UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Illia Tsiporenko

Going Beyond U-Net: Assessing Vision Transformers for Semantic Segmentation in Microscopy Image Analysis

Master’s Thesis (30 ECTS)

Supervisor(s): Dmytro Fishman, PhD
Pavel Chizhov, MSc

Tartu 2024

Going Beyond U-Net: Assessing Vision Transformers for Semantic Segmentation in Microscopy Image Analysis

Abstract:
Segmentation is one of the crucial steps in biomedical image analysis. Many approaches were developed over the past decade to segment biomedical images, ranging from classical segmentation algorithms to advanced deep learning models, with U-Net being one of the most prominent. Recently, a new class of models has appeared — transformers, which promise to enhance the segmentation process of biomedical images. We explore the efficacy of the well-established U-Net model and newer transformer-based models, including UNETR, Segment Anything Model, and Swin Transformer, across various image modalities such as electron microscopy, brightfield, histopathology, and phase-contrast. Additionally, we identified several limitations in the original Swin Transformer architecture and addressed those via custom modifications to the original model to optimise its performance. Our results indicate that these modifications improve segmentation performance compared to the classical U-Net model as well as to the original unmodified Swin. While results show that transformer models hold promise, especially in handling complex image structures, our practical experience shows that deploying these models can be difficult. This work compares popular transformer-based models against U-Net and shows that with thoughtful modifications, the efficiency and applicability of transformer models can be enhanced, paving the way for their future integration into microscopy image analysis tools.

Keywords:
Deep learning, Neural networks, Image segmentation

CERCS: T111, P176

U-Netist Edasi: Masinnägemise Transformerite Semantilise Segementeerimise Hindamine Mikroskoopia Pildianalüüsis

Lühikokkuvõte:

Segmenteerimine on biomeditsiinilise piltdiagnostika üks olulisemaid osi. Viimase kümnendi jooksul on biomeditsiiniliste piltide segmenteerimiseks välja töötatud palju lähenemisviise, alates klassikalistest segmenteerimisalgoritmidest kuni täiustatud süvaõppe mudeliteni nagu näiteks U-Net. Hiljuti on välja tuldud uue mudeliklassiga – transformeritega, mis lubavad suurendada biomeditsiiniliste piltide segmenteerimistäpsust. Magistritöös uuritakse U-Neti mudeli ja uuemate transformeripõhiste mudelite, sealhulgas UNETR, Segment Anything Model ja Swin Transformer, täpsust erinevate biomeditsiiniliste piltide modaalsuste puhul nagu elektron-, helevälja- ja faaskontrastimikroskoopia ning histopatoloogia. Lisaks tuvastatakse algses Swin Transformeri arhitektuuris mitmeid piiranguid ning pakutakse välja muudatusi mudelis selle täpsuse parandamiseks. Töö tulemused näitavad, et need modifikatsioonid parandavad segmenteerimistäpsust nii klassikalise U-Neti mudeli kui ka algse Swin mudeliga võrreldes. Kuigi tulemustest ilmneb, et transformerid on paljutõotavad, seda eriti keerukate pildistruktuuride käsitlemisel, näitab meie praktiline kogemus, et nende mudelite kasutuselevõtt võib-olla keeruline. Selles töös võrreldakse populaarseid transformeripõhiseid mudeleid U-Netiga ja näidatakse, et läbimõeldud muudatustega saab transformerite täpsust ja rakendatavust suurendada, mis sillutab teed nende integreerimiseks mikroskoopia pildianalüüsi meetoditesse.

Võtmesõnad:
Süvaõpe, närvivõrgud, pildi segmenteerimine

CERCS: T111, P176

Contents

1 Introduction

Identifying objects in microscopy images is one of the first steps for biological and medical studies [14]. The shift towards digital microscopy has considerably increased the volume of image data, creating a demand for tools capable of swiftly and accurately processing these images [3]. As the volume of digital images grows, manual analysis becomes impractical, highlighting the importance of automated approaches.

Given the need for automated tools due to the increase in digital image data, this study aims to compare traditional segmentation methods, notably Convolutional Neural Networks (CNNs), against transformer-based approaches. While CNNs have established themselves as solid and robust approaches for medical image segmentation [39], their limitations in capturing long-range dependencies and complex spatial relationships in images push us to explore alternative methods. To address these challenges, we turn to transformer-based models characterised by their use of attention mechanisms [48]. These mechanisms allow transformers to weigh the importance of different parts of an image, potentially improving their ability to understand complex patterns and relationships.

In this study, we examine and assess a selection of models to understand their performance in digital medical image segmentation. Among the traditional approaches, U-Net, a CNN known for its utility in medical image segmentation, serves as our baseline.

Transitioning to transformer-based models, we explore the Swin Transformer [34]. This model introduces a hierarchical structure, enabling efficient processing of images at varying scales, which is key for detailed feature capture within microscopy images.

Additionally, we assess UNETR [17], which is explicitly adapted for 2D segmentation in our work. UNETR uniquely merges the practical and well-known aspects of U-Net with the advanced capabilities of transformer architectures for segmentation tasks.

Furthermore, our study evaluates Segment Anything (SAM) [27], a foundational model tested in its default configuration. As suggested by its name, SAM is designed to segment a diverse range of objects, demonstrating its potential applicability across various segmentation tasks without fine-tuning.

By studying these models, we aim to reveal the advantages and challenges of both traditional and modern segmentation methods, particularly for analysing the images found in digital medical microscopy.

2 Background

Image segmentation is one of the key aspects when working with medical images, as it helps to outline important parts of the image. Effective segmentation methods and techniques provide accurate and fast results compared to manual techniques. In this section, we give an overview of concepts of image segmentation, ranging from classical algorithms to advanced deep learning methods.

2.1 Segmentation

In computer vision, image segmentation is a process of assigning a label to every pixel in an image. Simply put, this process involves partitioning an image into multiple segments, or sets of pixels, typically to identify objects or boundaries — "segments" within images. Segmentation can be classified into three categories:

- **Semantic segmentation** aims to categorise each pixel in an image without differentiating between separate objects of the same class. For instance, in an image with multiple cars, all cars would be labelled with the same class without distinguishing between individual cars.
- **Instance segmentation** aims to categorise each pixel in an image, as well as differentiate between individual objects of the same class. Following the previous example, each car would be assigned a unique identifier, allowing for individual recognition and analysis.
- **Panoptic segmentation** takes this process a step further combining the principles of semantic and instance segmentation to provide a comprehensive image analysis. It assigns a unique label to every pixel in an image, categorising them into broad classes while distinguishing between individual instances within the same class.

2.2 Classical segmentation algorithms

Classical segmentation algorithms are foundational techniques in image processing and computer vision that partition images into segments based on predefined criteria such as pixel intensity, colour, or texture. Before deep learning, classical segmentation algorithms were popular methods for categorising images into parts. Examples include Thresholding [37], which splits an image by brightness, and Region growth [1], which groups similar pixels. These techniques help identify and separate objects or areas in an image.

One of the most straightforward segmentation techniques is Thresholding. It involves segmenting an image based on the intensity values of the pixels. By selecting a threshold value, the image is divided into foreground and background, where pixels above the threshold are assigned to one segment, and those below are assigned to another. This method works well for images with high contrast between the segments but struggles with more complex scenes.

Region-based segmentation methods, such as Region Growing, focus on grouping pixels that satisfy a predefined similarity criterion. For example, the Region Growing algorithm starts with a set of seed points and expands these seeds by including neighbouring pixels with similar attributes (e.g., intensity or texture). This method is particularly effective for identifying objects within an image with a clear distinction in pixel characteristics.

Clustering Algorithms, like K-means [25], are unsupervised learning methods that group pixels into clusters based on their feature similarity without prior knowledge of the segment number or type. These algorithms analyse the entire image's pixel distribution and allocate each pixel to the nearest cluster centre based on colour, intensity, or texture features.

Another notable classical segmentation technique is the Watershed Algorithm [49], which can separate touching objects within an image. Conceptually, the algorithm visualises an image as a topographic landscape, with the brightness of pixels representing elevation. The process begins by identifying markers in the image, which act as seeds for future segments. These markers can be chosen based on intensity minima or through manual selection, representing the starting points of different segments.

As the algorithm simulates flooding this landscape from the markers, it constructs "dams" whenever "water" from different "basins" is about to merge, effectively delineating the boundaries between distinct regions. This flood simulation continues until the entire landscape is "submerged", with the resulting "dams" forming the watershed lines that segment the image.

2.3 Deep Learning

Even though classical segmentation methods are easy to apply and work well for basic tasks, Artificial Neural Networks (ANNs) take things to the next level by better handling complicated images [53]. In this work, we will explore ANNs and how they bring more advanced solutions to image segmentation. ANNs are models that mimic the behaviour of actual biological neural networks found in human and animal brains.

At the core of deep learning is an ANN model, which is a function that takes inputs, such as images or text and gives outputs, e.g. classifications or predictions. This model is defined by parameters — called weights that the learning algorithm adjusts to improve the model's performance. Adjusting these parameters is guided by a loss function, which quantifies how far the model's predictions are from the actual values.

A deep learning model can be thought of as a complex mathematical function $\mathcal{F}$ dependent on a set of parameters W that takes inputs x and produces outputs $\mathbf{y}$. Mathematically, the model can be expressed as:

$$\mathbf{y} = \mathcal{F}(x|W) \tag{1}$$

The loss function $\mathcal{L}$ measures the difference between the outputs $\hat{\mathbf{y}}$ of the model and the actual targets $\mathbf{y}$. This function is critical for evaluating the model's performance, guiding the optimisation process to improve its performance by minimising these discrepancies. A general formula for the loss function can be expressed as:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) \tag{2}$$

This formula represents the loss function $\mathcal{L}$ which calculates the sum of the discrepancies measured by L between the predicted outputs $\hat{\mathbf{y}}$ and the actual targets $\mathbf{y}$, across all n samples in the dataset.

Deep learning algorithms adjust the model parameters W to minimise the loss function $\mathcal{L}$ This adjustment is done using the gradient of the loss function concerning each weight and bias, a computation efficiently performed through the calculus chain rule. One of the common algorithms used to update the network parameters is Stochastic Gradient Descent(SGD) [41]. The update rule can be expressed as follows:

$$\Delta w = -\eta \nabla_w \mathcal{L} \tag{3}$$

where Δw is the change applied to the weight w, η is the learning rate (a small, positive parameter that controls the step size of the update), and $\frac{\partial \mathcal{L}}{\partial w}$ is the partial derivative of the loss function $\mathcal{L}$ with respect to the weight w.

