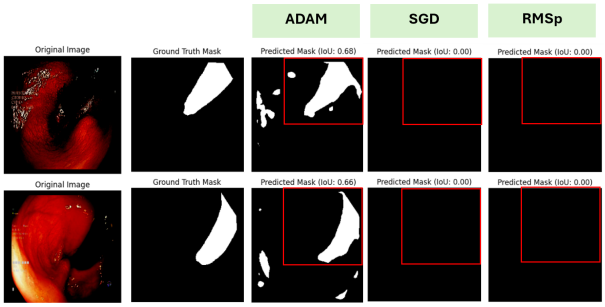

Fig. 21. Prediction Comparison - original, ground truth, and predicted masks

We performed random validation on 20 samples from the Kvasir testset to visually inspect the model's predictions alongside the true labels. This visualization provides insight into the model's performance with Adam optimizer corresponding with IoU scores from range between 0.14 - 0.92 showing its effectiveness in segmentation tasks and being able to separate the mask from the background. While the other two optimizers show 0.0 in IoU and DSC score, it fails to predict images with no output image to display, giving that because of their inability to segment images from unseen data.

D. Conclusion

In this experiment, the result proves that selecting the right optimizer is important as it can significantly impact the model performance. Among the options, Adam showed the best performance with the IoU and DSC score of 0.7, proving it to be the most balanced and effective optimizer for segmenting medical images in the Kvasir dataset. In contrast, SGD and RMPs's techniques failed with a poor score of f 0.0, likely

because of underfitting and failing to adapt to the complex data.

Therefore, the choice of optimization selection largely depends on the specific computer vision task and its configurations, such as the optimizer, learning rate and scheduler. For instance, experimental trials have shown that Cosine Annealing may not perform well with SGD and RMSp, while it works effectively with Plateau. Conversely, ADAM performs poorly when paired with ReduceLROnPlatea. Since this optimizers are sensitive to hyperparameters like learning rate and schedulers, we suggest on exploring different learning rate and schedulers, such as ReduceLROnPlateau or StepLR to address the underfitting and improve generalization to unseen data [15].

EXPERIMENT 5: ACTIVATION FUNCTIONS

The activation function is one of the most important elements when working with neural networks. Therefore, conducting research to pick the correct activation function is a must. This experiment investigates the impact of different activation functions on the final predictions and performance of the model. The objective here is to determine which of the chosen activation functions will produce the best segmentation based on the specified evaluation metrics.

A. *Methodology*

Three different instances of the U-NET model were created in notebook 6, one for each activation function to be tested, where the chosen activation functions are applied within the hidden convolutional layers of the U-NET model. This setup of the experiment is made to ensure efficiency while running the code and accuracy to compare the performance and outcome. Here are the chosen activation functions for this experiment and why they were chosen:

*1) **ReLU (Rectified Linear Unit)**:* ReLU is a common activation function used in the hidden layers as it is characterized by its simplicity and efficiency, as well as sparsity and ease of use, leading to a decrease in the vanishing gradient problem [16]. Even though this is one of the most popular activation functions in Deep Learning, it has its disadvantages. When the neuron only receives negative input values it will lead to the output “0”. This essentially means that the neuron is dead and no longer actively training [17].

*2) **LeakyReLU (Leaky Rectified Linear Unit)**:* LeakyReLU is a modified version of the ReLU activation function that is developed to address the challenge of “dead” neurons that ReLU suffers from [17]. This activation function is useful as it can prevent overfitting, and is capable of object recognition in images.

*3) **Tanh (Hyperbolic Tangent Function)**:* Tanh activation function is a common choice for classification task between two classes as it squashes the inputs to a range of [-1, 1]. This activation function is similar to the Sigmoid activation function, however this one is commonly used in the hidden layers of the U-NET model as it is zero-centered and helps prevent the vanishing gradient problem [18]. Tanh also has a tendency to saturate for large positive or negative values [19]. Limiting the model’s ability to learn effectively.

B. *Results*

20 samples from the Kvasir test dataset were used to perform random validation to inspect how well the model performs, and its ability to predict masks in comparison to the original true labels for every activation function experimented with. By visualizing the results, it makes it easier to assess and compare the different activation functions’ effect on performance. The table below shows the model performance based on the different activation functions tested, highlighting the importance of using the right activation function to obtain optimal results.

