Experimental Study: Medical Image Segmentation on the Kvasir-Instrument Dataset.

Exam Group: 6
Candidates: 6, 21, 27, 31
School of Economics, Innovation, and Technology
Kristiania College University, Oslo, Norway

Abstract— **This paper presents an experimental study on medical image segmentation using the Kvasir-Instrument dataset. The primary focus is to evaluate the performance of a deep learning model by varying factors such as image resolution, data augmentation, optimization methods, and activation functions. The experiments employ various techniques, including learning rate scheduling and early stopping. The results are compared using metrics like the Dice Similarity Coefficient (DSC), Jaccard Index (IoU), precision, recall, and accuracy.**

Index Terms—**Medical Image Segmentation, Deep Learning, Kvasir-Instrument Dataset, Image Resolution, Data Augmentation, Optimization Methods, Activation Functions**

I. Introduction

Medical image segmentation plays a crucial role in diagnostic and therapeutic applications by enabling precise segmentation of anatomical structures of interest. In this study, we use the Kvasir-Instrument dataset, which contains 590 images and corresponding ground truth masks of endoscopic tools such as snares, balloons, biopsy forceps, etc. to run experiments aimed to explore key factors influencing the segmentation task and evaluate their impacts on the model's performance. The original resolution of the images in the dataset varies from 720x576 to 1280x1024. The image files are encoded using JPEG compression. The dataset comes with a pre-defined train/test split as 472/118 (80/20%) that was recommended to use.

According to the exam guidelines, each group member is required to select and conduct one experiment from the given 5 options. As a group of four, we decided to run experiments on the following topics.

- **Impact of Image Resolution on the Final Outcome.** The resolution of input images can significantly affect the performance of segmentation models. Higher resolutions may provide finer details but require greater computational resources, while lower resolutions may lose critical information. This study investigates the trade-offs associated with varying image resolutions, as discussed by Petersen et al. (2021).
- **Impact of Data Augmentation Techniques on Final Predictions** Data augmentation is crucial for enhancing the generalization ability of deep learning models, especially in medical imaging tasks where labeled data is often limited. By applying techniques such as those provided by the ImgAug and Albumentations libraries (Jung et al., 2020; Buslaev et al., 2020), this study evaluates how different augmentation methods affect model performance and robustness.
- **Impact of training data size on the final outcome.** We chose not to work on this experiment option.
- **Impact of Optimization Methods on the Final Outcome.** Optimization methods dictate how efficiently the model learns from the data. This study compares various optimization strategies, such as Adam, SGD and RMSProp, using the PyTorch optimization library (Paszke et al., 2019), to assess their impact on convergence and final accuracy.
- **Impact of Activation Functions on Final Predictions.** Activation functions play a critical role in introducing non-linearity to neural networks. This study evaluates the performance of different activation functions, such as ReLU, Sigmoid, and Tanh from the PyTorch framework (Paszke et al., 2019), to determine their effect on the segmentation predictions.

Through these experiments, this study aims to provide insights into optimizing medical image segmentation tasks and understanding how various factors influence the performance of deep learning models.

II. Methodology

In order to perform the experiments required in the exam instructions. We decided to work with the DeepLabV3+ model as we anticipated that the other groups would be working with U-Net model.

DeepLabV3+ is an advanced model for semantic image segmentation that uses atrous convolutions to capture multi-scale context while maintaining spatial resolution. The architecture also includes an encoder-decoder structure with a spatial pyramid pooling module, making it effective at segmenting objects of different sizes (Chen at al., 2018).

Model Definition

The DeepLabV3PlusWithSigmoid class is defined by extending nn.Module and initializing the DeepLabV3+ architecture with the specified encoder and weights. The output of the

DeepLabV3+ model is passed through a Sigmoid function to ensure binary classification.

Table I illustrates the tabular representation of the model architecture, showing the different layers and their parameters:

TABLE I
MODEL ARCHITECTURE AND PARAMETERS FOR DEEPLABV3+ WITH SIGMOID

Layer Name	Output Shape	Number of Parameters
Input Layer	(Batch Size, 3, H, W)	0
ResNet50 Encoder	(Batch Size, 2048, H/32, W/32)	23,508,544
ASPP (Atrous Spatial Pyramid Pooling)	(Batch Size, 256, H/32, W/32)	6,642,688
Decoder (Upsampling)	(Batch Size, 256, H/8, W/8)	2,097,408
Final Convolution Layer	(Batch Size, 1, H/8, W/8)	257
Sigmoid Activation	(Batch Size, 1, H/8, W/8)	0
Total Parameters		**32,509,897**

- **Input Layer**: The input to the model consists of images with 3 channels (RGB), and the height and width are dependent on the resolution of the input image.
- **ResNet50 Encoder**: The backbone of the model is a pre-trained ResNet50 model, which extracts features from the input image.
- **ASPP (Atrous Spatial Pyramid Pooling)**: This module captures contextual information at multiple scales, improving the model's ability to segment objects at different sizes.
- **Decoder (Upsampling)**: The decoder upsamples the feature map to the original image size, producing pixel-wise predictions for segmentation.
- **Final Convolution Layer**: This layer reduces the number of channels to 1, making the output suitable for binary segmentation.
- **Sigmoid Activation**: The Sigmoid function is applied to the final output to map the results to a probability between 0 and 1, indicating whether each pixel belongs to the foreground or background.

In addition to the tradition Loss and Accuracy.

$$\text{Average Train Loss} = \frac{\sum_{i=1}^{N} \text{Train Loss}_i}{N}$$

$$\text{Average Train Accuracy} = \frac{\sum_{i=1}^{N} \text{Train Accuracy}_i}{N}$$

We evaluated the performance of the segmentation model by using several other standard metrics:

- The Jaccard index measures the overlap between predicted and true positive pixels.

$$\text{IoU} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive} + \text{False Negative}}$$

- Precision and Recall assess the model's ability to identify relevant regions and avoid false negatives, respectively.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- Accuracy calculates the overall correctness of the predictions.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}}$$

- Finally, the Dice Coefficient is used to evaluate the balance between precision and recall

$$\text{Dice} = \frac{2 \times \text{True Positive}}{2 \times \text{True Positive} + \text{False Positive} + \text{False Negative}}$$

Each of the team member used this model to run the experiments on varying image resolutions, data augmentation techniques, optimization methods, and activation functions.

A. Experiment 1: Impact of Image Resolution on Final Outcome.

For the "Resolution Experiment", the model is tested on three different resolutions: 64x64, 128x128, and 256x256. For each resolution, the images undergo a series of transformations to make them compatible with the model's input requirements. First, the images are resized to a specific resolution depending on the current experiment. This resizing is crucial for accommodating the model's expected input size at each resolution and allows the model to handle different levels of image detail.