2.4 Multilayer Perceptron

One of the first types of ANNs developed was the Multilayer Perceptron (MLP), also known as a Feedforward Neural Network (FFN) [42]. This model is rooted in the concept of the perceptron [40] and the way the animal brain functions. MLPs consist of layers of these perceptrons, or artificial neurons, where each neuron is connected to others through edges, mimicking synapses and neurons in a biological brain. Each neuron receives signals from the neurons in the previous layer, processes these signals internally, and transmits the output to subsequent neurons (see Figure 1). The connection strength, also known as the weight between neurons, can enhance or inhibit the signal transmitted between them.

Activation functions are present in the model to learn complex patterns in the data, introducing non-linearity. The output of each neuron in an MLP is passed through a non-linear activation function. The most common ones are Rectified Linear Unit(ReLU) in eq. (4), Tanh in eq. (5) and Sigmoid in eq. (6) functions. Among all of them, ReLU

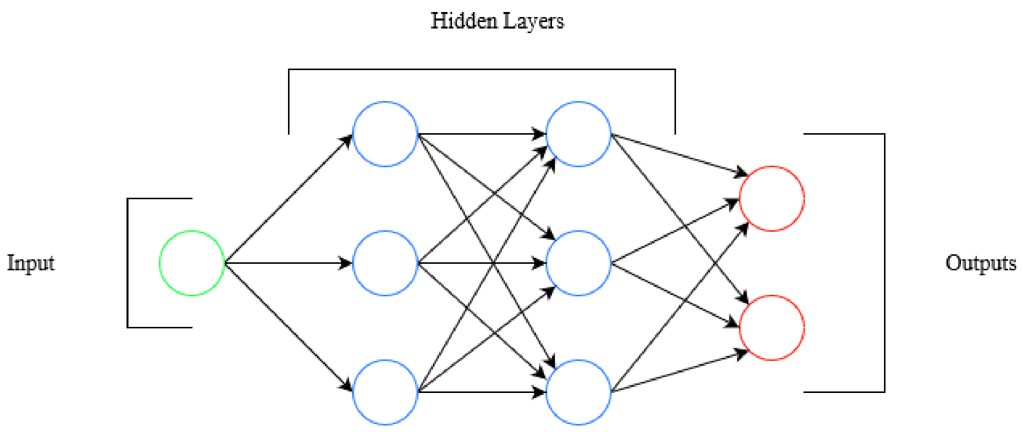

Figure 1. Example of Feed-Forward Artificial Neural Network. Circles represent artificial neurons, which are connected to each other. Green one — is an input to the network, blue ones — are the neurons in the hidden layers of the network, and the red ones — are the output of the network

usually leads to faster convergence of the loss and thus is broadly used as a common activation function in different ANN architectures [28].

$$\mathrm{ReLU}(x) = \max(0, x) \tag{4}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{6}$$

The behaviour of an individual neuron within an MLP layer is mathematically described as follows:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{7}$$

where x_i are the inputs to the neuron, w_i are the weights associated with each input, b is the bias, f is the activation function, and y is the output of the neuron.

2.5 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are designed to efficiently process images by mimicking how we see things [31]. They apply convolutional filters [31] across an image to pick up basic patterns like lines and curves at the start. As the image data moves deeper into the network, these basic patterns are combined to recognise more complex

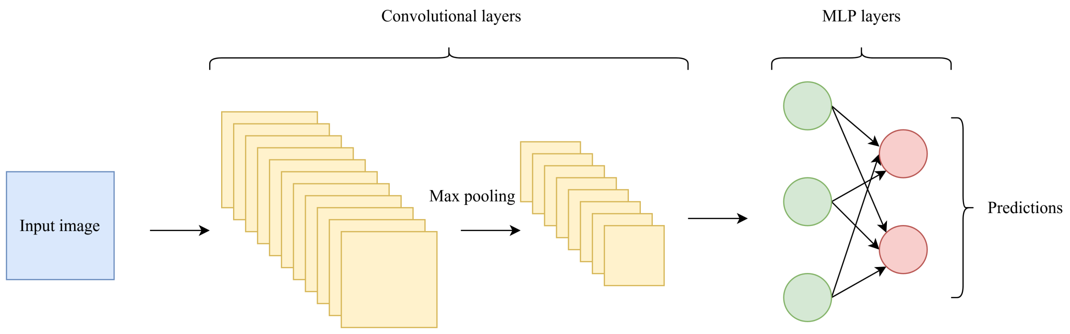

Figure 2. Example of Convolutional Neural Network. The input image is passed to the network and processed by convolutional layers, which produce feature maps. Max pooling is applied to reduce the dimensionality and keep relevant features. Afterwards, the feature maps are flattened and passed into MLP layers and the final prediction is obtained in the end

features, such as shapes or objects relevant to the task, like identifying what's in a picture (see Figure 2).

These filters create feature maps, essentially new versions of the image highlighting specific features. As we go deeper into the network, the number of feature maps increases, capturing more details, but their size gets smaller due to the downsampling process [31]. This process makes the network faster and helps it focus on the most essential parts of the image.

When figuring out what's in an image (classification), the CNN ends with a layer that takes all it has learned from the feature maps to make a final guess. For tasks where the network needs to output an image of the same size as the input, like in image segmentation, there are particular ways to resize the feature maps back to the original image size, like bilinear interpolation.

Training CNNs often results in gradient vanishing [5], where the value of the gradient of a loss function diminishes with each layer during backpropagation, leading to minimal or no updates in the weights of the earlier layers. This issue can drastically slow down the learning process or result in the network failing to converge. One clever solution is using residual blocks, as seen in the ResNet [28] (see Figure 3) architecture.

These blocks help the training process by allowing the initial input to skip some layers and be added back into the output of later layers. This helps the network to learn better and faster [28].

Besides that, it is a common thing to use the normalisation function in a network to stabilise and accelerate the training of neural networks. Among these, Batch Normalization [23] and Instance Normalization [46] are commonly used in many CNN architectures.

Batch Normalization (Batch Norm) operates across the batch dimension, normalizing

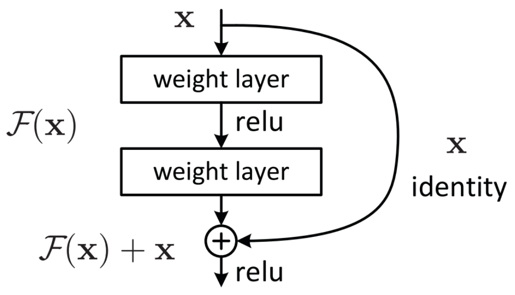

Figure 3. Example of Residual Block [28]

the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation. Additionally, it introduces two trainable parameters, scale and shift, to enhance the network's representational ability. This process helps to mitigate the issue of internal covariate shift, where the distribution of network activations changes during training, thus speeding up convergence and enabling the use of higher learning rates.

Instance Normalization (Instance Norm), on the other hand, is primarily used in style transfer applications. It normalises the data across each channel in each training example independently. By doing this, Instance Norm can preserve the contrast of each image, making it particularly effective for tasks where the relative distribution of features within an instance matters more than across the batch.

Both normalization techniques adjust the data in different ways to suit various types of neural network applications. Batch Norm is more universally applicable and beneficial in deep learning models for tasks like image classification. In contrast, Instance Norm finds its niche in style transfer and tasks emphasising per-instance data processing.

2.5.1 U-Net

Transitioning from general CNN architectures, it becomes apparent that traditional approaches may not fully meet the image segmentation tasks. Image segmentation requires not only identifying objects within an image but also precisely outlining their shapes. To address this, architectures like U-Net [39] have been developed for segmentation, offering a structured way to maintain the global context and fine details necessary for accurate segmentation.

U-Net is a CNN that was initially developed for biomedical image segmentation tasks. While traditional CNNs excel in identifying and classifying entire images into distinct categories, the process involves reducing the image size through its layers. However, segmentation tasks demand the network to generate an output — a segmentation mask, that maintains the original dimensions of the input image. U-net provides an elegant solution introducing encoder-decoder type architecture (see Figure 4).

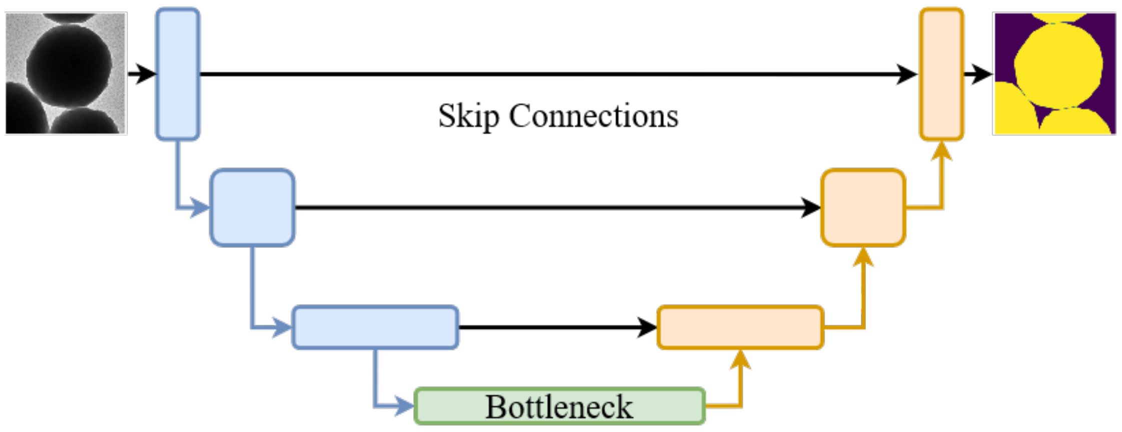

Figure 4. Representation of U-Net architecture [39]. The model consists of the encoder (blue blocks), which extracts necessary features from the input image by applying convolutional operations and max pooling (blue arrows), and the decoder (orange blocks), which applies transposed convolution operations (yellow arrows) to decrease the feature dimensionality and scale them up to original size of the input image. Black arrows represent skip connections.

The encoder part consists of convolutional blocks, which contain convolutional layers, normalisation layers, and activation function. This is followed by max pooling that contracts the image, essentially producing feature maps with a greater number of channels. The decoder part is symmetrical to the encoder. It consists of the same convolutional blocks, but a transposed convolutional layer is used instead of max pooling to upsample the feature maps produced by the encoder, decreasing the number of channels and increasing the resolution. Feature maps from the encoder concatenated with the ones from the decoder via skip connections, bringing local information to the model‘s output.

2.5.2 UperNet

UperNet [50] is designed as a decoder module for segmentation tasks, which handles hierarchical and multi-scale image features derived from any hierarchical encoder. This model is particularly adept at integrating features across different scales due to its comprehensive architecture, which includes the Pyramid Pooling Module (PPM) [18] and a Feature Pyramid Network(FPN) [32] — series of up-sampling and convolutional layers that refine the segmentation outputs (see Figure 5).