Metric	Leaky ReLU (Red)	Tanh (Blue)	ReLU (Green)
Epoch	30	18	24
Test Accuracy	95.98429	94.54467	96.00187
Test DSC	0.80748	0.67937	0.78156
Test F1-Score	0.80748	0.67937	0.78156
Test IoU	0.67713	0.51443	0.64144
Test Loss	0.097709	0.12868	0.10716
Test Precision	0.72995	0.75131	0.79637
Test Recall	0.90345	0.62	0.76729
Train Accuracy	95.72301	93.36602	95.03563
Train DSC	0.77413	0.63026	0.72614
Train IoU	0.63931	0.47117	0.58021
Train Loss	0.11391	0.16172	0.12944

Fig. 22. Activation Function Scores

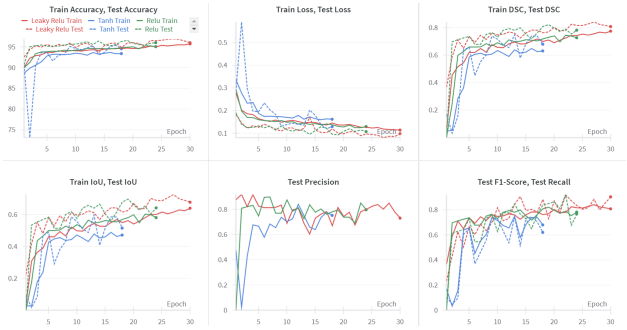

Fig. 23. Visualization of Activation Functions performance. For the live visualization, **click here**.

*1) **Performance with Relu** :* The ReLU activation function (Green line). demonstrated a strong and balanced performance across most metrics, it is considerably one of the most effective functions for this segmentation task as shown from the metrics scores below:

- **Test Accuracy:** 96%, indicating strong and reliable overall predictions.
- **Dice Similarity Coefficient (DSC):** 0.7816, a relatively high DSC score which reflects a good overlap between predictions and true masks.

- **Intersection over Union (IoU):** 0.6414, showing how well ReLU is performing on a pixel-level
- **Test Loss:** 0.1072, indicates effective error minimization.
- **Test Precision:** 0.7964, highlights how accurate ReLU identifies the relevant regions.
- **Test Recall:** 0.7673, shows great ability to capture the true positives.

ReLU's predictions varied, with IoU scores ranging from 0.10 at its worst, to 0.92 at its best, as can be seen in the figure below. The worst predictions struggled to separate the mask from the background, while the best case closely aligned with the ground truth, showing overall strong and balanced performance across all evaluation metrics.

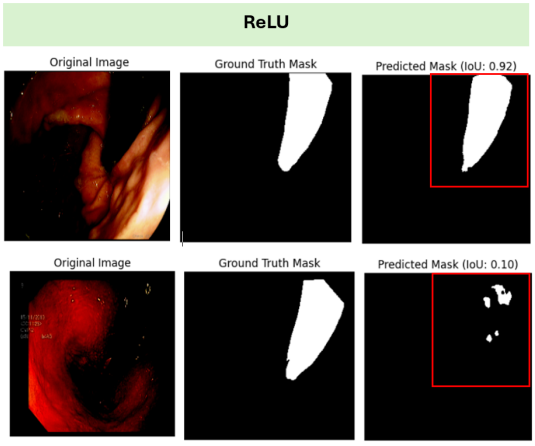

Fig. 24. Best and Worst predictions with ReLU

*2) **Performance with Tanh:** :* In comparison to ReLU, Tanh (Blue line) did not perform as well, as it scored lower across all metrics. This indicates that, for this particular task, Tanh may not be as effective.

- **Test Accuracy:** 94.54%,
- **Dice Similarity Coefficient (DSC):** 0.6794
- **Intersection over Union (IoU):** 0.5144
- **Test Loss:** 0.1287
- **Test Precision:** 0.7513
- **Test Recall:** 0.62

Tanh's effect on the U-NET model provided at its worst a prediction with an IoU of 0.03, which places the predicted mask far from the ground truth mask. The best prediction however, with an IoU of 0.86, showed relative similarity between the ground truth mask and its corresponding, predicted mask.

