Deep Learning for Cell Segmentation

Comparison of U-Net and W-Net on Microscopy Images

Project Work

Authors: Josefine Høgsted Voglhofer (s231255)
Micki Karnaiya Harning (s234866)
Poul Guo Skov (s224193)

Course: 02466 - Deep Learning for Image Analysis
Date: March, 2025

Supervisors:
Alisa Pavel
Manja G. Grønbjerg

1 Abstract

2 Preface

Contents

3 Introduction

3.1 Background and Motivation

Microscopy imaging provides a cost-effective alternative to more complex lab-based methods for analyzing cell behavior, particularly in cancer research and immunotherapy studies. Quantitative analysis of cellular dynamics, such as population growth rates, morphological changes, and spatial distribution patterns, depends on accurate delineation of every cell in each frame. Manual segmentation remains labor-intensive, subjective, and prone to inter-observer variability, creating a bottleneck for high-throughput analysis.

The development of automated segmentation methods for DU145 prostate cancer cells represents an important step toward understanding cancer cell behavior in controlled experimental conditions. DU145 cells serve as a widely-used model system for studying prostate cancer biology and therapeutic responses[1]. Accurate segmentation of these cells in microscopy images is essential for downstream analyses including cell counting, growth rate measurements, and morphological characterization. Traditional image processing methods often struggle with variable cell morphologies, uneven illumination, and complex cellular structures typical in cancer cell cultures[2].

3.2 Literature Review

Deep learning approaches have revolutionized biomedical image segmentation, with U-Net architectures establishing themselves as the gold standard for medical imaging tasks[3]. The original U-Net architecture, introduced by Ronneberger et al., demonstrated superior performance on cell segmentation challenges through its encoder-decoder design with skip connections[4]. This architecture effectively captures both local details and global context, making it particularly suitable for cellular boundary detection in microscopy images.

Recent advances in U-Net variants have addressed specific challenges in cell segmentation. Attention mechanisms have been integrated to focus on relevant cellular features while suppressing background noise[5]. ResNet backbones have been adopted to leverage pre-trained features from large-scale datasets like ImageNet, improving performance on limited biomedical datasets[6]. Loss function innovations, including Dice loss, focal loss, and boundary loss, have been developed to address class imbalance issues common in segmentation tasks where cell boundaries constitute a small fraction of total pixels[7, 8].

Data augmentation strategies have proven essential for training robust models on limited microscopy datasets[9]. Geometric transformations, elastic deformations, and photometric adjustments help models generalize across varying imaging conditions. Cross-validation methodologies require careful consideration in biomedical applications to prevent data leakage and ensure reliable performance estimates[10].

3.3 Research Gap and Objectives

Despite these advances, several challenges remain in automated cancer cell segmentation. Class imbalance along thin cell boundaries continues to pose difficulties for accurate boundary detection[7]. The optimal combination of backbone architectures, attention mechanisms, and data augmentation strategies for specific cancer cell lines has not been thoroughly investigated[11]. Additionally, the trade-offs between model complexity,

computational efficiency, and segmentation accuracy require systematic evaluation for practical implementation.

The goal of this project is to build and evaluate a U-Net supervised deep-learning method for segmenting broadband microscopy images of DU145 prostate cancer cells. The resulting model should maintain strong generalization across unseen images while achieving accurate cell boundary delineation for reliable quantitative analysis.

To achieve this goal, we examine:

- The trade-offs between computational constraints (input image resolution, memory requirements) and performance?
- Which preprocessing techniques (data augmentation strategies) and hyperparameters (backbone, loss function, attention mechanisms and etc.) yield the best segmentation performance.
- How does our proposed U-Net architecture compare against the traditional OpenCV-based baseline segmentation pipeline[12] in terms of accuracy, robustness, and generalization capability?

The remaining part of the report consists of seven main sections. Section 4 provides an overview of the microscopy image dataset. Section 5 describes our methodological approach, covering the U-Net architecture, training setup, hyperparameter optimization strategy, data augmentation techniques, and loss function evaluation. Section 6 presents our experimental results, including forward selection outcomes, loss function comparisons, and final model performance. Section 8 discusses the implications of our findings and limitations of the approach. Section 9 summarizes our conclusions and key contributions. Finally, Section 10 outlines future work directions, including potential applications of unsupervised learning methods and active learning strategies for dataset expansion.

4 Data

The microscopy images we use in this project were collected using an xCELLigence RTCA microscope setup. Every hour, two types of images were taken of each well in a 96-well plate:

- **Broadband (W)**: Regular white-light images showing clearly all cells and structures. But only the DU145 cancer cells.
- **Fluorescence (B)**: Images showing NK cells, which are stained with a fluorescent dye (CellTrace Violet). These images only highlight NK cells, but the brightness of the NK cells gradually fades over time due to the dye spreading out as cells divide and being bleached by the microscope's light.

Initially (0 h to 28 h), the wells only contain cancer cells, so there are no NK cells visible in the fluorescence images. At 28 hours, NK cells and antibodies are added to the wells, so after this point, the images contain both DU145 cancer cells and NK cells.

We chose to focus solely on the broadband (W) channel. The fluorescent images lose signal over time because the dye both bleaches and gets split between daughter cells when NK cells divide, so intensity drifts from frame to frame. That makes the data noisier and harder to normalize, and we have far fewer reliable labels in that channel. Broadband frames, in contrast, keep a stable grey-scale range and clearly show the DU145 cell clusters we want to segment, so the model can learn consistent features without fighting time-dependent artifacts. If we later need information that only the fluorescent channel contains, we can treat it separately, but for training and validation the broadband images give us the cleanest, most uniform dataset.

While this approach affords our model the most favorable starting conditions, it necessarily excludes training on data that accurately reproduces the complexity of real-world microscopy images. We accept this compromise in order to deliver a sufficiently robust baseline model that can serve as a foundation for subsequent work. Given the constraints of our project timeline and resources, we lack the opportunity to fine-tune the network to the level of reliability required to distinguish DU145 cancer cells from NK cells in a mixed-cell context. As a result, the present model should be regarded as a proof-of-concept rather than a final, deployable segmentation solution.

Our dataset consists of a small subset of manually labeled images (41 broadband images labeled with cancer cell masks and 41 fluorescent images labeled with NK cell masks), and a much larger set of unlabeled images (approximately 8000 broadband and 8000 fluorescent images) meant primarily for future experiments.

4.1 Data

4.2 Data Collection and Image Types

The microscopy images we use in this project were collected using an xCELLigence RTCA microscope setup. Every hour, two types of images were taken of each well in a 96-well plate:

- **Broadband (W)**: Regular white-light images showing all visible cellular structures, primarily DU145 cancer cells with clear morphological details.

- **Fluorescence (B)**: Images showing NK cells stained with fluorescent dye (CellTrace Violet). These images selectively highlight NK cells, but suffer from signal degradation over time due to dye bleaching and dilution during cell division.

The experimental timeline involves two distinct phases: initially (0-28h), wells contain only cancer cells, with no NK cells present in either imaging modality. At 28 hours, NK cells and antibodies are introduced, after which both cell types coexist in the culture.