UperNet begins by processing the multi-scale feature maps generated by the encoder using FPN. These features range from high-resolution details to coarser, contextual information, which is crucial for capturing both precise edges and broader textural patterns within an image. The core of UperNet’s functionality lies in its ability to fuse these diverse scale features effectively. This fusion process not only preserves the high-

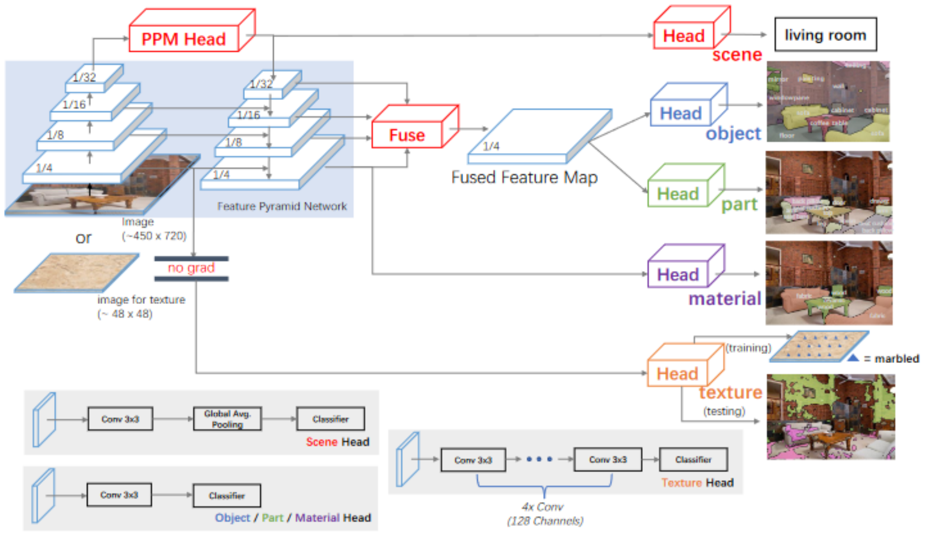

Figure 5. UperNet architecture [50]. The model consists of a hierarchical backbone and Feature Network Pyramid and Pyramid Pooling Module. FPN and PPM effectively utilise the features produced by the backbone to produce feature maps with a rich representation of the input image. These feature maps are passed to separate heads, each one designed to handle a specific task.

resolution details but also enhances the model's sensitivity to various spatial hierarchies present in the input data.

Following the feature fusion, UperNet employs the PPM, a module designed to aggregate contextual information at multiple scales. The PPM works by dividing the feature map into varying-sized regions, thereby creating a 'pyramid' of pooled features. Each level of this pyramid:

1. Processes its designated region through pooling operations such as average or max pooling, which summarizes the essential features within each area, thereby reducing spatial complexity while retaining critical information.

2. The pooling operation at each pyramid level helps to compress the spatial information, which emphasizes the most relevant attributes by focusing on broad to fine details, thus ensuring that all levels of context from the image are captured.

The pooled features from each level of the pyramid are then up-sampled to match the highest resolution level within the feature map. This step is crucial as it allows the integration of detailed textural information with the broader, contextual information that the lower levels of the pyramid capture. The up-sampling process ensures that the global context captured at various scales is uniformly integrated, maintaining the integrity of spatial details.

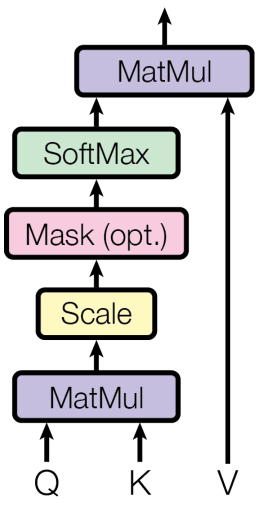

Figure 6. Dot-product self-attention [48]

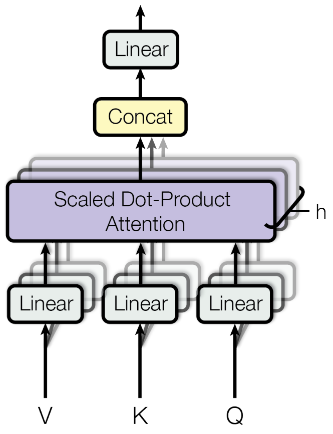

Figure 7. Multi-head self-attention [48]

Finally, the outputs from the PPM and the initially fused feature maps are combined. This combination leverages both the detailed local information and the global contextual information, which is essential for producing a coherent and finely segmented output. The combined feature map then undergoes further processing through additional convolutional layers and up-sampling steps. These layers refine the feature integration, enhancing the model's ability to delineate precise object boundaries and improve the overall accuracy of the segmentation task.

2.6 Attention-based Neural Networks

While U-Net efficiently captures spatial hierarchies and contextual information through convolutional operations and skip connections, recently developed attention mechanisms offer a complementary approach by explicitly modelling relationships between all parts of the input data, regardless of their spatial distance. This ability to focus on specific features across the entire image allows attention-based networks to handle complex patterns and dependencies.

Attention-based Neural Networks utilise an attention mechanism [4] at its core, mimicking human and animal cognitive attention. This mechanism can focus on specific and relevant parts of the input, ignoring less valuable parts of it. These networks have considerably improved the efficiency and accuracy of models in interpreting complex data patterns.

The core idea of attention mechanisms in neural networks is to allow the model to allocate its focus variably across different segments of the input data. For image segmentation tasks, the network can concentrate on specific areas of an image that

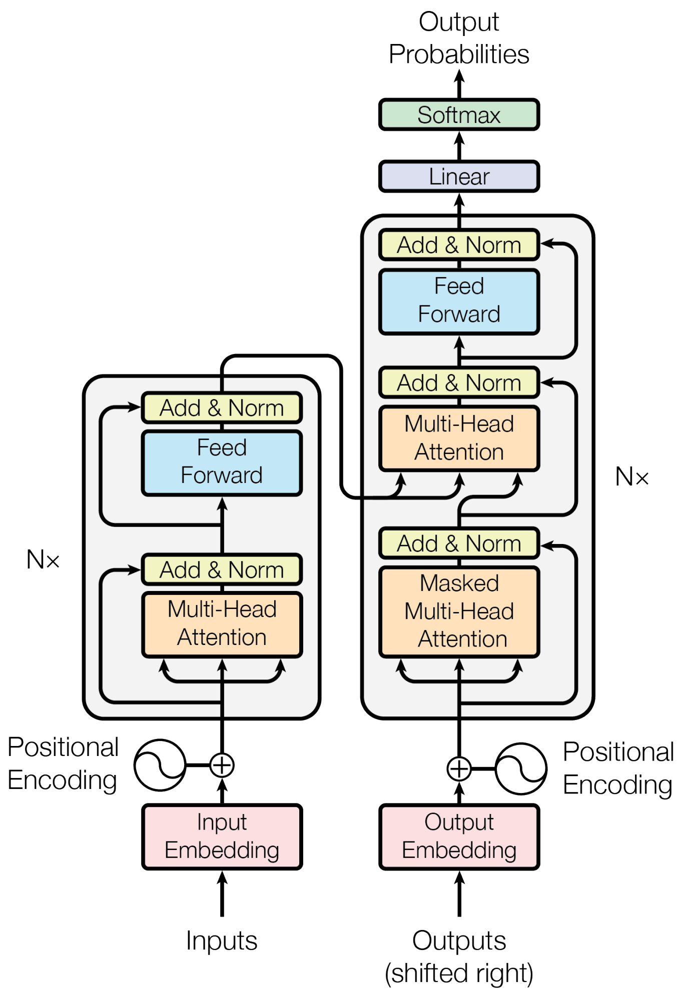

Figure 8. Transformer architecture, consisting of encoder and decoder parts [48].

are more relevant to the segmentation task at hand, enhancing the precision of the segmentation.

One of the most influential architectures incorporating attention mechanisms is the Transformer model (see Figure 8), introduced by Vaswani *et. al.* [48]. Initially designed for language processing tasks, its core idea has been adapted for use in various image processing tasks.

At its core, the Transformer model uses scaled dot-product self-attention (see Figure 6), allowing it to assess the importance of different parts of the input data [48]. The transformer consists of an encoder and decoder stacked N times and operates on a sequence of tokens. In the encoder part, the input sequence is linearly projected into queries Q, keys K of dimensionality d_k, and values V of dimensionality d_v. The next step is calculating attention: the dot-product between keys and queries is calculated,

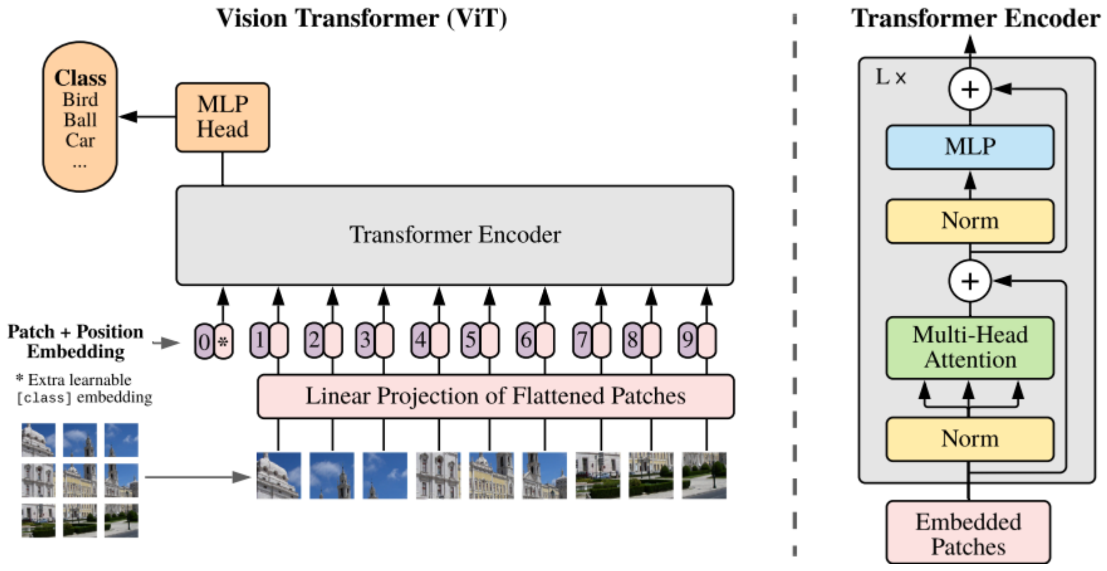

Figure 9. Vision Transformer architecture [11]. The input image is split into non-overlapping patches of the same size, which are then linearly projected and passed into the transformer encoder. The output of encoder is passed to MLP head to obtain final predictions

representing the similarity between each query and key (attention weights). To prevent the values of the dot product from becoming excessively large, attention weights are divided by the square root of dimensionality d_v. The final step is applying the softmax function to produce the probability distribution over the whole sequence and compute the weighted sum of the value vectors, producing the output representing the focus of attention within the sequence. The attention can be expressed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{8}$$

After computing the attention, the output passed through the normalisation layer and feed-forward network, followed by the final normalisation layer. For the model to be able to distinguish between the tokens‘ positions in a sequence, positional encoding is introduced in the model‘s architecture. Each input token is mapped to a positional encoding vector, added to the input afterwards, before linear projecting. There are multiple ways to create positional encodings [16]; the authors chose sine and cosine functions of different frequencies. These functions can model cyclical patterns, enabling the Transformer to capture and learn these repetitions quickly.