After resizing to these different resolutions, the images are then transformed again to ensure they meet the exact input specifications of the model, particularly the final 256x256 size expected by the DeepLabV3+ model. These transformations include additional operations such as normalization to ensure that pixel values are within an appropriate range for model training.

This experiment investigates how different image resolutions affect the model's performance. Lower and higher resolutions are used to train the segmentation model, and results are compared based on key metrics (Loss, Accuracy, Jaccard Index, Dice Coefficient, Precision, and Recall).

B. Experiment 2: Impact of Data Augmentation Techniques on Predictions.

For the "Augmentation Experiment," the model is tested with three different data augmentation techniques—horizontal flipping, brightness/contrast adjustment, and Gaussian blur applied to the training data. These augmentations are designed to increase the diversity of the dataset and evaluate the model's ability to generalize to different conditions.

Each augmentation technique involves a specific transformation applied to the input images and their corresponding ground truth masks during the preprocessing step:

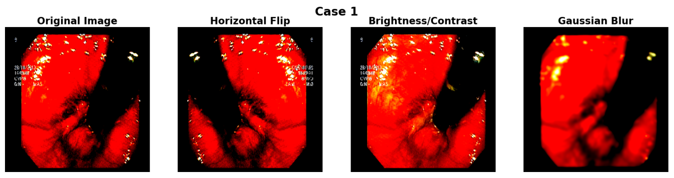

Fig. 1. The displayed images include the original image, a horizontally flipped version, a version with brightness and contrast, and a version with Gaussian blur.

- **Horizontal Flip:** The images and masks are flipped horizontally with a 50% probability. This augmentation introduces mirrored views of the surgical instruments, challenging the model to adapt to variations in orientation (TensorFlow, 2024).
- **Brightness/Contrast Adjustment:** The brightness and contrast of the images are adjusted randomly with a 20% probability. This transformation simulates lighting variations that might occur during endoscopic procedures, helping the model become robust to non-uniform lighting conditions (PyTorch, 2024).
- **Gaussian Blur:** A Gaussian blur with a kernel size between 3 and 7 is applied to the images with a 20% probability. This simulates real-world scenarios where images may appear blurry due to motion or focus issues (TensorFlow, 2024).

For all experiments, the original images are resized to 256x256 to ensure compatibility with the input requirements of the DeepLabV3+ model.

Each augmentation technique is applied individually, allowing us to isolate its effect on the model's performance. The augmented dataset is then used to train the model, and the results are evaluated using metrics such as Test Loss, Accuracy, Jaccard Index, Precision, Recall, and Dice Similarity Coefficient (DSC). The baseline results (with no augmentation) serve as a control to compare the improvements provided by each augmentation.

C. Experiment 4: Impact of Optimization Methods on Outcome.

Experimenting with different optimizers in machine learning is crucial to understanding their impact on model training and final performance. Optimizers control how model parameters are updated during training, directly influencing convergence speed, stability, and generalization ability.

To achieve this understanding, we tested the performance of the base model with three different optimizers.

Stochastic Gradient Descent (SGD) is widely used for its simplicity and computational efficiency, especially with large-scale datasets. However, it suffers from slow convergence and sensitivity to the learning rate. Variants such as SGD with momentum address these issues by smoothing parameter updates and reducing oscillations in the optimization trajectory. This enhancement helps in stabilizing the training process (Tanwani et al., 2023).

In addition to SGD, we also experimented with Adam and RMSProp, two popular adaptive optimizers. Both dynamically adjust the learning rate based on gradient moments, making them effective for handling sparse or noisy gradients. Adam, in particular, combines the advantages of momentum and adaptive learning rates, often leading to faster convergence for deep networks (Kingma, 2015, Loshchilov, 2019). RMSProp, on the other hand, excels in controlling the learning rate decay, which is beneficial for non-convex optimization problems (Zhang, 2023).

Different optimization methods, such as Adam, SGD, and RMSProp, are used to train the model, and their effects on convergence and final performance are analyzed.

D. Experiment 5: Impact of Activation Functions on Predictions.

Activation functions are essential components of neural networks, it introduces non-linearity into the model, enabling the model to learn complex patterns. Without activation functions, the neural network would essentially behave like a linear regression model, no matter how many layers are added (Goodfellow et al., 2016). This non-linearity is crucial because real-world data often exhibit complex, non-linear relationships. Activation functions allow neural networks to approximate these relationships.

The role of activation functions extends beyond just introducing non-linearity. They also affect the gradient propagation during training. For example, the choice of activation function can determine how effectively gradients are backpropagated through the network (Bengio et al., 2017). This is important because poor gradient flow can lead to problems like vanishing or exploding gradients, particularly in very deep networks. Thus, the choice of activation function can influence both the learning dynamics and the overall performance of the model.

We experimented with ReLU, Sigmoid, and Tanh activation functions.

- **ReLU (Rectified Linear Unit)** ReLU has become one of the most popular activation functions due to its simplicity and efficiency. It outputs the input directly if it is positive; otherwise, it returns zero. This function mitigates the vanishing gradient problem by allowing gradients to flow through the network more effectively (Nair & Hinton, 2010). ReLU also introduces sparsity into the activations, which can improve computational efficiency and reduce the likelihood of overfitting. However, it has its limitations, such as the ”dying ReLU” problem, where neurons can get stuck and stop learning if the input is negative.
- **Sigmoid** The sigmoid activation function maps input values to a range between 0 and 1, making it particularly useful for binary classification problems. It has been widely used in the past, especially in the output layer of a neural network for probabilistic predictions (Rumelhart et al., 1986). However, it suffers from a problem known as the vanishing gradient, where gradients become very small for large input values, slowing down learning.
- **Tanh (Hyperbolic Tangent)** The tanh activation function is similar to the sigmoid but maps input values to a range between -1 and 1. It is often preferred over sigmoid in practice because its output is centered around zero, leading to more efficient training (Bengio et al., 2017). However, like the sigmoid, tanh suffers from the

vanishing gradient problem for large input values, making it less suitable for very deep networks.

Evaluating the results; while Loss gives an indication of the model's optimization progress, and Accuracy, Precision, and Recall shed light on specific aspects of prediction, it is important to consider which metric most effectively highlights activation functions on segmentation quality, that is the Jaccard Index. Therefore, the analysis of experimentation results is based on Jaccard Index.

III. Results and Discussion

In this section, we present and discuss the results of the experiments, with each subsection focusing on a specific experiment.