4.3 Data Selection and Implications

We chose to focus solely on the broadband (W) channel for model development. This decision has several important implications for our model and training data:

Model Scope: Our trained model will exclusively segment DU145 cancer cells. It will not be capable of identifying or segmenting NK cells, as these are not included in labeled broadband images and we do not use the corresponding NK cell masks for training.

Training Masks: Although our dataset includes 41 fluorescent images with NK cell masks, we utilize only the 41 broadband images with cancer cell masks for model training and validation. The NK cell masks remain unused in this study.

Rationale: Broadband images provide several advantages for cancer cell segmentation: stable grey-scale intensity ranges across time points, consistent cellular morphology representation, and freedom from time-dependent fluorescence artifacts. In contrast, fluorescence images exhibit signal drift, photobleaching effects, and reduced annotation reliability over extended imaging periods.

Limitations: This approach means our model cannot distinguish between different cell types in mixed populations, limiting its applicability to pure cancer cell cultures or early time points before NK cell introduction.

4.4 Dataset Composition

Our final dataset consists of:

- **Training data**: 41 manually labeled broadband images with corresponding cancer cell segmentation masks
- **Unused annotations**: 41 fluorescence images with NK cell masks (reserved for potential future work)
- **Unlabeled data**: Approximately 8,000 broadband and 8,000 fluorescent images for potential future expansion

This focused approach provides a clean, consistent dataset for cancer cell segmentation while acknowledging the trade-off of reduced biological complexity compared to mixed cell population analysis.

5 Theory

5.1 Deep Learning vs. Traditional Methods for Cell Segmentation

Traditional approaches for biomedical image segmentation, particularly cell segmentation in microscopy images, rely heavily on handcrafted rules and pixel-level statistics. These methods include thresholding techniques, edge detection algorithms (such as Canny edge detection), region growing methods, watershed segmentation, and clustering approaches like k-means[**traditional_segmentation_review_2020**]. While these methods are computationally efficient and interpretable, their performance deteriorates significantly when confronted with the inherent challenges of microscopy images: variable cell morphologies, uneven illumination, weak cell boundaries, background noise, and overlapping cellular structures[2].

In contrast, deep learning methods have revolutionized biomedical image segmentation by automatically learning hierarchical feature representations directly from annotated data[13]. Convolutional Neural Networks (CNNs) extract spatial features through convolutional layers, while Fully Convolutional Networks (FCNs) extend this concept to dense prediction tasks like segmentation by replacing fully connected layers with convolutional layers, enabling pixel-wise classification[14]. These architectures automatically learn hierarchical features from raw data and can capture complex contextual relationships in images through their multi-scale feature extraction capabilities[**krizhevsky_imagenet_2012**, 13].

U-Net, introduced specifically for biomedical image segmentation[4], addresses several limitations of standard CNNs and FCNs for cell segmentation tasks. Unlike conventional CNNs that lose spatial resolution through pooling operations, U-Net preserves fine-grained spatial information through skip connections that directly link encoder and decoder layers at corresponding resolutions. This architectural innovation enables U-Net to maintain precise boundary localization while capturing semantic context—a critical requirement for accurate cell boundary delineation[4]. Furthermore, U-Net's symmetric encoder-decoder design with skip connections allows it to work effectively with limited training data, a common constraint in biomedical applications where manual annotation is expensive and time-consuming[3].

The superior performance of U-Net for cell segmentation tasks stems from its ability to combine local detail preservation with global context understanding, making it particularly suitable for segmenting cellular structures with complex morphologies and varying sizes typical in cancer cell cultures[15].

5.2 U-Net Architecture

5.2.1 Encoder-Decoder Structure
5.2.2 Skip Connections
5.2.3 Bottleneck Representation

5.3 Backbone Networks

5.3.1 ResNet Architecture
5.3.2 Transfer Learning Principles

5.4 Attention Mechanisms

5.4.1 Spatial Attention via Attention Gates
5.4.2 Channel Attention via Squeeze-and-Excitation

5.5 Evaluation Metrics

5.5.1 Intersection over Union (IoU)
5.5.2 Additional Segmentation Metrics

5.6 Loss Functions for Segmentation

5.6.1 Distribution-based Losses
5.6.2 Region-based Losses
5.6.3 Compound Losses

6 Methodology

6.1 Experimental Design Overview

6.2 Model Architecture Implementation

6.2.1 U-Net Configuration
6.2.2 Backbone Selection and Integration
6.2.3 Attention Module Integration

6.3 Training Protocol

6.3.1 Cross-Validation Strategy
6.3.2 Training Hyperparameters
6.3.3 Early Stopping and Learning Rate Scheduling

6.4 Hyperparameter Optimization

6.4.1 Grid Search Strategy
6.4.2 Parameter Selection Rationale
6.4.3 Computational Constraints

6.5 Data Augmentation Pipeline

6.5.1 Forward Selection Algorithm
6.5.2 Augmentation Techniques Evaluation
6.5.3 Implementation Details

6.6 Loss Function Selection

6.6.1 Evaluation Protocol
6.6.2 Cross-Validation for Loss Comparison

6.7 Baseline Comparison

6.7.1 Traditional OpenCV Pipeline
6.7.2 Evaluation Framework

7 Method

7.1 Deep Learning Method vs. Traditional Method

Traditional approaches, such as thresholding, edge detection, region growing, clustering, and graph-based techniques, rely heavily on handcrafted rules and pixel-level statistics. These methods are often computationally efficient and easy to interpret. However, their performance deteriorates significantly when faced with images featuring variable intensity, noise, weak boundaries, or high complexity, particularly in biomedical images.[16]

In contrast, deep learning methods, especially convolutional neural networks (CNNs), fully convolutional networks (FCNs), and U-Net variants, has dramatically boosted segmentaion accuracy. These architectures automatically learn hierarchical features from raw data and can capture complex contextual relationships in images. This often improves robustness to noise, variable illumination and complex shapes. The trade-off is a higher demand for annotated examples, greater computational cost and models that are harder to interpret or debug.[16]

We adopt U-Net because it combines pixel-level precision with global context, works reliably on the small biomedical datasets typical in microscopy, and is already the go-to architecture in medical imaging [15]. A more detailed explanation of the theory behind U-Net is provided below.

7.2 Workflow overview

Lav etv et training pipeline diagram

7.3 U-Net Theory

The U-Net architecture consists of an encoder, a decoder, and a bottleneck. The encoder follows a conventional convolutional neural network (CNN) structure, similar to VGG16 or ResNet. After the encoder has decomposed the input image into increasingly abstract feature representations, the decoder attempts to reconstruct an output image or segmentation mask with the same spatial dimensions as the input. This reconstruction process typically involves transposed convolutions (also known as upsampling) or dedicated upsampling layers to increase spatial resolution.

The bottleneck acts as a latent space representation, where the image is compressed into a lower-dimensional format while preserving semantically rich and informative features.

As seen in figure 4.1, the encoder and decoder are symmetrical, and each corresponding pair of encoder and decoder layers is connected via skip connections—U-Net's distinguishing feature. Specifically, before each pooling operation in the encoder, the feature maps are forwarded and concatenated with the output of the corresponding decoder layer after the up-convolution. These skip connections help the network retain both high-level semantic information and low-level spatial details such as edges, textures, and precise object locations.