The decoder operates on similar principles, employing the exact attention mechanism with a key difference: a mask is incorporated to prevent the decoder from accessing future tokens, ensuring that predictions are made based on available information only.

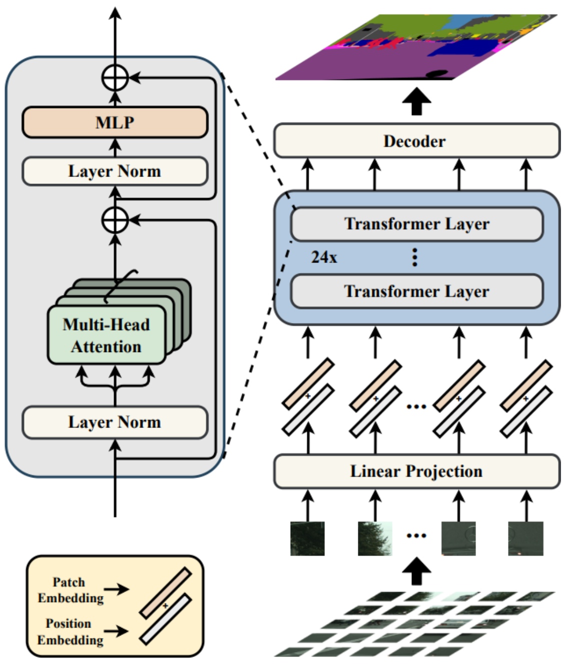

Figure 10. SETR Architecture. The input image is partitioned into non-overlapping patches, linearly projected and processed by the ViT encoder. The features produced by the encoder is upscaled through the decoder part of the network [54].

Inputs to the decoder's (MHSA) include outputs from the encoder. Generating the output sequentially, one token at a time, the decoder references the encoder's output at each step to construct a representation of the input token sequence. This representation guides the generation of subsequent tokens in the output sequence, continuing until the complete sequence is formed.

The authors of the paper discovered that projecting the queries, keys, and values into h distinct linear spaces and then calculating attention separately in each space in parallel proved beneficial [48]. Afterwards, each output is concatenated back, resulting in the final output values (see Figure 7).

2.6.1 Vision Transformers

Initially designed for handling sequential data like text, the Transformer model has found a new application in image analysis through Vision Transformers (ViT) introduced by Dosovitskiy *et. al.* [11], showing that its core principles are practical across different data types. By applying attention mechanisms to visual content, ViT offers a novel approach to understanding images, highlighting the potential for cross-domain applications of Transformer architectures.

The first Vision Transformer (ViT) (see Figure 9) was designed for image classifi-

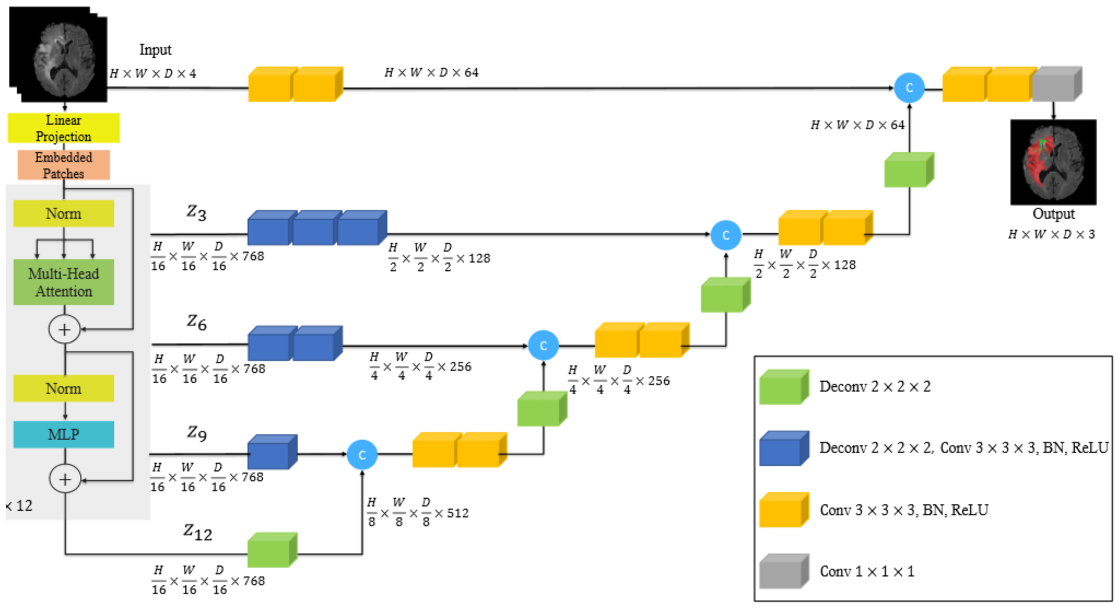
Input
$H \times W \times D \times 4$
$H \times W \times D \times 64$
Linear Projection
Embedded Patches
Norm
Multi-Head Attention
Norm
MLP
× 12
z_3
z_6
z_9
z_{12}
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2} \times 128$
$\frac{H}{4} \times \frac{W}{4} \times \frac{D}{4} \times 256$
$\frac{H}{8} \times \frac{W}{8} \times \frac{D}{8} \times 512$
$H \times W \times D \times 64$
Output
$H \times W \times D \times 3$
Deconv 2 × 2 × 2
Deconv 2 × 2 × 2, Conv 3 × 3 × 3, BN, ReLU
Conv 3 × 3 × 3, BN, ReLU
Conv 1 × 1 × 1

Figure 11. UNETR architecture[17]. The model consists of twelve stacked transformer encoders, serving as a backbone and the U-Net-like decoder that processes features generated by the backbone via transposed convolutions and convolution blocks.

cation tasks and demonstrated state-of-art results [11]. The core mechanisms that we have described above were adopted for vision tasks. Instead of processing the full image, as in CNNs, Vision Transformer splits the input image into non-overlapping patches, which are then linearly projected and, essentially, can be treated as just a sequence of tokens. Alongside, projected patches are summed up with trainable positional encodings and extra learnable class embedding. ViT utilises the exact N times stacked transformer encoder architecture as the original Transformer described in the previous section. The final output from the encoder is then passed through a fully connected classification layer to obtain the final predictions.

Based on the idea of ViT, the SETR model was introduced by Sixiao Zheng *et. al.* [54]. SETR segments the input image into non-overlapping patches, treating them as sequences and employing a ViT encoder to capture long-range dependencies and contextual information within the image. The decoder of the network consists of convolutional layers that process the feature produced by the encoder and bilinear interpolation operations to upscale the feature maps to the original dimensions of the input image (see Figure 10).

UNETR model, introduced by Hatamizadeh *et. al.* [17], is another model that utilises ViT as an encoder. It was developed for 3D segmentation tasks in the biomedical domain. The architecture of UNETR follows principles similar to those of U-Net (see Figure 11). It consists of the encoder part, a ViT with twelve stacked transformer encoders (stages),

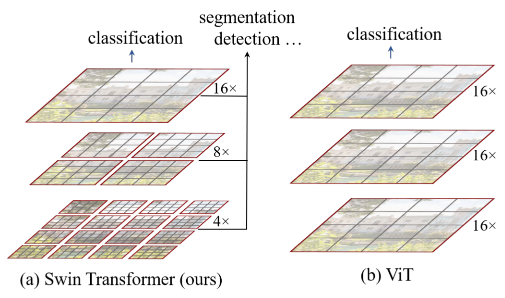

Figure 12. Hierarchical structure of Swin Transformer compared to ViT [34]

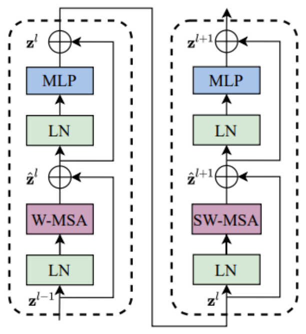

Figure 13. Combination of two Swin Transformer blocks, one with window MHSA, followed by the second block with shifted window MHSA [34]

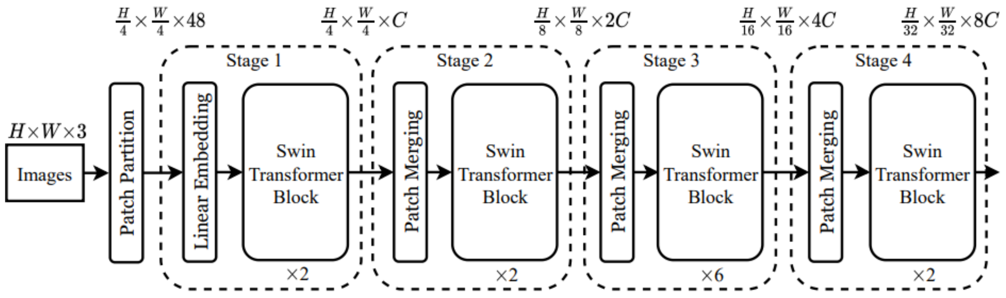

Figure 14. Swin Transformer architecture. It consists of four stages. Each one consists of Swin Transformer Blocks that utilise window MHSA and shifted window MHSA, followed by patch merging operation. The input image is partitioned into patches, linearly projected and passed into the transformer [34].

and the decoder part, which incorporates transposed convolution layers and convolutional blocks described previously. The output from each third stage of ViT is passed through convolutional blocks and concatenated with corresponding feature maps in the decoder.

Even though ViT and its modification showed commendable performance in various vision tasks back in the day, limitations in a big patch size of 16×16 and the concept of patch embeddings restricting the model from operating on various image sizes pushed researchers to find new approaches in applying transformers in computer vision tasks.