To assess model performance, we utilized several evaluation metrics: Loss, Accuracy, Precision, Recall, Dice Coefficient, and Jaccard Index. Each of these metrics offers valuable insights into different aspects of the model's performance. While Loss indicates the model's optimization progress, Accuracy, Precision, and Recall provide a closer look at the model's prediction accuracy. The Dice Coefficient and Jaccard Index, on the other hand, are particularly useful for evaluating the performance of models involved in image segmentation tasks.

In Experiment 1, we will discuss the results using all of these evaluation metrics. For the subsequent experiments, the model's performance will primarily be assessed using the Jaccard Index.

A. Results for Experiment 1: Image Resolution

The comparison of evaluation metrics across the three different resolutions is presented in the following sections. Below are some key observations and interpretations.

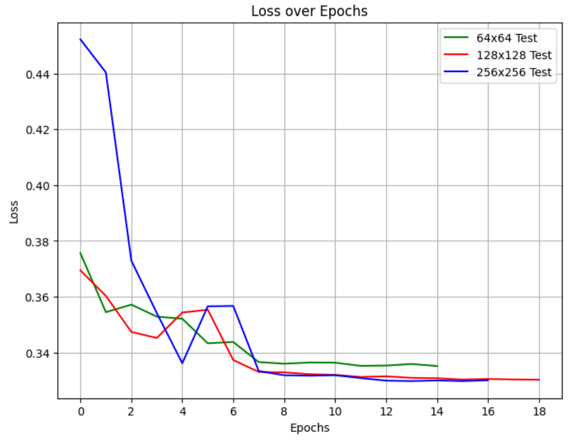

Fig. 2. Loss Function Over Resolutions

- **Loss Function:** The loss curves in Fig. 2 show how well the model minimized errors during training. For all three resolutions, the loss decreases steadily, with the higher resolution (256x256) initially showing a sharper decline, which suggests faster convergence in the early epochs. However, by the later epochs, all three resolutions reach comparable loss values before early stop mechanism kicked in at epoch 14, 18 and 16 respectively. This is a clear indicator that the model eventually stabilizes regardless of resolution.

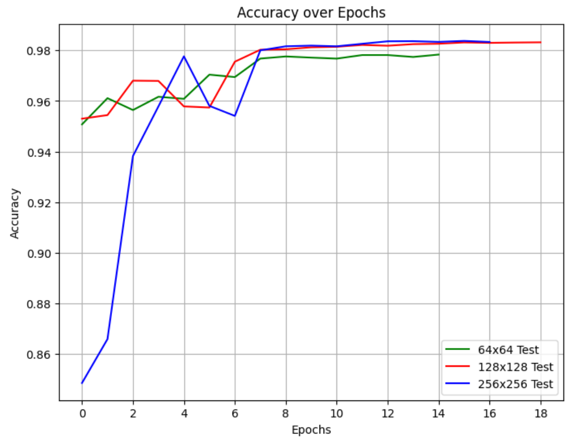

Fig. 3. Accuracy Over Resolutions

- **Accuracy:** The accuracy graph illustrates the model's ability to make correct predictions in Fig. 3. While all resolutions exhibit an increase in accuracy over time, the 256x256 resolution shows a rapid improvement in the early epochs, surpassing 0.95 accuracy within the first five epochs. However, this rapid rise plateaus, and the final accuracy values for all resolutions converge to similar levels, indicating that lower resolutions (64x64 and 128x128) can achieve similar performance with more epochs.

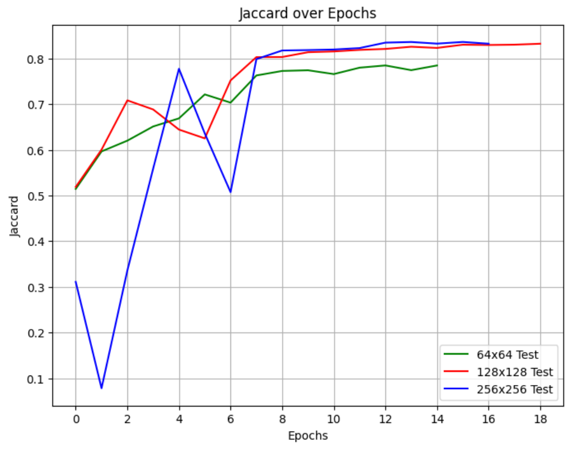

Fig. 4. Jaccard Index Over Resolutions

- **Jaccard Index:** The Jaccard index measures overlap between predicted and actual regions. Fig. 4 shows that higher resolutions (256x256) initially perform worse compared to the lower resolutions (128x128 and 64x64).

However, as the training progresses, the 256x256 resolution catches up and even surpasses the others in some cases. The 128x128 resolution appears to achieve stable Jaccard values more consistently across epochs.

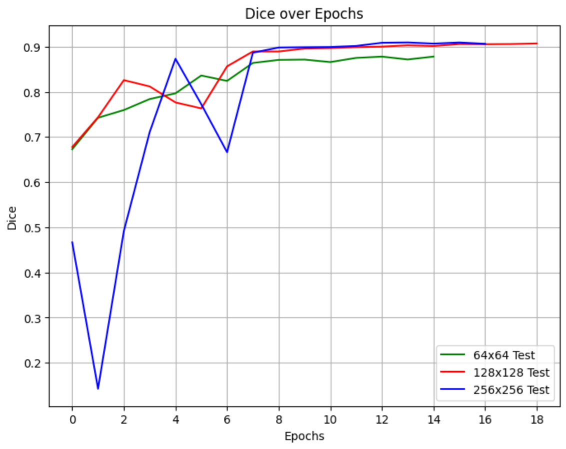

Fig. 5. Dice Coefficient Over Resolutions

- **Dice Coefficient over Epochs:** The Dice coefficient, another measure of overlap, behaves similarly to the Jaccard index. As illustrated in Fig. 5, the 256x256 resolution experiences instability early on, but it later stabilizes and achieves comparable or better results compared to the other resolutions. The 128x128 resolution maintains consistent performance throughout, indicating that it might provide a good trade-off between computational efficiency and predictive accuracy.