In our implementation, the U-Net is initialized with a pre-trained ResNet backbone, trained on the ImageNet dataset. Given the class imbalance present in our dataset, we will experiment with various loss functions and hyperparameters during fine-tuning to determine the best configuration.

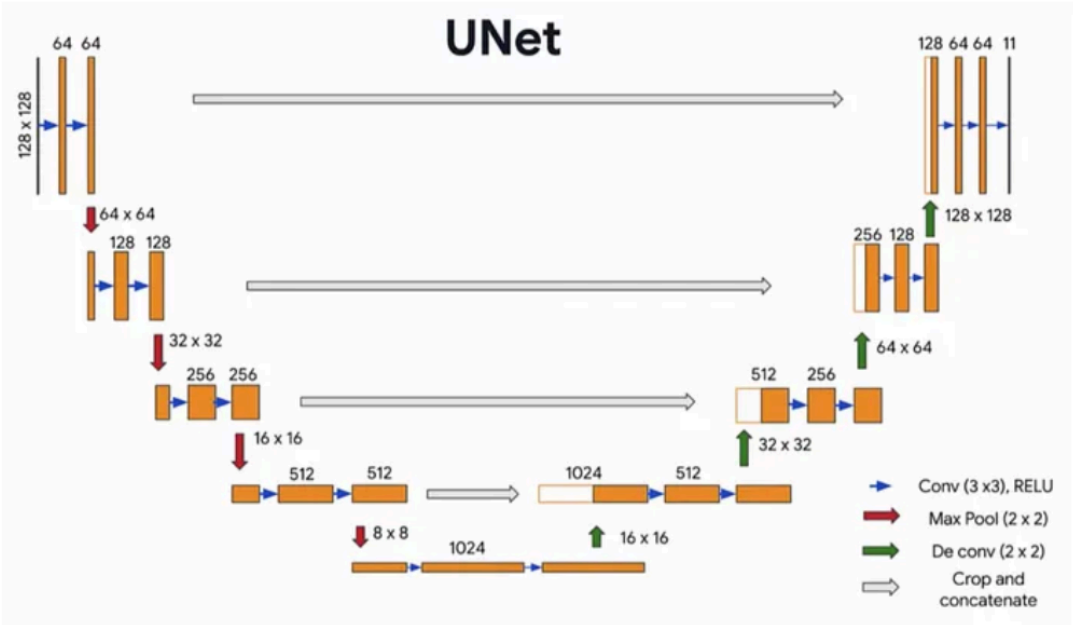

Figure 7.1: U-Net structure [17]

Note that the results shown in the figure are based on a plain U-Net architecture without a backbone. As our model includes a backbone, the reported metrics may differ accordingly.

7.4 Training Setup

We train the models using a $?$ fold GroupKFold cross validation to ensure fair evaluation. We did not choose the number of folds at random, we rather did cross validation over a number of different folds $2, 3, 4, 5$, to see which number of folds would yield the best results. Normally one would use a standard, tried and true number of folds, like 5 or 10. But because of our limited number of labeled data, each test fold would be very thin, making the results from each fold very dependent on which data point ended up there.

We also use early stopping based on validation IoU to prevent overfitting. The models are optimized using the Adam optimizer. For dynamic adjustment of the learning rate, we employ the `ReduceLROnPlateau` function.

`ReduceLROnPlateau`: This function monitors a chosen metric (e.g., validation IoU) and reduces the learning rate when the metric shows no significant improvement after a set number of epochs (the `patience` parameter). Mathematically, if the current learning rate is ℓ, then upon stagnation, it is updated as:

$$\ell_{\text{new}} = \ell \times \texttt{factor}$$

Typically, `factor` is set to 0.1, meaning the learning rate is reduced to 10% of its value. We also use early stopping based on validation IoU to further prevent overfitting.

For evaluation, we will compare models using Intersection over Union (IoU), F1-score, precision, recall, and accuracy for supervised and semi supervised models, while the unsupervised model will primarily be evaluated qualitatively and with clustering based metrics such as soft and cut loss.

7.5 IoU

The Intersection over Union also known as the Jaccard index, is one of the most popular evaluation metric for image segmentation. IoU is mathematically defined as:

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} = \frac{|\text{Intersection}|}{|\text{Union}|} \tag{7.1}$$

where A and B represent the predicted and ground truth segmentation masks, respectively. In our implementation we flatten both predicted and ground truth masks and classify each pixel into four categories:

- **True Positive (TP)**: Correctly predicted foreground pixels
- **False Positive (FP)**: Incorrectly predicted as foreground
- **False Negative (FN)**: Missed foreground pixels

Our IoU formula becomes:

$$\text{IoU} = \frac{TP}{TP + FP + FN + \epsilon} \tag{7.2}$$

where $\epsilon = 10^{-7}$ is added to prevent division by zero.

This formulation is equivalent to the intersection-over-union definition because:

- **Intersection** = True Positives (pixels correctly identified as foreground)
- **Union** = True Positives + False Positives + False Negatives (all pixels that should be or are predicted as foreground)

IoU is a good measure for image segmentation as it measures proportion of the overlap between the predicted mask and the ground truth, making it scale invariant. The score ranges from 0 to 1 where 0 means there is no overlap and 1 signifies a perfect overlap. [18]

7.6 Hyperparameters

When selecting hyperparameters for our model, we must balance thorough exploration against our limited computational budget. In principle, we could tune eight different parameters—backbone architecture, image resolution, loss function, batch size, attention mechanism, weight decay, optimizer choice, and learning rate—but we instead focus our search on the three most impactful settings: the backbone model, batch size, and whether to include an attention module. All other parameters are held constant at values known to work well for U-Net variants. First, we fix the input resolution at 128×128. Although higher resolutions such as 256×256 or 512×512 often improve segmentation accuracy, they incur prohibitive GPU memory and runtime costs in our environment. Second, we adopt the Adam optimizer — shown to perform robustly on U-Net–style architectures [19] — with an initial learning rate of 1×10^{-3} and a weight decay of 1×10^{-8}. Because Adam automatically adjusts its effective step size during training, we do not compare alternative optimizers or decay schedules.

Secondly, hyperparameter optimization involves two intertwined components: the search strategy and the evaluation protocol. Common search strategies include grid search, random search, and Bayesian optimization. Grid search systematically evaluates every combination in a predefined grid, but its cost grows exponentially with the number of parameters and candidate values. Random search, by contrast, samples configurations uniformly

and often locates high-performing settings more efficiently when only a few hyperparameters truly matter [20]. Bayesian optimization builds a probabilistic surrogate model of the objective function and sequentially proposes promising configurations, achieving greater sample efficiency at the expense of additional algorithmic overhead [21]. In our project, however, we restrict ourselves to tuning only three hyperparameters (backbone, batch size, and attention mechanism), each with a small set of candidate values.