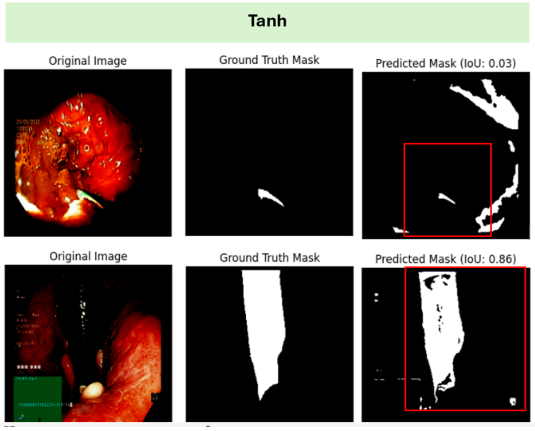

Fig. 25. Best and Worst predictions with Tanh

*3) **Performance with LeakyReLU:** :* While LeakyReLU (Red line) scores slightly lower on accuracy than ReLU did, it seems that it still performs effectively, as well as showcasing less test loss.

- **Test Accuracy:** 95.98%,
- **Dice Similarity Coefficient (DSC):** 0.8075
- **Intersection over Union (IoU):** 0.6771
- **Test Loss:** 0.0977, this suggests that LeakyReLU performs with less prediction errors than ReLU and Tanh.
- **Test Precision:** 0.7299
- **Test Recall:** 0.9034

Similar to tanh, the best recorded IoU score with LeakyReLU is 0.86, which shows close similarity between the predicted mask and the original ground truth mask. However, the prediction with the least similarity to the original ground truth mask shows an IoU score of 0.13. While the predictions were imprecise in some cases, it did show an overall effective performance on the majority of the predictions, as it showed close similarity to the ground truth mask, segmenting both small and large instrument regions.

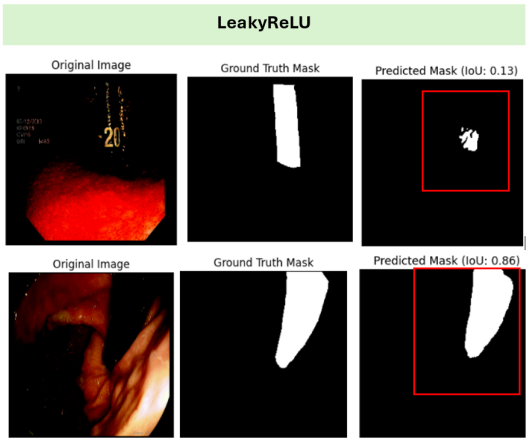

Fig. 26. Best and Worst predictions with LeakyReLU

C. *Discussion*

Among the three activation functions tested, ReLU seemed to outperform Tanh and LeakyReLU. LeakyReLU outperformed both ReLU and Tanh in several metrics, particularly in DSC and recall, Tanh underperformed across all metrics, making it the least effective activation function out of the three tested for this segmentation task. ReLU achieved a balance between precision and recall, ensuring that the performance stayed consistent across all metrics. It preserved high IoU and DSC scores, demonstrating how effective this activation function is for this image segmentation task. It seems however, that Tanh struggled with gradient saturation [19], which leads to a reduced quality in segmentation and poor predictions between predicted and true masks. Despite this, Tanh still performed reasonably at its best predictions. LeakyReLU showed promising performance based on the recall and DSC scores; it proved how well it can segment smaller instrument regions. However, based on the precision score, we can assume that it can occasionally "over-segment".

D. *Conclusion*

ReLU proved to be the most balanced and overall most fitting activation function for this image segmentation task, as it provided great scores across all metrics, and showed good predictions. LeakyReLU, while effective, may work better with tasks focusing on the smaller details. Tanh underperformed on most predictions, clearly highlighting limited performance in segmenting the medical images from the Kvasir Instrument dataset. These findings highlight the importance of selecting the correct activation functions based on the specific needs of the task. We opted to experiment with three different activation functions, as for this particular task, it provides enough data and diverse outcomes, making it easier to distinguish between which activation functions work best for image segmentation. However, limitations such as computational efficiency led to early stopping, potentially impacting performance and training progress. Future work could include exploring other popular and advanced activation functions such as Swish, ELU (Exponential Linear Unit), and Softmax to identify combinations that may lead to better results. Further testing with different loss functions in combination with activation functions could also provide additional insights.