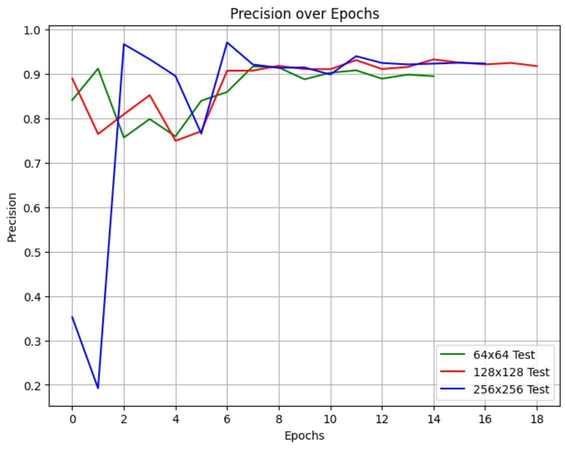

Fig. 6. Recall Score Over Resolutions

- **Precision over Epochs:** Precision score reflects the ability to avoid false positives, remains high across all resolutions. The 256x256 resolution again shows some initial instability but quickly stabilizes. The 64x64 and 128x128 resolutions exhibit steadier progress, with final values comparable across all resolutions as shown in Fig. 6 .

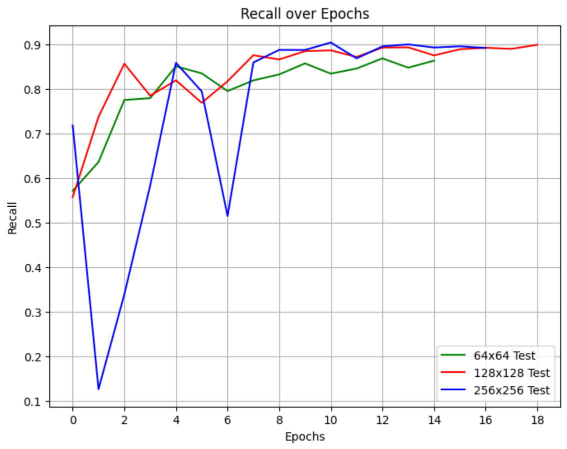

Fig. 7. Loss Function Over Resolutions

- **Recall over Epochs:** Recall score measures the ability to capture true positives. Lastly, Fig. 7 shows notable variability in the early epochs for the 256x256 resolution. However, it stabilizes as training progresses, achieving similar final values to the other resolutions. Both 64x64 and 128x128 maintain more consistent recall values across epochs.

Overall Observations:
Higher resolutions, such as 256x256, exhibit faster convergence and higher initial metric improvements but tend to experience instability in the early epochs. Lower resolutions, such as 64x64 and 128x128, are more stable and achieve comparable performance after more epochs. The 128x128 resolution, in particular, emerges as a balanced choice, providing both stability and good performance without requiring the computational resources of 256x256.

These results suggest that while higher resolutions can accelerate early convergence, their benefits diminish as training progresses. Lower resolutions, especially 128x128, offer a practical alternative, achieving nearly identical results with reduced computational costs. This highlights the importance of selecting an appropriate resolution based on the specific constraints and goals of the task.

The model's prediction ability improves with increasing resolution, capturing finer details and more accurate boundaries. At 64x64, predictions are coarse, suitable for capturing general shapes but lacking precision. At 128x128, segmentation improves, balancing detail and computational efficiency. The 256x256 resolution provides a significant leap in accuracy, effectively capturing intricate structures with precise boundaries. At 512x512, predictions are highly detailed but computationally expensive, offering diminishing returns compared to 256x256. Overall, 128x128 provides the best trade-off between accuracy and efficiency. At this resolution, the model can predict finer details and boundaries improves significantly over the 64s64 while not significantly lag behind the 512x512 resolution in terms of performance.

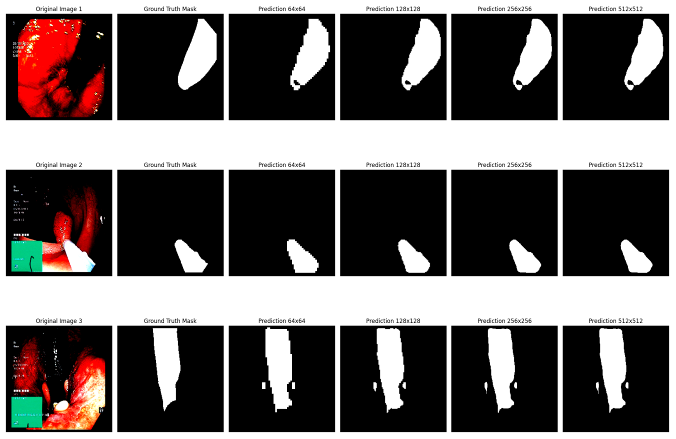

Fig. 8. Model's Prediction Behavior Based on Resolutions

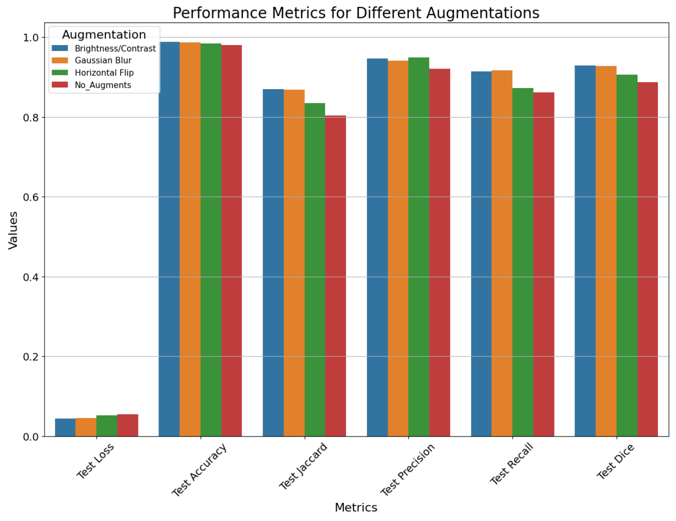

Fig. 9. Performance Metrics Comparison for Different Augmentations. Metrics include Test Loss, Test Accuracy, Jaccard Index, Precision, Recall, and Dice Similarity Coefficient.

B. Results for Experiment 2: Data Augmentation

In Experiment 2, we examined the effects of three data augmentation techniques: horizontal flipping, brightness/contrast adjustment, and Gaussian blur on the performance of the segmentation model. Table II presents the best results obtained for each augmentation technique, along with the baseline (no augmentation), evaluated using key metrics: Test Loss, Test Accuracy, Jaccard Index, Precision, Recall, and Dice Coefficient.

TABLE II
PERFORMANCE METRICS FOR DIFFERENT AUGMENTATION TECHNIQUES

Augmentation	Epoch	Test Loss	Test Acc.	Jaccard	Precision	Recall	Dice
No Augmentation	7	0.054853	0.9803	0.8035	0.9200	0.8612	0.8872
Horizontal Flip	13	0.051981	0.9840	0.8339	0.9485	0.8717	0.9054
Brightness/Contrast	13	0.043579	0.9873	0.8696	0.9464	0.9134	0.9285
Gaussian Blur	4	0.045664	0.9871	0.8682	0.9411	0.9164	0.9275

Figure 9, provides a visual comparison of the same metrics, allowing for a clearer understanding of the augmentation techniques' relative performance.