This makes a full grid search feasible and ensures that no configuration is overlooked. However, grid search also has drawbacks. It can lead us to focus too much on small differences that do not generalize beyond our cross validation splits. It can create a false sense of certainty by treating every combination as equally plausible. In addition, grid search does not make use of information gathered during the search process as adaptive methods such as Bayesian optimization do. For these reasons we adopt grid search for its simplicity and reproducibility under our resource constraints, while remaining cautious about overemphasizing marginal performance gains and recognizing that a more efficient search strategy would be preferable in larger or more complex tuning scenarios.

Regardless of the chosen search strategy, k-fold cross-validation is used to estimate each candidate's out-of-sample performance. Cross-validation partitions the data into k folds, trains on $k-1$ folds, and evaluates on the held-out fold, repeating this process k times to yield an average score [22]. Although cross-validation itself is an evaluation method rather than an optimization algorithm, it provides a fair comparison between hyperparameter configurations when embedded within a search loop.

Nested cross-validation—where an inner loop selects hyperparameters via cross-validation and an outer loop estimates the generalization error—offers a less biased performance estimate, especially on small datasets. However, its computational cost multiplies the number of folds by the number of parameter combinations, making it infeasible under our resource constraints. Therefore, we employ standard (non-nested) 5-fold cross-validation within a grid search over our three primary hyperparameters (backbone architecture, batch size, and attention mechanism). This approach balances robustness of evaluation with the practical need to limit compute time. Provided below, is theory for the three hyperparameters we chose to explore.

7.6.1 Backbone

When developing a machine learning model under tight time constraints, leveraging a pre-trained backbone can be an efficient and effective solution. Rather than training a network from scratch, we borrow a backbone that has already been trained on a large, general-purpose dataset such as ImageNet. To adapt this backbone for a specific segmentation or classification task, we remove its final fully-connected layer responsible for producing ImageNet logits.

For segmentation, we append a decoder, like the U-Net decoder, that upsamples and refines the backbone's multi-scale feature maps via skip-connections. At this point, we can either fine-tune or freeze the backbone. Fine-tuning allows the pre-trained weights to adjust to our dataset, specializing the filters to our domain. Freezing keeps the backbone weights fixed and trains only the newly added layers, which can be advantageous when the labeled dataset is small.

In our project, we examined two variations of the same backbone architecture: ResNet-34 and ResNet-50. We chose ResNet because of our prior positive experience and its demonstrated performance in the literature [17]. ResNet introduces residual (skip) connections that enable stable training of very deep convolutional networks without suffering from vanishing gradients or degradation. ResNet34 comprises roughly 21 million parameters organized into 34 layers of "basic" residual blocks, each block containing two 3×3 convolutions. ResNet50 increases depth to 50 layers and uses "bottleneck" blocks—each

block consisting of a 1×1 convolution, a 3×3 convolution, and another 1×1 convolution—totaling approximately 25 million parameters.

7.6.2 Attention Mechanisms

In cell segmentation, some areas contain a lot of useful information (like clear cell boundaries), while others might be less important. We therefore chose to include attention mechanisms to allows the network to weigh these regions differently, which can lead to more accurate and efficient segmentation. It helps the model not just look at everything equally, but instead prioritize the most meaningful features. We employ two types of attention: spatial attention (via `AttentionGate`) and channel attention (via a Squeeze-and-Excitation mechanism).

`Spatial Attention via Attention Gate:` The `AttentionGate` module refines skip connection features by emphasizing the most relevant spatial regions. It takes two inputs: a gating signal g from the decoder and a feature map x from the encoder. Both inputs are first transformed via 1×1 convolutions and batch normalization to an intermediate feature space, then added element-wise and passed through a ReLU activation. A subsequent 1×1 convolution, batch normalization, and Sigmoid function produce an attention map ψ, which is used to scale the encoder features:

$$\text{Output} = x \times \psi\big(\text{ReLU}(W_g g + W_x x)\big)$$

`Channel Attention via Squeeze-and-Excitation:` The channel attention module recalibrates feature maps by emphasizing informative channels. It uses both global average pooling and global max pooling to extract channel descriptors, which are then passed through a small convolutional network with a ReLU activation and finally a Sigmoid activation to produce channel-wise weights. These weights are multiplied with the original features to boost important channels.

Both attention modules are optionally integrated into the U-Net to perform ablation studies and assess their impact on segmentation results.

7.6.3 Batch Size

The batch size is the number of images from the training dataset, the network processes and uses to update the weights all at once. When running a forward and backwards pass over a batch, we calculate the gradient as the mean value from the loss over all images in the batch.

When using a small batch size, we get the following consequences; A more noisy gradient, as each update is only based on a few images, so the gradient estimates are more uncertain. We get to adjust the weights more frequently, which can make the model converge quicker. But this also means we do a lot more calls to the kernel, which can be computational inefficient.

When using a large batch size, we get the following consequences; A more stable gradient, as we use many samples per update, making the gradient approach the 'true' gradient, over the full dataset. Doing less updates, which can delay the fine-tuning process, but is also more GPU efficient.

7.7 Data augmentation strategy

Because our labeled data is so limited, doing augmentation to our images, and creating new labeled data, can be a great strategy to give our model better conditions. But there are many different augmentation strategies, and applying all at once, can at times do more harm than good. So to systematically identify the most effective data augmentation techniques for our cell segmentation task, we implemented a forward selection algorithm that

evaluates augmentation strategies based on their contribution to IoU performance. This approach ensures that our final augmentation pipeline contains only transformations that provide measurable improvements to model generalization. We began by identifying a set of candidate data augmentation techniques, that we selected based on previous work [23]. We selected only those augmentations that demonstrated at least a **0.5 percentage point** improvement in segmentation performance. From this pool, further refinement was performed to eliminate redundant or overlapping transformations:

- **Removed** `Flip`: Its functionality is fully covered by separate `HorizontalFlip` and `VerticalFlip`.
- **Removed** `Rotate`: Since `Affine` already includes rotation, translation, and scaling.
- **Dropped** `RandomBrightness`: We rely on `ColorJitter`, which encompasses brightness adjustments.
- **Omitted** `HueSaturationValue`: This is irrelevant for grayscale images, and its effect is addressed by `RandomGamma`.
- **Excluded** `GaussianBlur`: Its effect is subsumed by the more general `AdvancedBlur`.

This filtering process left us with 18 distinct augmentation methods, forming our initial pool.

7.7.1 Forward Selection Algorithm

Our selection process employed a forward selection strategy. Forward selection is a stepwise optimization strategy where the algorithm iteratively adds the single augmentation from the remaining candidate pool that yields the highest improvement in model performance (measured by IoU in our case). At each step, the best-performing augmentation is added to the current set, and this process repeats until no further augmentation leads to a significant improvement beyond a predefined threshold. We implemented this with the following characteristics:

- **Candidate Pool**: 18 augmentation techniques spanning geometric transformations, intensity modifications, blur effects, and dropout methods
- **Selection Criterion**: Minimum IoU improvement of 0.0005 (0.05%) to continue selection
- **Validation**: 3-fold cross-validation with group-based splitting to prevent data leakage between image variants
- **Evaluation**: Each candidate was tested by adding it to the current best set and measuring the resulting performance

The algorithm iteratively selects the augmentation that provides the largest improvement to cross-validated IoU score, continuing until no candidate meets the minimum improvement threshold.