Addressing the limitations encountered with Vision Transformers (ViT) in tasks requiring finer local details, the Swin Transformer model introduced by Liu *et. al.* [34] provides an innovative approach that enhances the model's applicability to segmentation and detection tasks and is a universal encoder for computer vision tasks. The model‘s architecture consists of stages built of Swin Transformer Blocks and Patch Merging operations (see Figure 13 and Figure 14).

Central to its design is hierarchical representation, achieved through smaller, non-

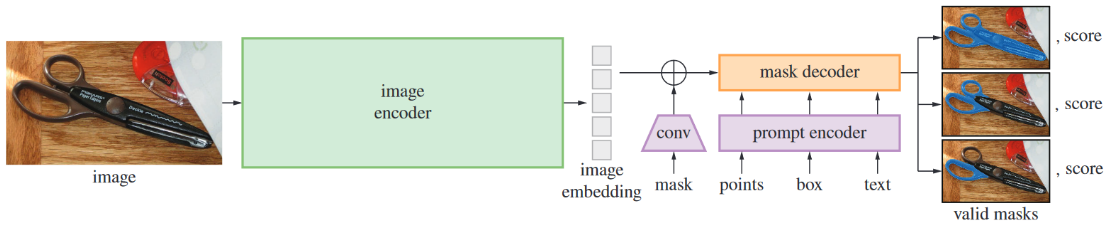

Figure 15. Segment Anything architecture.

overlapping patches and a novel window-based self-attention mechanism. The input image is split into non-overlapping small patches, usually of size 4×4, which allows the model to learn local information. Afterwards, the attention between patches is computed within windows. To maintain the relationship and context between the pixels within the different windows, the shifting window operation was introduced[figure]. The windows are shifted each second Swin Transformer block diagonally. To capture a local and global representation of the image, the Patch Merging operation was introduced. Patch Merging essentially merges small patches into bigger ones in a hierarchical style (see Figure 12). Passing the input through the Swin Transformer blocks and applying Patch merging for N stages results in the final encoded representation of an input image that can be used for various computer vision tasks.

The original approach for segmentation with Swin Transformer utilises UperNet [50] as a decoder part of the network. The input image is passed through the Swin Transformer, which produces the encoded representation, which is fed into the decoder to obtain the final prediction. Though the UperNet itself is a separate model designed for segmentation and scene understanding, only part of its architecture, consisting of PPM [18] and FPN [32], is used in the segmentation process.

2.6.2 Segment Anything

Segment Anything Model (SAM) [27] has recently emerged as a noteworthy advancement in image segmentation. SAM's robust training on the extensive SA-1B [27] dataset that contains 1 billion masks across 11 million images showcases a substantial effort towards enhancing and uniforming segmentation techniques in computer vision. The model has a heavyweight image encoder, which outputs dense image embeddings. These embeddings are further passed into the mask decoder module alongside encoded prompts, which can be either points or bounding boxes (see Figure 15). SAM operates in a portable mode, facilitating a user-centric approach and enabling segmentation tasks with various prompts like foreground/background points and rough boxes. The input image is passed into the model alongside the bounding box prompts or point pormpts highlighting the object of interest. Another option that SAM provides for segmentation is an automatic mode. The process functions by creating a grid of points overlaying the input image. From each point, SAM can generate multiple masks. The duplicates are removed via non-maximal suppression [20]. Further enhancements to both the quality and quantity of the masks can be achieved by employing additional techniques, such as making predictions on various cropped sections of the image or refining the masks post-prediction to eliminate small, isolated areas and fill in gaps.

2.7 Metrics

In the field of semantic segmentation, two commonly used metrics for evaluating model performance are the F1 score and the Intersection over Union (IoU) score. Both metrics provide insights into the accuracy and precision of the segmentation results.

The F1 score is a statistical measure used to evaluate the accuracy of a test. It represents the harmonic mean of precision (p) and recall (r), incorporating both measures into a single score. Precision refers to the proportion of positive identifications that were correct, such as accurately segmented cells in a microscopy image, divided by the total number of elements labelled as positive. On the other hand, Recall measures the proportion of correct positives that were correctly identified, showing how many actual positives were captured by the test. The F1 score provides a balanced view contribution of recall and precision to the accuracy of the model. The F1 score can be mathematically expressed as follows:

$$\mathrm{F1} = 2 \cdot \frac{p \cdot r}{p + r} \tag{9}$$

The Intersection over Union (IoU) score, also known as the Jaccard index, evaluates the accuracy of a segmentation model. It measures the overlap between the predicted areas and the actual areas by dividing the area where both predictions and true values overlap by the total area covered by both.

For example, if a model predicts the boundaries of cells in a microscopic image, IoU would be calculated by taking the area where the predicted cell boundaries overlap with the actual cell boundaries and dividing it by the area covered by both the predicted and actual cell boundaries combined. This score helps determine how closely the model's predictions match the true data. The IoU can be mathematically expressed as follows:

$$\mathrm{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{10}$$

Both the F1 and IoU scores are crucial for evaluating the performance of semantic segmentation models, as they provide complementary information. The F1 score focuses on the balance between precision and recall, helpful in assessing the model's accuracy. In contrast, the IoU score evaluates how well the predicted segmentation aligns with the actual boundaries of objects, which is crucial for understanding the model's ability to delineate objects precisely.

2.8 Loss functions

In semantic segmentation, accurately quantifying the difference between the predicted segmentation maps and the ground truth is crucial for training robust models. Two common loss functions used in segmentation tasks are Dice Loss [44] and Focal Loss [33], each addressing specific challenges in segmentation tasks.

Dice Loss is designed to optimise the F1 score, making it particularly effective for data with imbalanced class distributions. It measures the similarity between the predicted segmentation and the ground truth, focusing on the overlap. The Dice Loss is formulated as:

$$\mathcal{L}_{dice} = 1 - \frac{2 \times |Y \cap \hat{Y}|}{|Y| + |\hat{Y}|} \tag{11}$$

Here, Y represents the ground truth and $\hat{Y}$ the predicted segmentation. By maximising the overlap between Y and$\hat{Y}$, Dice Loss effectively minimises the segmentation error.

Focal Loss, on the other hand, is designed to address the issue of class imbalance by modifying the cross-entropy [9] loss so that it places more focus on hard-to-classify examples. It reduces the relative loss for well-classified examples, directing the model's attention toward complex cases. The Focal Loss is given by:

$$\mathcal{L}_{focal} = -\alpha_t(1 - p_t)^\gamma \log(p_t) \tag{12}$$

Here, p_t is the model's estimated probability for each class being correct, α_t is a weighting factor to balance class importance, and γ is a focusing parameter that adjusts the rate at which easy examples are down-weighted.

2.9 Optimisers

Optimisers serve as the primary tools for the process of weight optimisation during the training of the models. Adam [26], AdamW [35], and Lion [8] are commonly used optimisers for solving segmentation tasks.

Adam is a highly popular optimizer renowned for its effectiveness in managing sparse gradients in noisy scenarios [26]. It merges the benefits of two other stochastic gradient descent extensions: Adaptive Gradient Algorithm (AdaGrad) [12] and Root Mean Square Propagation (RMSProp) [45].

Adam computes an exponential moving average of both the gradient and its squared values. The parameters, beta1 and beta2, control the decay rates of these averages. This feature allows Adam to adjust the learning rate for each model weight individually, based on the estimated first and second moments of the gradients. Consequently, Adam

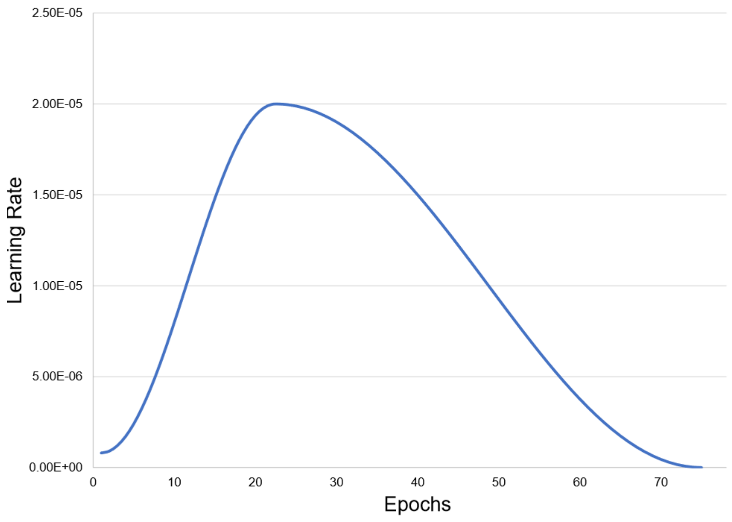

Figure 16. OneCycle learning rate schedule over epochs with peak learning rate of 2e-5 on the twenty-fifth epoch [2].

often surpasses other stochastic optimization techniques in a variety of deep-learning applications [26].

AdamW is an extension of the Adam optimiser, known for its adaptive learning rate capabilities [35]. It addresses an issue where Adam's adaptive learning rates could interfere with regular weight decay, leading to suboptimal regularisation. By reformulating weight decay to directly adjust the weights independently of the adaptive learning rate mechanism of the optimiser, AdamW enhances training stability and performance, particularly in complex models.

Lion, on the other hand, is a newer optimizer that aims to combine the strengths of adaptive learning rate optimizers [8] like Adam with those of momentum-based methods. It introduces a dynamic learning rate adjustment strategy based on the model's training phase, potentially improving convergence speed and model performance.

2.10 Scheduler

The learning rate scheduler is a learning rate management strategy designed to optimise the training process of deep learning models. It operates by varying the learning rate in a pre-defined cyclical pattern that spans a single cycle across the total number of training epochs. This cycle typically starts with a low learning rate, gradually increases to a maximum, and then decreases back to the minimum. This method aims to quickly converge to a reasonable solution initially when the learning rate is increasing and then refine the solution during the decreasing phase, potentially leading to better overall performance and faster convergence. One of the popular learning rate schedulers is

OneCycleLR [43] (see Figure 16).

The essential advantage of the OneCycleLR scheduler lies in its ability to prevent the model from getting stuck in local minima, a common issue in training deep neural networks. By strategically increasing the learning rate, it allows the model to explore a broader range of solutions. Then, decreasing the learning rate ensures the model can settle into a more optimal solution.

3 Method

In the previous section, we described various methods and techniques developed over the past decade for solving segmentation problems ranging from classical algorithms to advanced attention-based networks. Our work focuses on drawing a thorough comparison between U-Net, which is recognised for its effectiveness in medical image segmentation and notable transformer-based models: UNETR, SAM, and Swin Transformer. In this section, we will describe the configuration of those models, the training and evaluation approaches, and the datasets.