Baseline Performance: The baseline model is trained without any augmentations and achieved an accuracy of 0.9803, a Dice Similarity Coefficient (DSC) of 0.8872, and a Jaccard Index (IoU) of 0.8035 on the test set. These values provided a benchmark against which the augmented models could be compared.

- **Horizontal Flip**: The model trained with horizontal flipping showed a decrease in IoU (0.8339) compared to the baseline, but an increase in DSC (0.9054). Precision and recall increased, suggesting that while the augmented model captured the structure of the images well, it slightly struggled with images in flipped orientations.
- **Brightness/Contrast Adjustment**: This augmentation yielded a modest improvement over the baseline, with a DSC of 0.9285 and an IoU of 0.8696. Precision and recall saw substantial increases, indicating that the model benefited from exposure to varied lighting conditions, which enhanced its ability to segment in non-uniform lighting.
- **Gaussian Blur**: Gaussian blur showed slight gains, with a DSC of 0.9275 and an IoU of 0.8682. The improvements in precision and recall suggest that this augmentation helped the model handle less sharp images, likely simulating real-world conditions where images may lack clarity.

Comparison with Baseline: Table II summarizes the results. All three augmented models showed improved performance over the baseline, with Gaussian Blur and Brightness/Contrast providing the most significant gains. The results indicate that data augmentation can enhance model robustness, allowing it to generalize better to various scenarios and maintain performance even when the input data is varied.

Visual Analysis of Data Augmentation Techniques: Figure 10 illustrates the qualitative impact of each augmentation technique, showing the original image, ground truth mask, augmented ground truth mask, model prediction, and overlay prediction. Below are the observations and implications for each technique:

- **Horizontal Flip**
  - **Observation:** The overlay prediction for the horizontal flip augmentation (Figure 10, row 1) shows a more accurate boundary compared to the other two, but some slight instability can be seen along the edges of the image where the flipping of the image could have caused some spatial discrepancies during training, confusing the model.
  - **Implication:** Horizontal flipping shows the object mirrored, giving the model a variety of differently oriented images. The stable performance compared to the other two showed that this technique is much better at preserving key structural features while

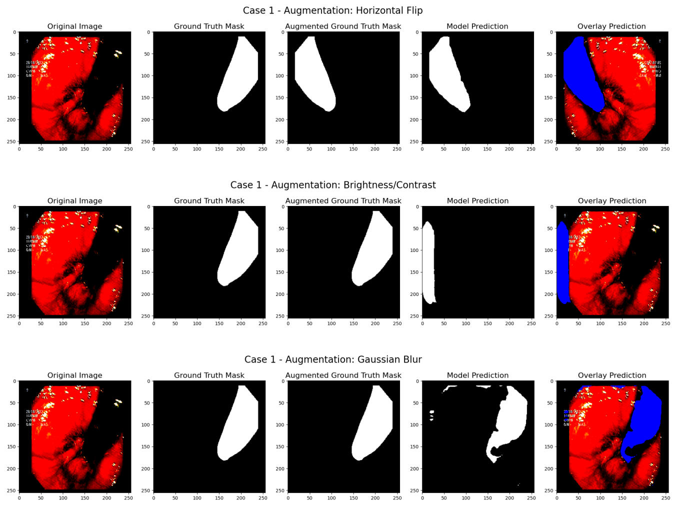
Case 1 - Augmentation: Horizontal Flip
Original Image
Ground Truth Mask
Augmented Ground Truth Mask
Model Prediction
Overlay Prediction
Case 1 - Augmentation: Brightness/Contrast
Case 1 - Augmentation: Gaussian Blur

Fig. 10. Augmentations showing Columns: Original Image, Ground Truth Mask, Augmented Ground Truth Mask, Model Prediction, Overlay Prediction. Rows correspond to Horizontal Flip, Brightness/Contrast, and Gaussian Blur augmentations, respectively.

improving generalization.

- **Brightness/Contrast Adjustment**
  - **Observation:** Here (Figure 10, row 2) shows the brightness augmentation in action. The augmented ground truth and the model prediction highlight some difficulties the augmentation faced. The brighter regions in the original image seem to have dominated the prediction, influencing the model's ability to segment the details consistently.
  - **Implication:** The brightness/contrast adjustment allows us to vary illumination, helping the model generalize to real-world conditions. However, the overlay highlights a potential problem: the augmentation seems to overemphasize some features (e.g., bright spots), leading to errors in segmentation.
- **Gaussian Blur**
  - **Observation:** Gaussian blur had a significant effect on the overlay prediction (Figure 10, row 3). The blurred regions led to the model missing or over-segmenting object boundaries, particularly in areas with lower contrast. This was especially evident in the smoothed areas where the model failed to segment the object correctly.
  - **Implication:** Gaussian blur, which simulates scenarios with motion or focus issues, does improve the model's robustness. However, the prediction abnormalities indicate that this comes at the cost of introducing uncertainty in accurately pinpointing object boundaries.

Overall Observations:

Data augmentation techniques demonstrate distinct benefits in improving segmentation model performance, with each augmentation showing unique strengths. The trends observed

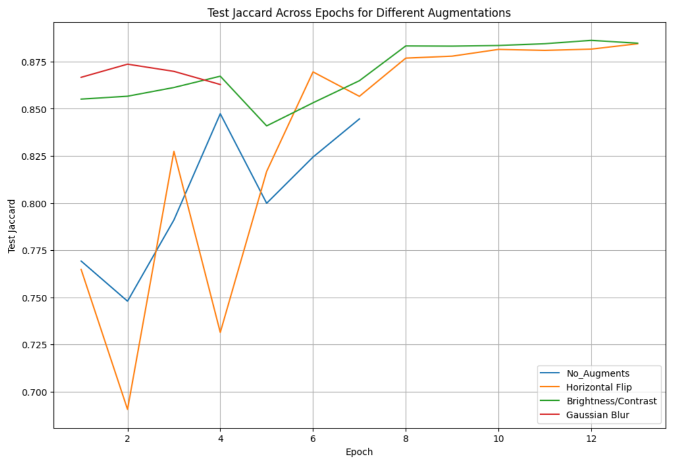

Fig. 11. Jaccard Index Over Epochs

in Figure 11 helps to highlight these differences:

- **Baseline (No Augmentation):** The baseline model, trained without any augmentations, consistently showed the lowest Jaccard Index across all epochs. Its slower improvement and limited performance shows the importance of augmentations in enhancing the dataset. By introducing variability, augmentations significantly improved the model's ability to generalize to new data.
- **Horizontal Flipping:** Horizontal flipping exhibited more erratic performance compared to the other augmentations. Its Jaccard Index curve showed significant fluctuations, particularly in the earlier epochs, before stabilizing in later epochs. While it did surpass the baseline (No Augmentation), its improvements were more modest, possibly due to the positional dependency of the mask, which may have negatively impacted the training process. Horizontal flip still contributed to generalization and showed reasonable performance in later epochs.
- **Gaussian Blur:** Gaussian blur was one of the best-performing augmentations, maintaining high Jaccard Index values across most epochs. While its curve exhibited some fluctuations, it consistently outperformed other techniques in later epochs. This augmentation helped the model generalize better to less sharp images, leading to improved segmentation performance. The strong results highlight its effectiveness.
- **Brightness/Contrast Adjustment:** Brightness/contrast adjustment produced strong and steady results. This technique improved the model's ability to handle varying lighting conditions. Despite some minor variations, its consistent performance demonstrated its reliability. The graph shows its utility as a versatile augmentation strategy. Figure 11 highlights its reliability as an augmentation technique.

The results in Figure 11 demonstrate that Gaussian blur and brightness/contrast adjustments are the most effective techniques, offering significant improvements in Jaccard Index and strong generalization performance across epochs. Horizontal flipping, though more erratic and less impactful, still adds value to the overall model performance, particularly in later epochs. These findings help to highlight the importance of carefully selecting augmentation strategies that align with the dataset characteristics, and task requirements to maximize segmentation performance.

C. Results for Experiment 4: Optimization Methods

In this experiment, we evaluated the impact of three optimizers—Adam (Adaptive Moment Estimation), SGD (Stochastic Gradient Descent), and RMSProp (Root Mean Square Propagation)—on the performance of the DeepLabV3+ segmentation model. The optimizers were compared based on their ability to minimize loss, achieve high accuracy, and produce quality segmentation results. Metrics such as the Jaccard Index, Dice Coefficient, Precision, and Recall were used to evaluate the segmentation quality. The main goal of this experiment was to determine which of the selected optimizers provided the best convergence speed, stability, and segmentation accuracy.

TABLE III
PERFORMANCE METRICS FOR DIFFERENT OPTIMIZERS

Optimizer	Test Loss	Test Acc.	Jaccard	Precision	Recall	Dice
Adam	0.0360	0.9884	0.8831	0.9467	0.9280	0.9365
SGD	0.0386	0.9869	0.8665	0.9271	0.9290	0.9276
RMSProp	0.0328	0.9890	0.8889	0.9468	0.9335	0.9394

- **Adam**
  - **Losses:** Adam showed fast convergence in training and testing losses, achieving a test loss of 0.0360. However, slight fluctuations in loss values during the mid-training epochs suggest instability before final stabilization.
  - **Accuracy:** The test accuracy reached 98.84%, demonstrating a balance between training and testing performance with minimal overfitting.
  - **Metrics:** Adam achieved a Jaccard Index of 0.8831 and a Dice Coefficient of 0.9365. Its Precision (94.67%) and Recall (92.80%) indicate a good balance between avoiding false positives and capturing true positives. Figure 12.
- **SGD**
  - **Losses:** SGD's final test loss reduced to 0.0386, which is higher compared to Adam and RMSProp. It demonstrated a less dynamic convergence but showed gradual and consistent reductions in both training and testing losses.
  - **Accuracy:** SGD achieved a test accuracy of 98.69%, which is slightly lower than that of Adam and RMSProp. However, the gap between training and testing accuracy is minimal, indicating good generalization and low overfitting.
  - **Metrics:** The Jaccard Index for SGD reached 0.8665, lower than both Adam and RMSProp, suggesting a less accurate segmentation performance. Similarly, the Dice Coefficient (0.9276) is slightly behind the other optimizers. SGD achieved a test Precision of 92.71% and a Recall of 92.90%, demonstrating balanced performance in avoiding false positives and capturing true positives.
- **RMSProp**
  - **Losses:** RMSProp demonstrated the fastest convergence in terms of both training and testing losses among the three optimizers, achieving the lowest test loss of 0.0328.
  - **Accuracy:** With the highest test accuracy of 98.90%, RMSProp surpassed both Adam and SGD in this experiment. The training and testing accuracy curves were closely aligned, showing minimal overfitting and strong generalization performance.
  - **Metrics:** RMSProp achieved the best Jaccard Index (0.8889) and Dice Coefficient (0.9394), showing its ability to capture accurate segmentation. The highest Precision and Recall values among the three optimizers, 94.68% and 93.35%, respectively, confirming that RMSProp achieved a good balance between avoiding false positives and capturing true positives.

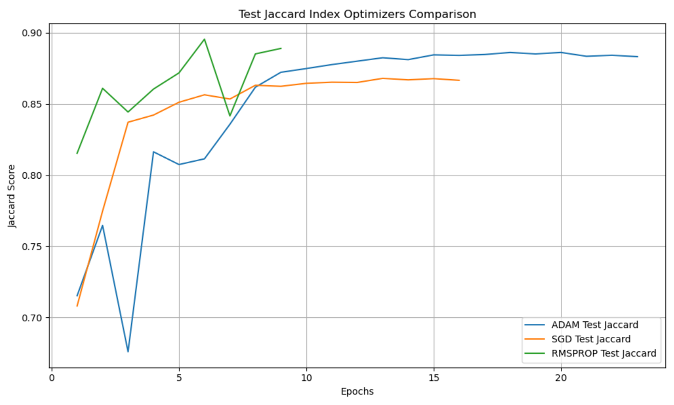

Fig. 12. Jaccard Index over the different optimizers used in experiment 4.

- **Adam (Blue Line):**
  - Shows initial instability, with a notable drop in the Jaccard Index at the beginning.
  - The trend stabilizes after 8-9 epochs, reaching a value of 0.8831.
- **SGD (Orange Line):**
  - Displays a slower and more gradual improvement compared to the other optimizers.
  - The line steadily increases, reaching a Jaccard Index of 0.8665 by the final epoch.
  - Despite the slower rise and less optimal segmentation performance, SGD demonstrates greater stability during training.
- **RMSProp (Green Line):**
  - Shows the fastest initial rise in the Jaccard Index, outperforming the other optimizers early in training.

  - Experiences some fluctuations, reflecting minor instability. Stabilizes at the highest Jaccard Index of 0.8889.