7.8 Loss Function Evaluation

The choice of loss function significantly impacts segmentation model performance, particularly for tasks with class imbalance like cell segmentation where background pixels vastly outnumber cell pixels. Following the categorization by Jadon [24], we evaluated 11 different loss functions from three main categories:

- **Distribution-based losses**: Binary Cross-Entropy (BCE), Weighted BCE, Balanced BCE, Focal Loss, and Distance Map BCE. These losses are derived from probability distributions and handle pixel-level classification.
- **Region-based losses**: Dice, Tversky, Sensitivity-Specificity, and Log-Cosh Dice. These losses optimize overlap-based metrics and handle class imbalance better than distribution-based losses.
- **Compound losses**: Combo loss, which combine multiple loss functions into one. Combo loss is a mix of BCE and Dice.

8 Results

This section presents the findings from our supervised UNet experiments. We first establish the experimental setup, determine the best augmentation methods and number of augmentations, the optimal number of cross-validation folds. Then we evaluate different loss functions, and analyze how well our models generalize to unseen data.

8.1 Experimental Setup

All experiments follow a consistent protocol to ensure fair comparison across different configurations. The shared experimental parameters are summarized in Table 6.1.

Table 8.1: Fixed training configuration used throughout.

Parameter	Value
Backbone	ResNet-34
Image size	128×128 pixels
Batch size	4
Optimizer	Adam ($\eta = 10^{-3}$, weight decay 10^{-5})
Epochs	25 with early stopping on val IoU

We employ GroupKFold cross-validation to ensure that all augmented variants of a given base image remain within the same split, preventing information leakage between training and validation sets. This approach turned out to be critical given our data augmentation strategy, as it ensures the model is never tested on transformed versions of images it has seen during training.

8.2 Forward Selection Results

Table 8.2: Progressive IoU improvements during forward selection

Iteration	Added Augmentation	IoU Score	Improvement
Baseline	None	0.8081	-
1	RandomRotate90	0.8400	+0.0319
2	Affine	0.8450	+0.0050
3	VerticalFlip	0.8476	+0.0027
4	AdvancedBlur	0.8482	+0.0006

The selection terminated after 4 iterations when no remaining candidates achieved the 0.0005 improvement threshold, suggesting an optimal balance between augmentation diversity and diminishing returns.

The forward selection algorithm identified four augmentations that collectively achieved a 4.97% improvement in IoU performance:

RandomRotate90 provided the largest gain (+0.0319 IoU), indicating strong directional bias in our dataset where rotational invariance significantly enhances generalization. **Affine** transformation (+0.0050 IoU) combines scale, translation, rotation, and shear for comprehensive geometric augmentation. **AdvancedBlur** (+0.0006 IoU) simulates optical effects

and focus variations. The dominance of geometric transformations (three of four selected) indicates that spatial invariance is more critical than intensity variations for our images.

Since our dataset is relatively small we apply data augmentation techniques such as horizontal flipping, brightness and contrast adjustments, and coarse dropout to increase the variability in the training data. These transformations help the model become more robust to changes in lighting, orientation, and partial occlusions, which improves its ability to generalize to new data.

To ensure a fair evaluation during cross-validation, we group all augmented samples based on their original image. This means that all versions of a specific image are placed in the same fold, preventing data leakage between training and validation sets. In this way, the model is never tested on augmented versions of images it has already seen during training.

8.3 Loss Function Evaluation

We evaluated 11 different loss functions using 5-fold cross-validation to identify the most suitable objective for our cell segmentation task. The complete results are shown in Table 6.3.

Table 8.3: 5-fold cross-validation performance of 11 loss functions (mean ± SD).

Loss function	**CV IoU**
BCE Loss	0.8250 ± 0.0419
Combo Loss	0.8243 ± 0.0393
Distance Map BCE	0.8195 ± 0.0356
Focal Loss	0.8193 ± 0.0396
Dice Loss	0.8164 ± 0.0376
Focal Tversky	0.8145 ± 0.0520
Weighted BCE	0.8003 ± 0.0474
Tversky Loss	0.7980 ± 0.0329
Log-Cosh Dice	0.7335 ± 0.1821
Sensitivity–Specificity	0.7300 ± 0.0718
Balanced BCE	0.7161 ± 0.0704

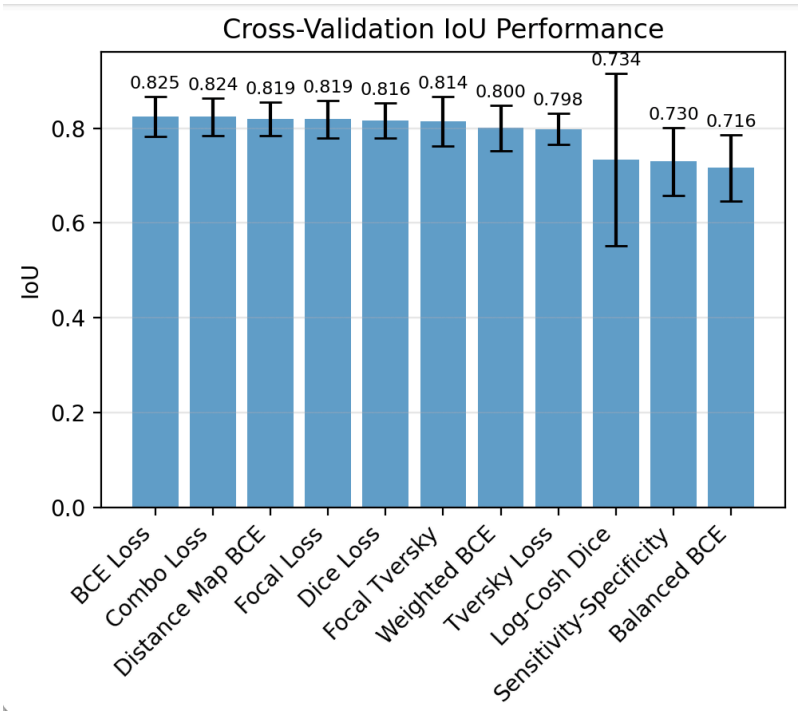

Figure 8.1: Cross-validation performance summary showing IoU and F1 scores for all 11 loss functions, with detailed rankings and statistics.

The top-performing loss functions in cross-validation were BCE Loss (0.8250), Combo Loss (0.824) and Distance Map BCE (0.819). However, we suspected that cross-validation performance alone might not tell the whole story about how well these models would perform on completely new data.

8.4 Final Model Selection

Based on our evaluation results, we selected **BCE Loss** as our final objective function. This decision was based on its superior test performance (highest IoU of 0.7579), minimal overfitting (gap of 0.0033), and best performance during the cross validation evaluation. BCE Loss also offers computational advantages due to its simplicity compared to complex compound loss functions.

8.5 Hyperparameter Optimization Results

After selecting BCE Loss as our objective function, we conducted hyperparameter optimization focusing on three parameters: backbone architecture (ResNet34 vs ResNet50), attention mechanism (enabled vs disabled), and batch size (2 vs 4). Cross-validation results showed no statistically significant differences between configurations ($p > 0.05$ using paired t-tests), with all combinations achieving similar IoU scores in the range of 0.75-0.76.