3.1 Datasets

For our study, we've assembled a diverse set of datasets, each representing different microscopy image modalities and offering unique segmentation challenges (see Figure 17 for more details). These modalities include electron microscopy, brightfield, histopathology, and phase-contrast. The LIVECell [13] dataset includes 5,239 images, focusing on phase contrast. The Seven Cell Lines [15] dataset consists 3027 brightfield images. The original MoNuSeg [29] [30] dataset includes 30 histopathology images, which we tiled into smaller images of size 512×512. The Electron Microscopy [52] dataset contains 465 images, focusing on electron microscopy type of images. Each dataset in our study is partitioned into predefined training and validation splits. This comprehensive collection allows us to thoroughly evaluate the segmentation capabilities of chosen models across various real-world scenarios, ensuring a broad and in-depth assessment.

Table 1. Overview of datasets used in the study

Attribute	**LIVECell**	**Seven Cell Lines**	**MoNuSeg**	**Electron Microscopy**
Modality	Phase Contrast	Brightfield	Histopathology	Electron Microscopy
Target Object	Individual cells	Nuclei of cells	Nuclei of tissue cells	Various objects
Channels	1	1	3	3
Images	5239	3023	140	465
Resolution	768×512	1080×1080	512×512	Varies

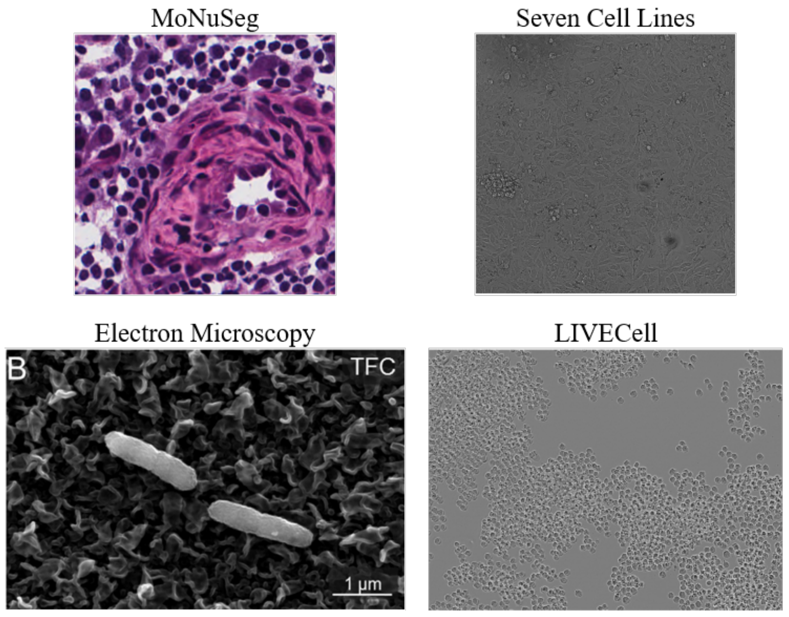

Figure 17. Example image for each of the datasets. From left to right, from top to bottom: histopathology image from MoNuSeg dataset [29] [30], brightfield image from Seven Cell Lines dataset [15], electron microscopy image from Electron Microscopy dataset [52], and phase-contrast image from LIVECell dataset [13].

3.2 Segmentation models

To effectively compare CNNs and transformers in microscopy image segmentation, we selected specific models for our research. U-Net is a well-established CNN architecture known for its robust performance in the microscopy domain [39]. Its architecture, designed specifically for medical image segmentation [39], features a symmetric encoder-decoder structure that effectively captures spatial hierarchies in the image.

We chose UNETR, SAM, and Swin Transformer for the transformer-based models. UNETR is particularly interesting as it incorporates the efficient attention mechanisms of transformers in its encoder, combined with a U-Net-like decoder, making it promising for detailed image analysis [17]. We excluded SETR from our study for two reasons: despite its architectural similarities to UNETR, the latter is specifically designed for biomedical image segmentation, making it more suitable for our needs [17]. Additionally, the Swin Transformer has demonstrated superior performance over SETR in handling varying image scales and resolutions, due to its hierarchical visual features and shifted windows mechanism, essential for processing the multi-scale structures commonly found in microscopy images [34], making it a suitable choice for our study.

Lastly, SAM, trained on huge SA-1B dataset [27], introduces a unique segmentation approach using user-generated prompts, such as bounding boxes or points, potentially

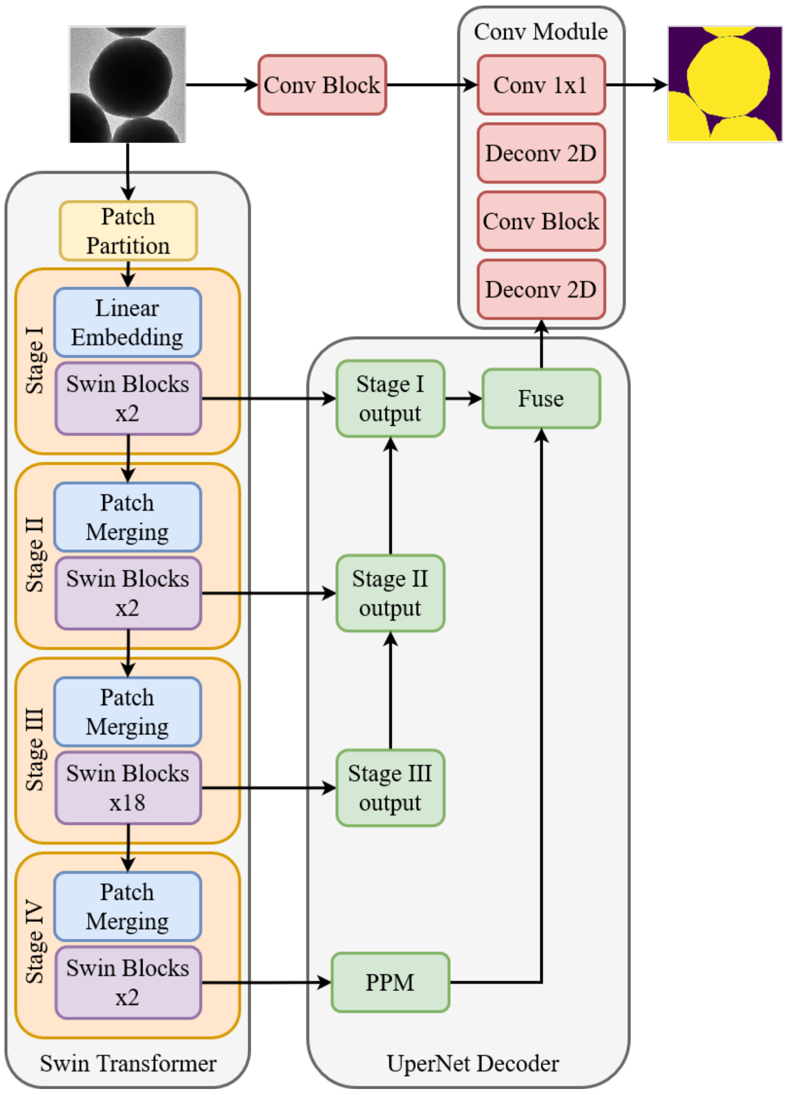

Figure 18. Swin-Conv modification. Here, we changed bilinear interpolation in the decoder part of the network with the series of transposed convolutions and convolutional blocks (Conv Module). The convolutional block consists of a convolutional layer, batch normalisation and ReLU activation function. We also added a skip connection from the input image right to the top layers of the decoder part of the network, passing the input image through the convolutional block and concatenating it with the feature maps from the decoder.

guiding the model towards generating more accurate and precise segmentation masks. This feature makes it a valuable addition to our comparative study, potentially enhancing the model's performance in complex segmentation tasks.

3.2.1 U-Net

In this work, we chose U-net as our baseline model. We utilised the Segmentation Models Pytorch framework (SMP) [21] to build it. ResNet34 [19], pre-trained on the ImageNet [10] dataset, was used as the backbone for the network. We followed the default parameters for the U-Net in the SMP framework and kept the number of stages in the encoder part of the network set to 5. Each stage generates feature maps that are two times smaller in spatial dimensions than the previous one.

3.2.2 UNETR

We have adapted the original UNETR model designed for 3D medical image segmentation to handle 2D images. We followed the original architectural design of the model presented in the "UNETR: Transformers for 3D Medical Image Segmentation" paper [17] with slight adjustments to make it work with 2D images - all of the Conv3D layers in the decoder part of the network were replaced with Conv2D. As for the encoder part, we utilised the base version of ViT (ViT-base) with a patch size of 16×16 pre-trained on the ImageNet [10] dataset.

3.2.3 SAM

We used a pre-trained SAM model, which utilises a ViT-base encoder, trained on the SA-1B [27] dataset. We explored all three ways to segment images using SAM: activating automatic segmentation, prompting it with points and providing bounding boxes as prompts. Bounding box prompts represent the highest degree of interaction with a model compared to point prompts and automatic segmentation modes. The model expects bounding boxes as input in the [B $\times$ 4] format, where B represents the number of output masks. Similarly, the input format for point prompts is [B $\times$ N $\times$ 2], where B is the number of output masks, and N represents the number of points per object. Automatic segmentation, conversely, involves segmenting all possible objects and requires only the input image. The image and corresponding point or box prompts must be supplied to obtain the segmentation mask. To accurately assess the performance of the model, we used the OpenCV [24] framework to generate relevant box and point prompts from binary ground-truth masks. We created corresponding bounding boxes and point prompts for each detected object in the binary mask. These prompts and the input image were then fed into the model to obtain the final result.