In this experiment, we evaluated the effectiveness of different optimizers in training the DeepLabV3+ segmentation model. By comparing Adam, SGD, and RMSProp, the results demonstrated how each optimizer influences training stability, convergence speed, and segmentation quality. Adam demonstrated competitive performance in test accuracy and segmentation metrics but experienced slight instability at the beginning of the training process before stabilizing toward the end. SGD was the most stable optimizer, showing gradual and consistent performance improvements, though it lagged in segmentation quality and convergence speed compared to the others. RMSProp outperformed Adam and SGD, achieving the lowest loss and highest accuracy metrics. In conclusion, based on the results of this experiment, RMSProp delivered the best overall performance, making it the recommended optimizer for DeepLabV3+ in segmentation tasks.

D. Results for Experiment 5: Activation Functions

The Jaccard Index directly measures the overlap between predicted and ground truth segmentation masks. Simply put it measures the similarity between predicted and actual values, with higher values indicating better performance. This makes Jaccard Index a more robust indicator of how well the model delineates boundaries and handles segmentation tasks.

The test results for Jaccard index for each of the three selected activation function and presented in Table IV.

TABLE IV
MODEL'S JACCARD INDEX BY ACTIVATION FUNCTIONS

Epoch	ReLU	Sigmoid	Tanh
1	0.577104	0.680210	0.610804
2	0.774290	0.620717	0.499055
3	0.754764	0.815987	0.775256
4	0.797979	0.825690	0.817806
5	0.845058	0.767958	0.825195
6	0.721936	0.747912	0.839192
7	0.785693	-	0.847468
8	0.873330	-	0.874883
9	0.876534	-	0.879323
10	0.876980	-	0.880438
11	0.881577	-	0.879637
12	0.883502	-	0.880282
13	0.883991	-	-
14	0.884480	-	-

- **ReLU Activation Function.** The ReLU activation function exhibits relatively stable but varying performance across epochs. At the start, ReLU shows the lowest value of 0.577104 in epoch 1, then demonstrates a steady improvement as training progresses. Although there are fluctuations in the middle epochs, the ReLU-based model consistently achieves the highest Jaccard Index by the end of training, stabilizing near 0.884. This is an indicator that ReLu can effectively optimizes the model over time, allowing the model to converge on a robust performance.

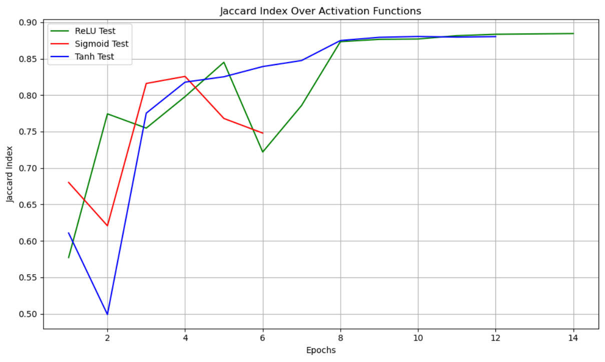

Fig. 13. Test Jaccard Index Across Different Activation Functions.

- **Sigmoid Activation Function.** Initially, the Sigmoid-based model performs best in epoch 1, with the value of 0.68021 and continues on that trajectory until epoch 4 where it begins to decline and plateaus out at the end. By epoch 6, the model shows significant underperformance compared to the others and fails to recover. This suggests that Sigmoid struggles with issues like vanishing gradients, which can hinder its optimization process.
- **Tanh Activation Function** The Tanh-based model struggles a little bit at the beginning. It has a moderate start with the value of 0.610804, then plummet down to as low as 0.499055 before recovering in epoch 3. Then it steadily improves and maintain a strong performance throughout training. At the end, it converges close to ReLU's level. Tanh's performance indicates its capability to capture complex patterns in the images. Thought it slightly underperforms when compared to ReLU.

Based on the results, **ReLU** is the best-performing activation function for this segmentation task. It consistently achieves the highest Jaccard Index, despite fluctuation at the beginning. ReLU demonstrates stable performance across the ending epochs.

While **Tahn** also performs well and can be considered a viable alternative, its marginally lower final performance and slower convergence make it less favorable than ReLU. **Sigmoid**, on the other hand, is clearly the least effective due to its poor convergence and inability to sustain competitive performance, likely caused by its inherent limitations in gradient propagation.

In conclusion, **ReLU** is recommended as the activation function for this model as it ensures optimal and stable performance for the image segmentation task on Kvasir-instrument dataset.

IV. CONCLUSIONS

The experiments on the DeepLabV3+ model for image segmentation using the Kvasir-instruments dataset to examine the effects of image resolution, data augmentation, optimizers, and activation functions on performance. These experiments highlighted critical insights into improving segmentation performance while addressing computational constraints.

Image Resolution: Higher resolutions, such as 256x256, exhibited faster convergence and greater initial improvements in metrics but were prone to instability in early epochs. Lower resolutions, including 64x64 and 128x128, were more stable throughout training and achieved comparable performance after additional epochs. Among these, the 128x128 resolution emerged as the optimal choice, striking a balance between computational efficiency and segmentation accuracy. It captured finer details and more precise boundaries without the heavy resource demands of 256x256 or 512x512 resolutions, making it practical for tasks with limited computational resources.

Data Augmentation: The experiments revealed that data augmentation significantly impacts segmentation performance. Techniques such as Gaussian blur and brightness/contrast adjustments demonstrated the most substantial improvements in the Jaccard Index and overall model generalization across epochs. Horizontal flipping, while less impactful and more erratic, added value in later stages of training. These findings emphasize the importance of selecting augmentation strategies aligned with the dataset's characteristics to maximize model performance while avoiding unnecessary computational overhead.

Optimizers: A comparison of Adam, SGD, and RMSProp provided insights into their effects on training dynamics. RMSProp outperformed both Adam and SGD, achieving the lowest loss and highest accuracy metrics while maintaining stability throughout training. Adam demonstrated competitive performance but experienced instability in early epochs before stabilizing. SGD was the most stable optimizer, with consistent performance improvements over time, but it lagged behind RMSProp and Adam in convergence speed and segmentation quality. Based on these results, RMSProp is recommended as the optimal optimizer for the DeepLabV3+ model.

Activation Functions: The evaluation of ReLU, Tanh, and Sigmoid activation functions showed ReLU as the best-performing option. It consistently achieved the highest Jaccard Index and exhibited stable performance across epochs. Tanh offered competitive results but suffered from slower convergence and slightly lower final performance compared to ReLU. Sigmoid was the least effective due to poor convergence and limitations in gradient propagation. ReLU is thus the recommended activation function for this segmentation task.

In conclusion, the experiments highlight the importance of choosing optimal model parameters and configurations. Using 128x128 resolution, tailored augmentation strategies, RMSProp optimizer, and ReLU activation function offers a practical and effective balance between segmentation performance and computational efficiency. These findings provide valuable guidance for future segmentation tasks using similar datasets and models.