Given the minimal performance differences, we selected the most computationally efficient configuration: ResNet34 backbone (fewer parameters), disabled attention mechanism (reduced computational overhead), and batch size of 2 (due to GPU memory constraints). This configuration maintains competitive performance while optimizing for computational efficiency and was necessary to avoid out-of-memory errors during training.

9 Discussion

10 Conclusion

11 Future Work

- Unsupervised learning
- Active learning

11.1 Unsupervised learning model

Given the large number of unlabeled data points, we initially intended to include an unsupervised model in our project. Based on preliminary research, we selected the W-Net architecture for this purpose. However, during implementation we encountered several issues. Initial testing resulted in a low IoU score of 0.12, and we found it challenging to define an appropriate metric for comparing the unsupervised model fairly with the supervised one. Considering our limited time and resources, we concluded that improving the performance of the unsupervised model would require more effort than we could allocate. As a result, we chose to focus our work entirely on the supervised pipeline to ensure a more complete and reliable outcome.

In appendix A.1, you will find a W-Net theory section, as we had a U-Net theory section in our method section.

11.2 Active Learning

For future work, it could be beneficial to explore the use of an active learning strategy. The current labeled dataset is quite small, which makes it difficult to train a reliable machine learning model. Since we have around 16,000 unlabeled data points, it may be worth investing time in labeling more data. However, because manual labeling is costly, the additional data should not be chosen at random. Instead, a pool-based active learning approach, such as uncertainty sampling or query-by-committee, could help us identify the most valuable images to label, in order to improve the model as efficiently as possible.

In this project, we encountered another challenge where the use of active learning could be beneficial: selecting the optimal number of augmentations per image when working with a small labeled dataset. Data augmentation is indispensable for enriching limited data, yet the ideal augmentation budget is far from obvious. On the one hand, too few transforms fail to introduce sufficient diversity, leaving the model exposed to nearly identical examples, on the other hand, beyond roughly ten variants per image, marginal returns rapidly diminish, and extreme transformations can introduce artifacts that the model overfits to, ultimately degrading its performance on unseen, real-world data.

To address this, we propose an active-learning strategy: first, train a baseline U-Net model without augmentation and use cross-validation to verify its performance; then, estimate each training image's uncertainty, for example, by computing the average per-pixel entropy under MC-dropout. Next, partition the dataset into "easy" (low uncertainty) and "hard" (high uncertainty) subsets, and assign a smaller augmentation budget (e.g., three to five variants) to the easy set and a larger budget (e.g., ten to fifteen variants) to the hard set. By concentrating our computational resources where the model struggles most, we avoid both under- and over-augmenting, resulting in a more efficient and effective use of data. Although time constraints prevented a full implementation, this approach offers a promising path toward smarter, uncertainty-aware data augmentation.

Bibliography

[1] Kenneth R Stone et al. “DU145 prostate cancer cell line: characteristics and applications in cancer research”. In: *Cancer Research* 39.11 (1978), pp. 4455–4459. URL: https://cancerres.aacrjournals.org/content/39/11/4455.

[2] Alden A Dima et al. “Comparison of segmentation algorithms for fluorescence microscopy images of cells”. In: *Cytometry Part A* 79.7 (2011), pp. 545–559. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/cyto.a.21079.

[3] Yingying Xu et al. “Advances in Medical Image Segmentation: A Comprehensive Review of Traditional, Deep Learning and Hybrid Approaches”. In: *Bioengineering* 11.10 (2024), p. 1034. URL: https://www.mdpi.com/2306-5354/11/10/1034.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*. 2015, pp. 234–241. URL: https://arxiv.org/abs/1505.04597.

[5] Ozan Oktay et al. “Attention U-Net: Learning Where to Look for the Pancreas”. In: *Medical Image Analysis* 53 (2018), pp. 197–207. URL: https://arxiv.org/abs/1804.03999.

[6] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778. URL: https://arxiv.org/abs/1512.03385.

[7] Hoel Kervadec et al. “Boundary loss for highly unbalanced segmentation”. In: *Medical Image Analysis* 67 (2021), p. 101851. URL: https://www.sciencedirect.com/science/article/abs/pii/S1361841520302152.

[8] Yuxuan Wang, Hengyong Zhang, and Ge Zhang. “Focal Loss Based Deep Convolutional Neural Network for Chest Disease Detection”. In: *IEEE Access* 8 (2020), pp. 146317–146326. URL: https://doi.org/10.1109/ACCESS.2020.3024904.

[9] Jingchao Ma et al. “Review of Image Augmentation Used in Deep Learning-Based Material Microscopic Image Segmentation”. In: *Applied Sciences* 13.11 (2023), p. 6478. URL: https://doi.org/10.3390/app13116478.

[10] Carlos A Silva et al. “A Guide to Cross-Validation for Artificial Intelligence in Medical Imaging”. In: *Radiology: Artificial Intelligence* 5.4 (2023), e230785. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC10388213/.

[11] Narinder Singh Punn and Sonali Agarwal. “Modality Specific U-Net Variants for Biomedical Image Segmentation: A Survey”. In: *Artificial Intelligence Review* 55.6 (2022), pp. 4845–4889. URL: https://link.springer.com/article/10.1007/s10462-022-10152-1.

[12] Maja Jønck Hjuler. *DigitSTEM cell image analysis*. Tech. rep. Research Assistant Report. Technical University of Denmark, Dec. 2024.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. URL: https://www.nature.com/articles/nature14539.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440. URL: https://arxiv.org/abs/1411.4038.

[15] Narinder Singh Punn and Sonali Agarwal. “Modality Specific U-Net Variants for Biomedical Image Segmentation: A Survey”. In: (2022). URL: https://link.springer.com/article/10.1007/s10462-022-10152-1.

[16] Y Xu et al. “Advances in Medical Image Segmentation: A Comprehensive Review of Traditional, Deep Learning and Hybrid Approaches”. In: (2024). URL: https://www.mdpi.com/2306-5354/11/10/1034.

[17] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: 2016. URL: https://arxiv.org/abs/1512.03385.

[18] Hamid Rezatofighi et al. “Generalized Intersection over Union”. In: (June 2019).

[19] Parth Thakur, Abhishek Roy, and Sushil Kumar Jain. “A Review of Medical Image Segmentation Models”. In: (2023). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11300773/.

[20] James Bergstra and Yoshua Bengio. “Random Search for Hyper-Parameter Optimization”. In: *Journal of Machine Learning Research* 13 (2012).

[21] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. In: *Advances in Neural Information Processing Systems*. Vol. 25. 2012.

[22] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2009. URL: https://hastie.su.domains/ElemStatLearn/.

[23] Jingchao Ma et al. “Review of Image Augmentation Used in Deep Learning-Based Material Microscopic Image Segmentation”. In: *Applied Sciences* (May 2023). DOI: 10.3390/app13116478. URL: https://doi.org/10.3390/app13116478.