IV. Limitations and Suggested Solutions

- **Choosing the best fit transformation technique:** One challenge we encountered was deciding between Pytorch transformation and Albumentation. When comparing both techniques under the same configuration, it proved that Albumentation did in fact improve the model's performance with different augmentation techniques. However, we decided to only use it exclusively in the augmentation experiment hence further research and experiments are needed to fully rely on the method [15].
- **Hyperparameter Sensitivity:** During experiments, choosing the appropriate learning rate proved to be quite challenging. During many trials, we found that a small number of learning rates can lead to slow convergence, while large numbers can lead to high loss scores, making comparisons difficult across optimizers. Our experiment demonstrated that even promising setups could fail, which suggests the need to experiment with different learning rate schedulers and optimizers to be able to find the best fit [15].
- **Addressing Artifacts and Preprocessing Issues:** In this experiment, we identified artifacts such as dark borders and unwanted padding in edge pixel values, as shown in Fig.27, which may interfere with model training and predictions. These issue could explain why, the 20 prediction images evaluated across all experiments, not all achieved accuracy above 0.92, with IoU scores ranging from 0.00 to 0.92. As CNNs requires properly scaled and consistent preprocessing, inconsistencies and improper scaling can degrade performance. James Deva Koresh(2023), in their study titled 'Impact of Preprocessing Steps in Deep Learning-Based Image Analysis.", emphasized on how preprocessing and artifact handling can significantly affect model accuracy and reliability [20]. Therefore, future work should focus on identifying the optimal preprocessing techniques to address scaling and artifacts issues, which could improve the model's performance and prediction accuracy.

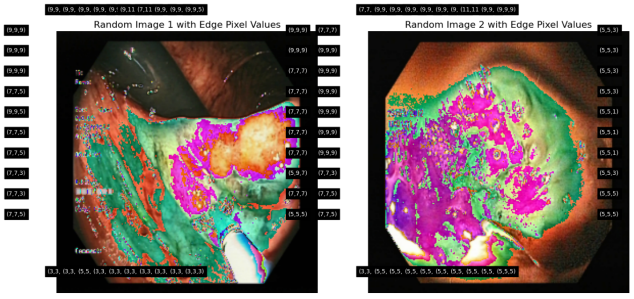

Fig. 27. Artifacts and Edge Padding Observed in Sample Predictions

V. Conclusion

The experiments conducted in this study provide valuable insights into how different factors influence the segmentation accuracy of the U-Net model. The findings highlight possible areas for improvement to create a more robust and reliable model for medical image segmentation. The U-NET model has proven to be an effective choice for image segmentation of the Kvasir Instrument dataset, providing valuable insight into how factors such as -image resolution, data augmentation techniques, training data size, optimization methods, and activation functions- impact the model's performance. Metrics such as Dice Coefficient (DSC), Intersection over Union (IoU), precision, recall, and accuracy provided a comprehensive evalutation of the model's performance and segmentation capabilities.

While the U-NET model has proven to be a good and robust choice for identifying medical instruments, there is

still room for improvements, as investigated in this study. Further refinement of deep learning models such as the U-NET model, show promising potential to enchance the reliability and generalization in clinical applications.

VI. Individual Contributions

Member	Responsibility
Kandidat nr. 15	Worked on Experiment 4. Collaborated with team members to integrate all analysis and Jupyter Notebooks. Contributed to the report by writing the section on Experiment 4 and refinding the report.
Kandidat nr. 17	Worked on Experiment 2, analyzing the impact of data augmentation techniques on U-Net's performance. Contributed to the creation of the outline and structure of the report and collaborated with the team members in writing and organizing the final report.
Kandidat nr. 18	Worked on Experiment 1, analyzing the impact of resolution. Joined in creating the master U-Net Jupyter Notebook and collaborated with the team members in writing and structuring the final report.
Kandidat nr. 33	Worked on Experiment 3, analyzing data training size impact. Furthermore I contributed to transferring the report to overleaf and collaborated with team members in writing and finalizing the report.
Kandidat nr. 42	Worked on Experiment 5, analyzing the impact of different activation functions. Collaborated with team members in writing and refining the report, as well as cited the sources in the IEEE format.

VII. Supplementary Materials

The experiments described in this report were conducted using Python and Jupyter Notebooks. The full implementation details, including code and outputs, are available at the following links:

U-Net Model (Base File): Notebook 1

Experiments:

Image resolution:	Notebook 2	WandB Link
Augmentation:	Notebook 3	WandB Link
Training size:	Notebook 4	WandB Link
Optimizer:	Notebook 5	WandB Link
Activation functions:	Notebook 6	WandB Link

References

[1] 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*, vol. 9351, Lecture Notes in Computer Science, Cham: Springer, pp. 234–241, 2015. doi: https://doi.org/10.1007/978-3-319-24574-4_28

[2] D. Jha *et al.*, "Kvasir-Instrument: Diagnostic and Therapeutic Tool Segmentation Dataset in Gastrointestinal Endoscopy," in *MultiMedia Modeling. MMM 2021*, vol. 12573, Lecture Notes in Computer Science, Cham: Springer, pp. 218–229, 2021. doi: https://doi.org/10.1007/978-3-030-67835-7_19