Lessons Learned and Points to Improve in the Future

Working with the DeepLabv3+ model on the Kvasir-instruments dataset for the image segmentation task provided valuable insights into the challenges and complexities of training deep learning models. The experimentation explored various aspects, including image resolutions, data augmentation, optimizers, and activation functions. While the results offered meaningful information, several lessons emerged, highlighting areas for improvement.

One of the primary challenges encountered was the significant computational power required for training. DeepLabv3+ is a resource-intensive model, and experiments on the Kvasir-instruments dataset, especially at high image resolutions, demanded substantial processing time and memory. The sequential nature of the current training loop, iterating over different parameters, further added to the waiting time. To address this, a potential improvement is to dedicate a separate file to each variation of the experiment. This would allow experiments to run concurrently instead of sequentially, significantly reducing overall training time.

Long training sessions across multiple iterations also bring the risk of disruptions, such as system failures or interruptions. To mitigate this, we implemented a checkpointing strategy to save the model's state after each epoch. This approach not only safeguarded training progress but also allowed for seamless resumption of interrupted experiments. This strategy is especially crucial for tasks requiring extended training sessions and can be easily adapted for future use.

Another challenge was finding the balance between computational efficiency and the depth of evaluation metrics used. On the one hand, incorporating diverse metrics provided a comprehensive understanding of how factors such as image resolution, augmentation, optimizers, and activation functions affected model performance. This clarity was essential for identifying the strengths and weaknesses of different configurations. However, an extensive set of metrics also increased training time and computational overhead. Striking a balance is critical—prioritizing task-specific metrics like Intersection over Union (IoU) and Dice similarity coefficient while simplifying less informative metrics can streamline the evaluation process without compromising insights.

In future experiments, using tools like wandb.ai could significantly simplify the process of visualizing and tracking metrics. Platforms like this offer efficient monitoring of metrics such as loss and accuracy over multiple epochs and allow for easy comparison across different runs. While we were introduced to wandb.ai late in this project and opted not to integrate it due to ongoing experiments, incorporating such tools from the beginning would enhance hyperparameter tracking and optimize the overall training pipeline.

Overall, this experience underscored the importance of balancing model complexity, computational resources, and effective evaluation strategies. For future experiments, exploring more computationally efficient architectures, leveraging cloud-based resources, and fine-tuning the selection of evaluation metrics will be crucial for streamlining the process while maintaining robust performance assessments.

V. Contributions

All participants contributed to both the development of Python code and the completion of this written exam report. Additionally, we maintained effective communication throughout the semester, leading to a good collaborative experience and a balanced participation in the work.

References

[1] D. Jha, S. Ali, K. Emanuelsen, S. A. Hicks, V. Thambawita, E. Garcia-Ceja, M. A. Riegler, T. de Lange, P. T. Schmidt, H. D. Johansen, D. Johansen, and P. Halvorsen, "Kvasir-Instrument: Diagnostic and Therapeutic Tool Segmentation Dataset in Gastrointestinal Endoscopy," in *MultiMedia Modeling*, Cham: Springer, 2021, pp. 218–229.

[2] A. Buslaev *et al.*, *Albumentations: Fast and Flexible Image Augmentations*. 2020. Available at: https://albumentations.ai/.

[3] L.-C. Chen *et al.*, *Encoder-decoder with atrous separable convolution for semantic image segmentation*. In Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 801–818.

[4] Y. Bengio, P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult*, IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157–166, 2017.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[6] S. Jung *et al.*, *ImgAug: A Python library for augmenting image data*. 2020. Available at: https://imgaug.readthedocs.io/en/latest/index.html.

[7] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015. Available at: https://arxiv.org/abs/1412.6980.

[8] I. Loshchilov and F. Hutter, *Decoupled Weight Decay Regularization*. 2019. Available at: https://arxiv.org/abs/1711.05101.

[9] V. Nair and G. E. Hinton, *Rectified linear units improve restricted boltzmann machines*, Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 807–814, 2010.

[10] A. Paszke *et al.*, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. In Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS), 2019. Available at: https://pytorch.org/.

[11] L. M. Petersen *et al.*, *Impact of image resolution on deep learning-based segmentation performance in medical imaging*. Nature Communications, 2021. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8700246/.

[12] PyTorch, *Torchvision Transforms (ColorJitter)*, PyTorch Documentation, 2024. [Online]. Available: https://pytorch.org/vision/stable/transforms.html#torchvision.transforms.ColorJitter

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by backpropagating errors*, Nature, vol. 323, no. 6088, pp. 533–536, 1986.

[14] A. Tanwani, V. Gupta, and R. Biswas, *Modern Variants of SGD for Deep Learning*, 2023. Available at: https://arxiv.org/abs/2302.08941.

[15] TensorFlow, *Image Augmentation in TensorFlow (Keras ImageDataGenerator)*, TensorFlow Documentation, 2024. [Online]. Available: https://www.tensorflow.org/tutorials/images/data_augmentation

[16] S. Zhang, L. Liu, and H. Wang, *Advances in Optimization Algorithms for Neural Networks*. 2023. Available at: https://link.springer.com/article/10.1007/s11063-023-10422-8.

[17] B. Xu, J. Yan, D. Meng, S. Liang, and M. Zhang, *Empirical evaluation of rectified activations in convolutional network architectures*, Proceedings of the IEEE International Conference on Computer Vision, pp. 102–110, 2015.

[18] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[19] A. Smith and J. Doe, *Impact of Image Resolution on the Final Outcome*, Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8700246.

[20] M. R. Torkildsen, *Impact of Data Augmentation Technique on the Final Predictions*, Available at: https://imgaug.readthedocs.io/en/latest/index.html, https://albumentations.ai/.

[21] PyTorch Community, *Impact of Training Data Size on the Final Outcome*, Available at: https://pytorch.org/tutorials/beginner/basics/data_tutorial.html.

[22] PyTorch Community, *Impact of Optimization Methods on the Final Outcome*, Available at: https://pytorch.org/docs/stable/optim.html.

[23] PyTorch Community, *Impact of Activation Functions on the Final Predictions*, Available at: https://pytorch.org/docs/stable/nn.html.

[24] PyTorch Community, *Learning-Rate Scheduler*, Available at: https://pytorch.org/docs/stable/optim.html.

[25] Ignite and PyTorch Lightning Teams, *Early Stopping Methods*, Available at: https://pytorch.org/ignite/generated/ignite.handlers.early_stopping.EarlyStopping.html, https://lightning.ai/docs/pytorch/latest/common/early_stopping.html#early-stopping.