[24] Shruti Jadon. “A survey of loss functions for semantic segmentation”. In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, Oct. 2020, pp. 1–7. DOI: 10.1109/cibcb48159.2020.9277638. URL: http://dx.doi.org/10.1109/CIBCB48159.2020.9277638.

A Appendix

A.1 W-Net Theory

The W-Net architecture consists of two sequential U-Net-like networks, effectively forming an encoder-decoder-encoder-decoder structure. The first U-Net acts as an encoder-decoder network that compresses and reconstructs the input image, producing an initial segmentation. The second U-Net then refines this segmentation by taking the output of the first as input, further processing and enhancing the segmentation mask.

Each of the encoder and decoder paths follows a standard convolutional neural network design, with downsampling performed via convolution and pooling operations, and upsampling performed through transposed convolutions or upsampling layers. Skip connections are used in both U-Net components to retain spatial details and facilitate gradient flow, allowing low-level features from early layers to influence the reconstruction in later stages.

The bottleneck of each U-Net serves as a compressed latent space that encodes the most informative features of the input while reducing dimensionality. By stacking two such networks, W-Net effectively performs a coarse-to-fine segmentation process, where the second network corrects and sharpens the output of the first.

husk at skrive formler og mere formel matematik for loss functions osv

Hvor kan vi gå igang:

- **log bog**
- **skriv om loss funktionerne**
- **skriv mere matematisk disposition**

spørgsmål

- Hvilken rækkefølge ville være god? Hvordan holder vi rød tråd?
- should we split method into two; method and theory
- Skal vi i vores introduktion inkludere at vi undersøgte unsupervised, men det virkede ikke, så fokus ender på supervised. Eller skal vi holde os til at skrive kun om supervised i selve rapporten, og så nævne det i future work?
- Preface til hvert afsnit
- We are a little confused about how much we should write theroy about. Earlier we have always learned we should not write about something we have learned in school earlier. If we should write about CNN's and Adam Optimizers, it will be a very long report. Were is the limit?
-

talk about why we split into multiple experiments. The best would be to add augmentation method selection into the large experiment, but it would be unfeasable. We choose a base model for this first experiment, however we should be aware that when we change the model later on, like the loss function, this could also change the optimal augmentation methods.

nævn at vi bruger albumentations library til at lave augmentations, den sørger for at tilføje augmentations korrekt mellem images og masks

Skriv reseurch spørgsmål til slut i introduktion og skriv også hvor mange sections vi har og hvad alle indeholder

B Logbog

02466 Project work in Artificial Intelligence and Data LOGBOOK

Josefine, s231255@student.dtu.dk
Poul, s224193@student.dtu.dk
Micki, s234866@student.dtu.dk

The main purpose of the logbook is that it serves as a tool for you to keep track of the project and document project meetings.

Project Meetings

Week 2: 13.02.25-17.02.25

We met for the first time on discord on the 13/2, talked about the initial idea we had for the project. We filled out the group contract, which specified when and how we will meet and communicate on a weekly basis, what expectations we had for the project and each other, and how we will handle problems along the way.
We concluded that we would primarily work on Wednesdays before and after the lecture, and if there is any assignment we need more time to work on, we would meet as fitted.
We also wrote to our supervisor for the first time, to try and schedule a meeting for the introduction of the project.
Lastly we setup a github repository and made sure every group member had acess to it.

Week 3: 17.02.25-23.02.25

On the 19/2 we had our first meeting with the supervisors, they gave os a presentation of the project, and gave os some idea of what we could choose to work on.
After the meeting, we met on discord and further discussed what we now know about the project, we did some initial research, and we thought that it could be interesting to focus on unsupervised learning methods, since we had much more unlabeled data then labeled data. So we drafted our first draft of the project plan, and concluded the meeting by agreeing on we will individually research some more about different ideas, models and approach.

Week 4: 24.02.25-02.03.25

This week we met both on Wednesday the 26/2 and Thursday the 27/2, as we felt like we had to make some progress to be able to complete the assignment for 5/3.
On Wednesday we started by presenting each of our research, and discussing what we had come up with, and we thought that it could actually be interesting to compare unsupervised and supervised models rather than just unsupervised method. With the supervised approach, we found a model called U-Net, which was good to segmentat medical images, so with that we started implementing the basic u-net architecture, and trained it on our labeled data. We did not get the best performance from the initial training, which suggested that there is still a lot of research to do. We decided to keep exploring the u-net architecture and modifications we can do. Also discussed if other models would be interesting to explore further.
On Thursday, we started by editing on our first draft of the project plan, since we felt like we had more knowledge about the project, to be more specific in our second draft of the project plan. When we was done, we shared our project plan with the group that has

the same project as us, since we didn't want to end up researching the same angle of the project. Also we shared our project plan with our supervisors to get some feedback before the assignment for next week.
Other then that, we also started working on the Gantt Chart, Leaning Purpose, and Project Canvas, but we wasn't done at the end of the day, so we agreed that we will each individually be working on one part of it, so we can finish on time for the assignment next week.

Week 5: 03.03.25-09.03.25

This week we had another meeting with the supervisors, were they gave us their feedback on our project plan. After the meeting on the 5/3, we finalized our project plan based on their feedback so it was ready to be delivered.
We also looked through the Gantt Chart, Learning Purpose and Project Canvas that we individually worked on, and together we made some adjustments and finished it so we could deliver it in that day.
After that we started implementing data augmentation since this would be a way to make the most use out of our limited data. The data augmentation provided performance boosts to our segmentation U-net model, however we still need to explore more about the different ways of data augmentation.
Now that we have setup the model we want to use for the supervised setting, we still need to research more about the unsupervised, so we will look into the literature of the W-net architecture, which is what we are considering for the unsupervised setting. We also decided to look further into the different data augmentation techniques.

Week 6: 10.03.25-16.03.25

This week we had a short meeting on the 12/3, to discuss our individual progress on the research we are doing. We settled on the Albumentations package for the augmentation task. And everyone was still working on experimenting with different setups/parameters for our supervised U-net model.
We decided that we would individually keep testing out some different backbones, performance metrics and configurations for the U-net model.

Week 7: 17.03.25-23.03.25

This week we met up with the other group and discussed our progress on the project. We exchanged ideas and discussed how we could best make use of the labeled data we have. We also talked about maybe using some techniques from active machine learning, that would tell us which of the unlabelled data points would provide the most performance increase if they were labelled.
We set up a lot of the experiments we would like to carry out.
For the next week we will look into how we can use the computers on DTU to carry out our experiments. We will also perform tests using our unsupervised W-net model. We will also look into adding an attention mechanism to our U-net model, that will hopefully provide a performance boost

Week 8: 24.03.25-30.03.25

Can we improve the unsupervised model?
Reading, who and what
We got the W-net model working, however as of now it hasn't showed very promising results, so we need to do some more research and experimentation on that model. We have not carried out the experiments with our supervised model yet, as we haven't gotten the DTU computers to work with our data. However we have carried out some of the experiments on our own computers, which showed promising results for the supervised

setting, where we get IOU scores of around 75% using some of our configurations. We also implemented the attention mechanism into some of the model configurations, which showed promising results

We have decided to continue trying to improve the W-net model, and attempt to get some better results with it. Additionally we will keep doing experiments with our supervised models, to find the best configurations.