3.2.4 Swin Transformer

In this work, we used a small version of Swin Transformer (Swin-small). The architecture consists of Swin Transformer, which serves as the encoder part of the network, and the UperNet decoder head, which processes the features Swin produces and outputs the

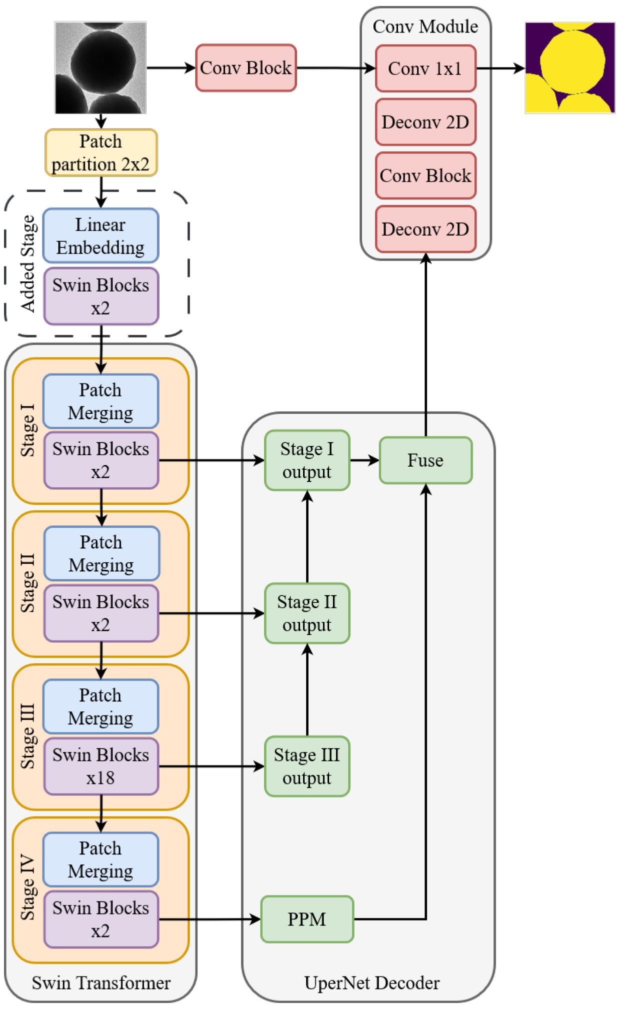

Figure 19. Swin-TB modification. Here we introduce an additional Swin stage. We changed the patch size to 2×2 and kept the idea from the previous experiment — bilinear interpolation is replaced with convolutional operations in the decoder part of the network. We also kept the skip connection from the input image to the convolutional part of the network in the decoder.

segmentation mask. We used a pre-trained model on the ImageNet dataset. The default configuration of Swin utilises a patch size of 4×4 with a window size of 7 (for more details, see Table 3).

While exploring the Swin Transformer architecture, we identified several issues inherent in the default network. Table 2 provides a detailed overview of these issues.

Table 2. Architectural Improvements. In this table, we describe the problems that we found in the architecture of Swin Transformer and provide the way to solve them alongside with explanations.

Limitations of the model	**Our Improvements**	**Explanation**
Patch size	Decreased patches from 4×4 to 2×2	Smaller patches increase the resolution at which features are processed, allowing the model to capture more detailed information crucial for accurate segmentation.
Bilinear Interpolation in decoder	Replaced bilinear interpolation with deconvolutions	Deconvolutions introduce learnability to the upsampling steps, aiming to preserve more detailed information.
Lack of low-level textural information	Added skip connections	Skip connections convolve the input image and concatenate it with the output from the decoder, helping in preserving global context and edge details crucial for high-quality segmentation.
Limited Depth and Feature Extraction	Integrated additional transformer stages	More stages allow the model to learn richer and more abstract representations of the input data, enhancing the model's ability to aggregate and synthesize features across various scales.

3.2.5 Swin Modifications

In addressing the issues identified in the default Swin Transformer configuration, we developed several architectural and configurational improvements. These modifications are aimed at potentially enhancing the performance and adaptability of the model. Below, we detail each variant and the changes implemented:

Swin-PS2 In this variant, we modified the patch size to 2×2, focusing on refining the ability of the model to capture finer details. All other parameters remained unchanged to isolate the effect of this adjustment on the model's performance.

Swin-Conv In this adaptation, we replaced bilinear interpolation after the feature fusion of the decoder with transposed convolutions, followed by sequential convolutional blocks. Each block comprises Conv2D layers with a kernel size of 3, padding of 1, a BatchNorm layer, and ReLU activation. The transposed convolutions are set with a kernel size of 2, zero padding, and a stride of 2. We further enhanced the model by integrating a skip connection, which processes the input image through a convolutional block and then merges these feature maps with those generated by the decoder, as detailed in Figure 18. This change is designed to improve the precision and clarity of the segmentation output.

Swin-TB In this modification, we first expanded the architecture by adding an extra stage at the beginning of the backbone, which includes two Swin Transformer blocks (see Figure 19). This addition aims to increase the ability of the model to process complex features. A patch size of 2×2 introduced even more local context to the model. We also retained the changes made in previous experiments, such as replacing bilinear interpolation with convolutional blocks and keeping skip connection. These adjustments enhance the detail and the overall effectiveness of the segmentation process.

Swin-Pyramid Building on previous concepts, we changed the patch size to 1×1 and extended the model by adding two more stages, altering the embedding dimension to 24, in order to align with the desired embedding dimension of the default stages of Swin and keeping the pre-trained weights of the backbone. The architecture was adjusted so that outputs from the two additional stages are processed by an FPN as well, yielding an output mask with the same width and height as the input image without the need for additional convolutional or interpolation operations (see Figure 20).

3.2.6 Additional modifications of Swin-TB

In this section, we explore the further ways to enhance the Swin-TB modification. We chose to modify this architecture specifically as we thought that it was a promising combination of all our architectural improvements and ideas, described in Table 2.

Our first idea was to remove the skip connection on top of the network and keep the rest of the parameters and layers the same. We wanted to see how skip connection contributes to the ability of the model to produce accurate segmentation masks.

The other idea was to extend the model even more — adding two more stages and changing the patch size to 1×1 to get rich feature representation and capture even more local information from the image. We modified the patch merging operation for these

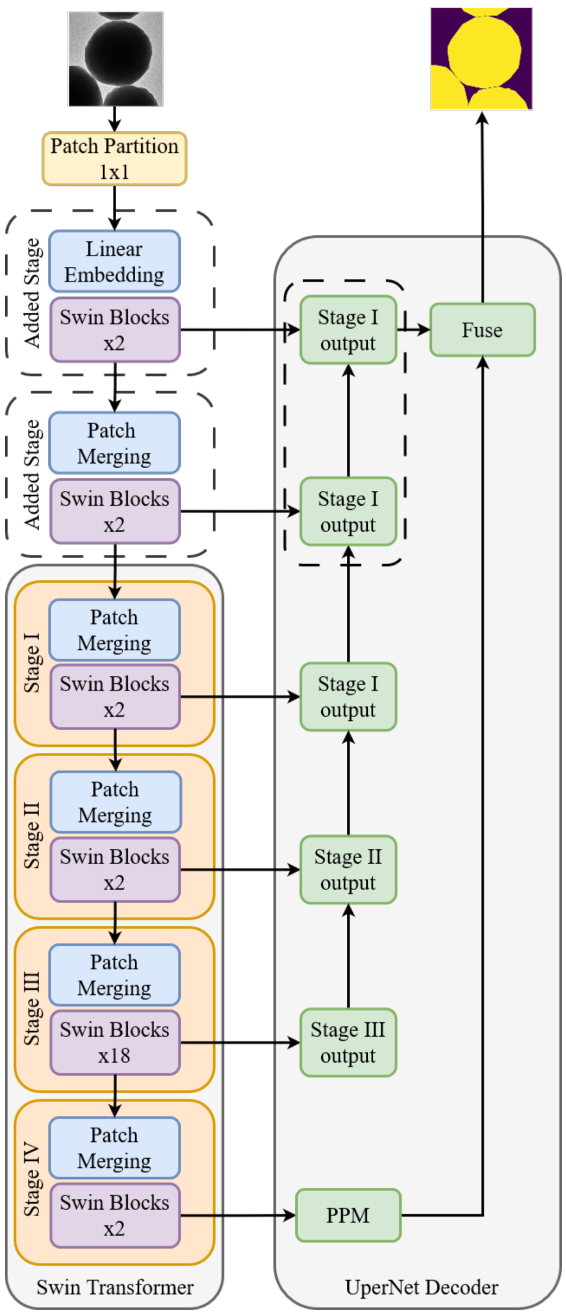

Figure 20. Swin-Pyramid modification. Here, we changed the patch size to 1×1 and added two additional Swin Transformer stages with patch merging operation. The output from these stages is passed right away to the FPN in the decoder part of the network.

two stages to keep the embedding dimension the same across added stages. The reason behind this is to align the output dimensions from these two stages with the desired input dimension to the pre-trained part of the network to keep the pre-trained weights. We followed this idea by conducting another experiment, where we kept the original patch merging operation and changed the initial embedding dimension to 24. In this case, after two patch merging operations, the embedding dimension increases to 96 and perfectly aligns with the rest of the network, keeping the pre-trained weights of the network

Table 3. Detailed overview of parameters of the small version of Swin Transformer model (Swin-small).

Parameter	**Value**
Patch size	4×4
Embedding dimension	96
Window size	7
Depth of transformer	2, 2, 18, 2
Number of heads in each stage	3, 6, 12, 24
Hidden size in MLP layer	768

3.3 Training pipeline

Our training pipeline is designed to efficiently train the deep learning models described above for microscopy image segmentation. This pipeline is implemented in Python, utilising the PyTorch [38] library alongside various utilities to enhance the training process and model performance.

The pipeline employs Hydra [51] for flexible configuration management, allowing easy experimentation with different model architectures and training settings. The Weights and Biases [6] framework was used to track, log, and compare experimental results for this research.

3.3.1 Data Preparation and Augmentation

We initiate our pipeline by setting a deterministic seed to ensure reproducibility. Input data is normalised and transformed using Albumentations [7]. We apply horizontal and vertical flips and a random rotation to the input image. The choice of these augmentations specifically aims to enhance the ability of the model to generalise across different orientations and scales of the input images.

Horizontal and Vertical Flips help the model learn to recognise objects irrespective of their orientation, which is especially important in medical imaging and microscopy, where the orientation of samples can highly vary. Random rotations introduce rotational variance, simulating different angles from which samples might be viewed, further helping the model maintain accuracy regardless of rotational changes.

We also used random cropping of size 224×224. One of the reasons is to avoid unnecessary padding in the Swin Transformer model. If the image dimensions were not a multiple of the product of window sizes and scaling factors through the network's layers, the model would need to apply padding to process the data. This padding can introduce artefacts and affect the model's performance. Another key advantage of using random crops is that it accelerates the training process and decreases its computational costs.

3.3.2 Training Details

Each model, along with its modifications, was trained for 150 epochs, a duration chosen as optimal for convergence. We used a batch size of 16, the maximum that could fit into our GPU memory. Throughout the training process, we sampled 500 random images from the dataset for each epoch, providing diverse examples to enhance model robustness and generalisation. We chose the combination of Dice Loss and Focal Loss to train our models. We compute it as follows:

$$\mathcal{L}_{total} = \alpha \times \mathcal{L}_{dice}(Y, \hat{Y}) + \beta \times \mathcal{L}_{focal}(Y, \hat{Y}) \tag{13}$$

where Y is ground truth, $\hat{Y}$ is the predicted mask, and α and β are the weight coefficients set to 0.9 and 0.1, respectively, as it serves as the standard ratio.