[3] PyTorch Tutorials, "Data Loading and Processing Tutorial," available: https://pytorch.org/tutorials/beginner/data_loading_tutorial.html.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *arXiv preprint arXiv:1511.00561*, 2015. doi: https://doi.org/10.48550/arXiv.1511.00561

[5] V. Pagare *et al.*, *Deep Learning with PyTorch: A Practical Approach to Building Neural Network Models Using PyTorch*, Birmingham, UK: Packt Publishing, 2018. available: https://github.com/yangyutu/bigfiles/blob/master/Vishnu%20Subramanian%20-%20Deep%20Learning%20with%20PyTorch-Packt%20(2018).pdf

[6] S. Sabater *et al.*, "Analysing the integration of MR images acquired in a non-radiotherapy treatment position into the radiotherapy workflow using deformable and rigid registration," *Radiotherapy and Oncology*, vol. 119, no. 1, pp. 179–184, 2016. doi: https://doi.org/10.1016/j.radonc.2016.02.032

[7] Y. Iwasa *et al.*, "Automatic segmentation of pancreatic tumors using deep learning on a video image of contrast-enhanced endoscopic ultrasound," *Journal of Clinical Medicine*, vol. 10, p. 3589, 2021. doi: https://doi.org/10.3390/jcm10163589

[8] H. Nunley *et al.*, "A novel ground truth dataset enables robust 3D nuclear instance segmentation in early mouse embryos," *bioRxiv*, vol. 1, no. 1, Mar. 2023. doi: https://doi.org/10.1101/2023.03.14.532646

[9] S. Umirzakova, S. Ahmad, L. U. Khan, and T. Whangbo, "Medical image super-resolution for smart healthcare applications: A comprehensive survey," *Information Fusion*, vol. 103, p. 102075, 2024. doi: https://doi.org/10.1016/j.inffus.2023.102075.

[10] J. Jiang *et al.*, "Super-resolution semantic segmentation with relation calibrating network," *Pattern Recognition*, vol. 124, p. 108501, 2022. doi: https://doi.org/10.1016/j.patcog.2021.108501

[11] X. Wang, L. Song, and Q. Zhang, "Hybrid deep learning approach for segmentation of medical images: A review," *Journal of Medical Imaging*, vol. 29, no. 5, pp. 251–259, 2022. doi: https://doi.org/10.48550/arXiv.2204.08610.

[12] E. D. Cubuk *et al.*, "AutoAugment: Learning Augmentation Strategies from Data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. doi: https://doi.org/10.48550/arXiv.1805.09501.

[13] F. Garcea *et al.*, "Data augmentation for medical imaging: A systematic literature review," *Journal of Medical Imaging*, vol. 7, no. 3, pp. 1–19, 2020. doi: https://doi.org/10.1016/j.compbiomed.2022.106391

[14] K. Ding, K. Ma, S. Wang, and E. P. Simoncelli, "Comparison of Full-Reference Image Quality Models for Optimization of Image Processing Systems," *International Journal of Computer Vision*, vol. 129, pp. 1258–1281, Apr. 2021. doi: https://doi.org/10.1007/s11263-020-01419-7

[15] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, Jun. 2017. doi: https://doi.org/10.48550/arXiv.1609.04747

[16] X. Wang *et al.*, "Smish: A novel activation function for deep learning methods," *Electronics*, vol. 11, no. 4, p. 540, 2022. doi: https://doi.org/10.3390/electronics11040540

[17] A. V. Morozov *et al.*, "Activation Functions in Neural Networks: Overview and Comparison," *Zhytomyr Polytechnic State University*, 2024. doi: https://doi.org/10.32782/2663-5941/2024.3.1/22.

[18] M. Zhang, "A Comparison of Image Classification with Different Activation Functions in Balanced and Imbalanced Datasets," Virginia Polytechnic Institute and State University, 2021. Available: https://vtechworks.lib.vt.edu/server/api/core/bitstreams/ccc922ed-0cf9-4947-9ee0-a8d7eced2899/content.

[19] C. C. Aggarwal, *Neural Networks and Deep Learning*, p.11, New York: Springer, 2018.doi: https://doi.org/10.1007/978-3-319-94463-0

[20] H. J. D. Koresh, "Impact of the Preprocessing Steps in Deep Learning-Based Image Classifications," *National Academy Science Letters*, Jan. 2024. doi: https://doi.org/10.1007/s40009-023-01372-2.