Week 9: 31.03.25-06.04.25

This week we all had a few projects from other courses due, we therefore decided to skip working on this project this week and focus on our other project.

Week 10: 07.04.25-13.04.25

This week our project group went from 4 to 3, as Magnus chose to opt out of this course. We therefore spent some time discussing how to best move forward from here, being only 3 in our group. This week we all startet up training our model in different configurations, to get a better idea of where our hyperperameters should be. Furthermore we did more research, looking into the literature of past project researching the u-net structure, and how different backbone models perform. This research showed different backbone models perform similarly, but a u-net model built from scratch performs the best, which is now making us try actually building the u-net model ourself.

Week 11: 14.04.25-21.04.25

This week we decided to keep as holidays, as we feel we are keeping up well with our plans.

Week 12: 22.04.25-27.04.25

This week we decided not to meet, but each do some work ourself. We all trained a few models. Poul worked on building the u-net from scratch. Josefine did more research into which loss function to use, and if there are any other relevant literature to help with hyperparameters. Micki looked at what we have written on our project until now, and fine tuned it.

Week 13: 28.04.25-04.05.25

This week we had our feedback session with our feedback group. Before the actual session with the other group, we each read their paper and did our own feedback. We then met, just the three of us to discuss our thoughts on their paper. We were having a bit of trouble with the feedback, even after reading through the book and the slides again, as their paper was a lot different from ours, heavily discussing math, and not so much machine learning. After speaking with our supervisors, we ended up giving feedback on their overall writing style, and their delivering of tough math, as we were having a hard time understanding some of it. With this being the last week before our exam period, where we have decided to take a break from this project, and focus fully on our exams. We therefore spend this week wrapping up this portion of the project, with tying up lose ends, and writing down where we left off both the coding part and the writing part.

Week 18: 26.04.25-01.05.25

We met once this week, at the supervisor meeting, and shortly after. Because some of us still had exams this week, we decided push work on this project back a bit more, and pick back up week 19.

Week 19: 02.05.25-08.05.25

With this being the first official week back, we spend the first two days getting back into the project; rereading code and the report. We also spend a bit more time than usual, preparing for our supervisor meeting, wanting to fully utilize our time.
After our meeting, we spend the next three days figuring our how to choose our hyperparameters, implementing the code and writing about it in our report. We took Friday and Saturday off, and worked further with different test of hyperparameters.

Week 20: 09.05.25-15.05.25

This week was a bit shorter than normal because of some family stuff. But our focus of this week was hyperparameters and augmentation. When finishing this, and arguing our choices in the report, we are at a point where we can run our final training and test, and start writing on the results and discussion.

Supervisor Meetings

Week 3: 17.02.25-23.02.25

Our supervisors presented the project for us, and we talked about how we envisioned the project going. We also asked them to read our project plan, so we can finalize our draft this week.
We agreed on meeting every other week. Until next meeting we will do more research, and get a better idea on how to approach the problem more specifically

Week 5: 03.03.25-23.02.25

We got started working on our project and presented the ideas about which models we would implement. They agreed with our plan of using a unet architecture, as well as our proposition of trying to develop a model that could work in the unsupervised setting. Again we asked them to look at our project plan, and give feedback, so we can get the final plan delivered, with the changes from the feedback
Our plan for the next week is to use data augmentation on our limited data that also has corresponding segmentation masks. We will continue working on developing the model and get initial training setup.

Week 7: 17.03.25-23.03.25

This week we met up with the second group during the supervisor meeting, and discussed how far both groups were. We shared ideas about the models and techniques we would use. For example we discussed that we could maybe try to get additional labeled data, and finding out which specific data samples would be most useful by using techniques we have learned from active machine learning
For the next week we will try to implement our unsupervised model w-net, and do some basic testing to get it to work. We will also set up the experiments for our supervised models, that we will use to find the final model to train

Week 9: 31.03.25-06.04.25

We had trouble this week with setting up the HPC computers on DTU, and we only carried out a few more experiments. The supervisors gave us advice on how we can setup the HPC computers and who to get in contact with.
For the next week we will continue to carry out experiments, and researching our models further.

Week 10: 07.04.25-13 13.04.25

This week we did more research on the literature on image segmentation. We discovered some different loss functions that we want to implement and try out, as our previous

experiments didn't provide such good results. Our supervisors agreed that it would be a good idea to look into different loss functions, and trying to find the one that works best for our specific task.
For the next week we will try to focus more on getting the unsupervised and semisupervised models to work

Week 13: 28.04.25-04.05.25

We implemented the different loss functions, and have done basic experiments with all the configurations of models, and we are now ready to do the real experiments with a lot of training for each. Besides the supervised models, we have experimented a lot with the unsupervised methods, and come to the conclusion that we would have to put a lot of our energy into getting it to work. With our current experiments we have gotten very bad results, and after some research, we came to the conclusion that w-net is not suitable for our task of medical image segmentation. After presenting our results, and our idea for our supervisors, they gave the feedback to focus purely on supervised, like we also thought about, but to still keep the model in the project, in a smaller capacity. We had not thought about this, but it made sense, as we used a lot of energy on it, so we should still write what worked and what did not, and what we learned from it. Lastly we spoke a little about our feedback group, as we were having a bit of trouble with what to give feedback on, as their project was a lot different from ours. This helped a lot, and our supervisors gave some good pointers on how best to approach.
From here our exams begin, so we plan so focus solely on them, and get back to this project a week or so before the 3 week period begins.

Week 18: 26.04.25-01.05.25

These past few weeks we focused on our 13 week course exams, and didn't work much on the project. We had a very short meeting with the supervisors, where we agreed to push the meeting a week, such that we could get started on our project again before we met.
For the next week we will get back into the project and make a concrete plan on how we want to structure this 3-week period.

Week 19: 02.05.25-08.05.25

This week we did our first official supervisor meeting, kick starting the three week period. We spend Monday and Tuesday slowly getting back into the project, figuring out exactly how far we were and where we should pick back up. We talked with our supervisors about this, specifically if we should keep working on our supervised U-Net model, or if it would be beneficial to look into other models, to compare. We agreed it would be best to focus solely on our U-Net model, and have that fully finished, before looking into other models. If we do finish the U-Net model, we can always look into an unsupervised model. We also talked a lot about hyperparameter optimization, where we had not chosen an optimal method of optimizing. Lastly we spoke a bit about the oral exam, and if our current project would be sufficient. We agreed what we are working on currently, should be quite sufficient if we tie it all together correctly.
For the next week, we will focus on hyperparameters, using the correct methods. We will also focus on the writing process, and get our discussed point written into the project.

Week 20: 09.05.25-15.05.25

This week the other group was also at the meeting, so we spoke a bit with them about some of our struggles. Because our results from last week, regarding the hyperparameters did not yield significant results. The big focus this time was augmentations, as this

was kind of the last place we could put our focus to improve our model. We found out we should not augment our test data, which did improve our model. We also talked a bit about our report. How much method, and what should be included. But we ended up agreeing they should read it until next time, and give some feedback there.