We use a 0.9 coefficient for Dice loss to prioritize segmentation accuracy and manage class imbalances effectively, while a 0.1 coefficient for Focal loss helps focus training on challenging cases, ensuring a balanced approach to model specificity and generalisation.

3.3.3 Evaluation metrics and Testing

We used the F1 score and Intersection over Union (IoU) metrics to assess model performance. These metrics help quantify the accuracy and precision of the segmentation results. The evaluation of the models was done on a separate test set. We evaluated the models using full-size images. As for the U-Net34 and UNETR, we applied a tiling algorithm, partitioning the input image into overlapping tiles, predicting the mask for each tile separately and then merging overlapping tiles with pyramidal weights back into the final predicted mask.

3.3.4 Computational resources

All of the models were trained on the High-Performance Computer Cluster of the University of Tartu, which contains Nvidia Tesla V100 GPU with 32 gigabytes of VRAM and Nvidia Tesla A100 GPU with 40 and 80 gigabytes of VRAM [47].

3.4 Writing aid

We employed ChatGPT [36] and Grammarly [22] for assistance with text structuring and generating LaTeX code for complex tables and figures, enhancing our document's efficiency and presentation.

4 Experimental Results

In this section, we will detail each experiment conducted and present the results for each model, including their modifications, across various datasets. These experiments aimed to thoroughly compare the efficacy of conventional U-Net and transformer-based segmentation models across diverse microscopy image datasets, aiming to understand their strengths and limitations in real-world applications. We will offer a comprehensive comparison of U-Net34 against the transformer models. Additionally, we will analyse our custom modifications to the Swin Transformer, comparing them to the baseline Swin-small and U-Net models.

4.1 Comparison of Transformer Models

Table 4. Performance of transformer models compared to U-Net across datasets. Each row represents the model, while each column represents the obtained F1 and IoU values on each of the datasets. The best scores are highlighted in **bold**, and the second best scores are underlined.

Model	**LIVECell**		**Seven Cell Lines**		**MoNuSeg**		**Electron Microscopy**	
	F1	**IoU**	**F1**	**IoU**	**F1**	**IoU**	**F1**	**IoU**
U-Net	<u>0.92</u>	<u>0.86</u>	**0.81**	**0.70**	0.80	0.68	**0.92**	**0.88**
UNETR	**0.93**	**0.87**	<u>0.80</u>	<u>0.68</u>	<u>0.82</u>	0.70	0.83	0.75
Swin-small	0.90	0.84	0.79	0.66	<u>0.82</u>	<u>0.71</u>	<u>0.91</u>	<u>0.85</u>
SAM (Bounding Box)	0.86	0.76	0.78	0.64	**0.88**	**0.79**	0.87	0.80
SAM (Point Prompts)	0.57	0.46	0.27	0.16	0.71	0.57	0.61	0.52
SAM (Automatic Mode)	0.46	0.35	0.17	0.10	0.66	0.50	0.77	0.67

We fine-tuned U-Net, UNETR, and Swin-Basic (referred to as Swin-small) on each dataset separately, following the training protocol outlined in Section 3.3. In contrast, we assessed SAM's out-of-the-box performance without any fine-tuning to evaluate its immediate usability. The results, detailed in Table 4, show that U-Net consistently performs well across all datasets, achieving the highest IoU of 0.88 on the Electron Microscopy dataset. UNETR generally matches U-Net34's performance but lags slightly on the Electron Microscopy dataset. SAM, when using bounding box prompts, demonstrates reasonable performance but does not reach the levels of the fine-tuned models. Performance significantly drops when using point prompts and in automatic mode, particularly on the Seven Cell Lines dataset.

4.2 Comaprison of Swin Modifications

Table 5. Detailed Comparison of Swin Modifications against U-Net Across Datasets. Each row represents the Swin modification. The checkmarks show what has been added to the modification. **Deconv** denotes a series of convolutional blocks and transposed convolutional layers. **Skip** denotes the presence of skip connection from the input image to the decoder part of the network. **Extra Stage** denotes the additional Swin stages in the encoder of the network. **Pyramid** denotes the modification described in paragraph 3.2.5. The best score is highlighted in **bold**, the second-best score is underlined.

Model	**Modifications**					**LIVECell**		**Seven Cell Lines**		**MoNuSeg**		**Electron Microscopy**	
	Patch Size	**Deconv**	**Skip**	**Extra Stage**	**Pyramid**	**F1**	**IoU**	**F1**	**IoU**	**F1**	**IoU**	**F1**	**IoU**
U-Net	—	—	—	—	—	0.92	0.86	0.81	0.70	0.80	0.68	**0.92**	0.88
Swin-Basic	4×4	—	—	—	—	0.90	0.84	0.79	0.66	0.82	0.71	0.91	0.85
Swin-PS2	2×2	—	—	—	—	0.93	0.86	0.80	0.68	0.83	0.71	0.89	0.87
Swin-Conv	4×4	✓	✓	—	—	0.91	0.84	0.77	0.61	0.77	0.61	0.85	0.82
Swin-TB	2×2	✓	✓	✓	—	**0.93**	**0.88**	**0.83**	**0.74**	**0.83**	**0.71**	0.91	**0.88**
Swin-Pyramid	1×1	—	✓	✓	✓	0.91	0.85	0.80	0.67	0.80	0.67	0.90	0.84

Table 6. Comparison of different parameters and modifications of Swin-TB across all datasets

Stages	**Patch Size**	**Embed Dim**	**Our PM**	**Original PM**	**Skip**	**LIVECell**		**Seven Cell Lines**		**MoNuSeg**		**Electron Microscopy**	
						F1	**IoU**	**F1**	**IoU**	**F1**	**IoU**	**F1**	**IoU**
+1	2×2	96				0.92	0.87	0.81	0.70	0.83	0.71	0.90	0.85
+2	2×2	24	✓		✓	0.91	0.85	0.78	0.65	0.82	0.69	0.91	0.87
+2	4×4	96		✓	✓	0.91	0.85	0.78	0.65	0.83	0.70	0.92	0.88

We fine-tuned all of the designed modifications on each dataset separately, utilising the proposed train pipeline, described in Section 3.3 and draw a comparison between U-Net and Swin-Basic models. From the Table 5 we can see that Swin-TB modification excels across all datasets, surpassing U-Net. Notably, U-Net maintains almost the same performance on Electron Microscopy. Also, Swin-PS2 shows solid performance and does not fall behind too much. We also conducted a series of additional experiments on the Swin-TB modification described in Section 3.2.6. As we can see in Table 6, the performance of those improvements slightly falls behind the original Swin-TB modifications and does not yield better results. Thus, we consider Swin-TB to be our best modification.

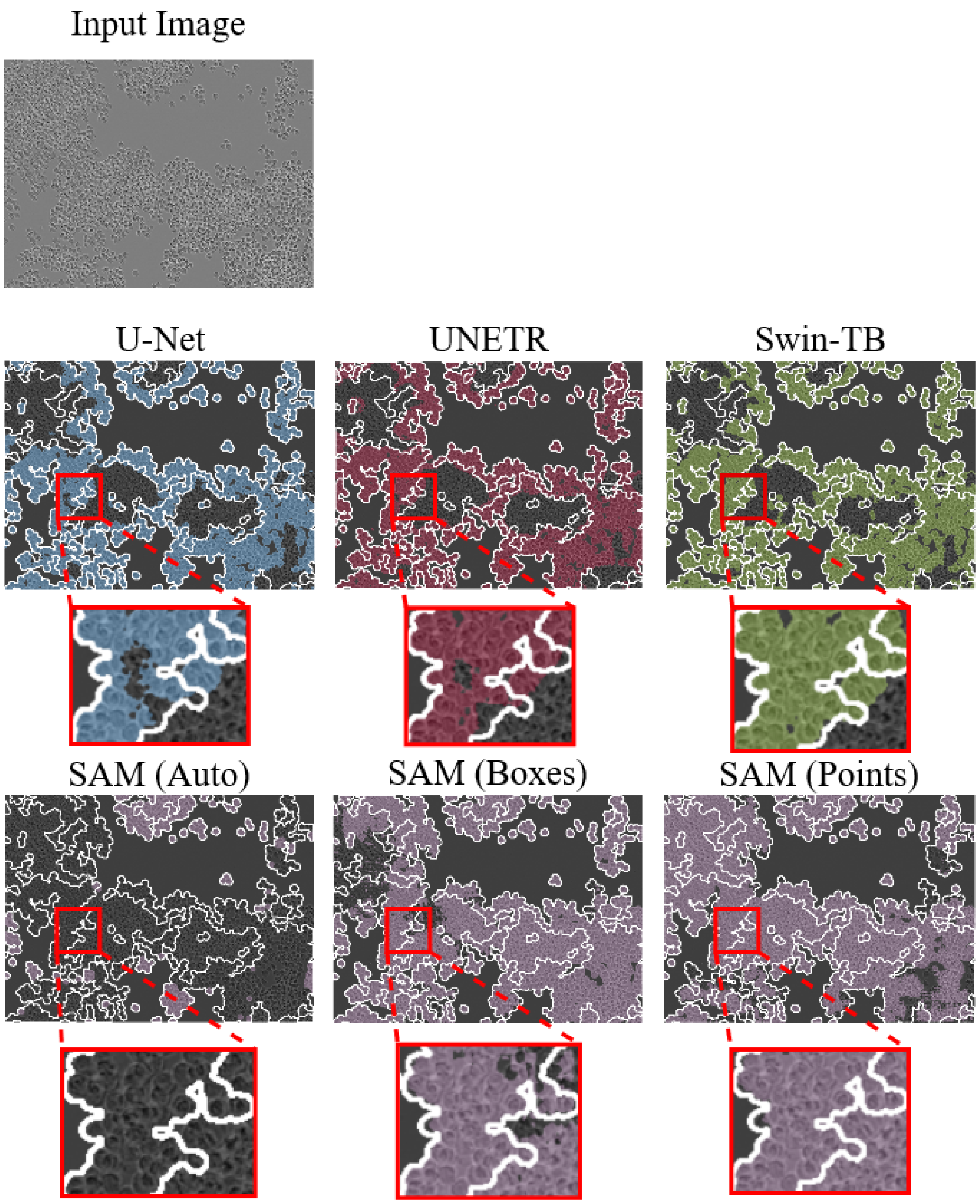

Figure 21. Predicted segmentation masks on the given phase-contrast image from LIVECell dataset. The colour in the image represents the predicted masks. Ground truth mask represented as the white contour.

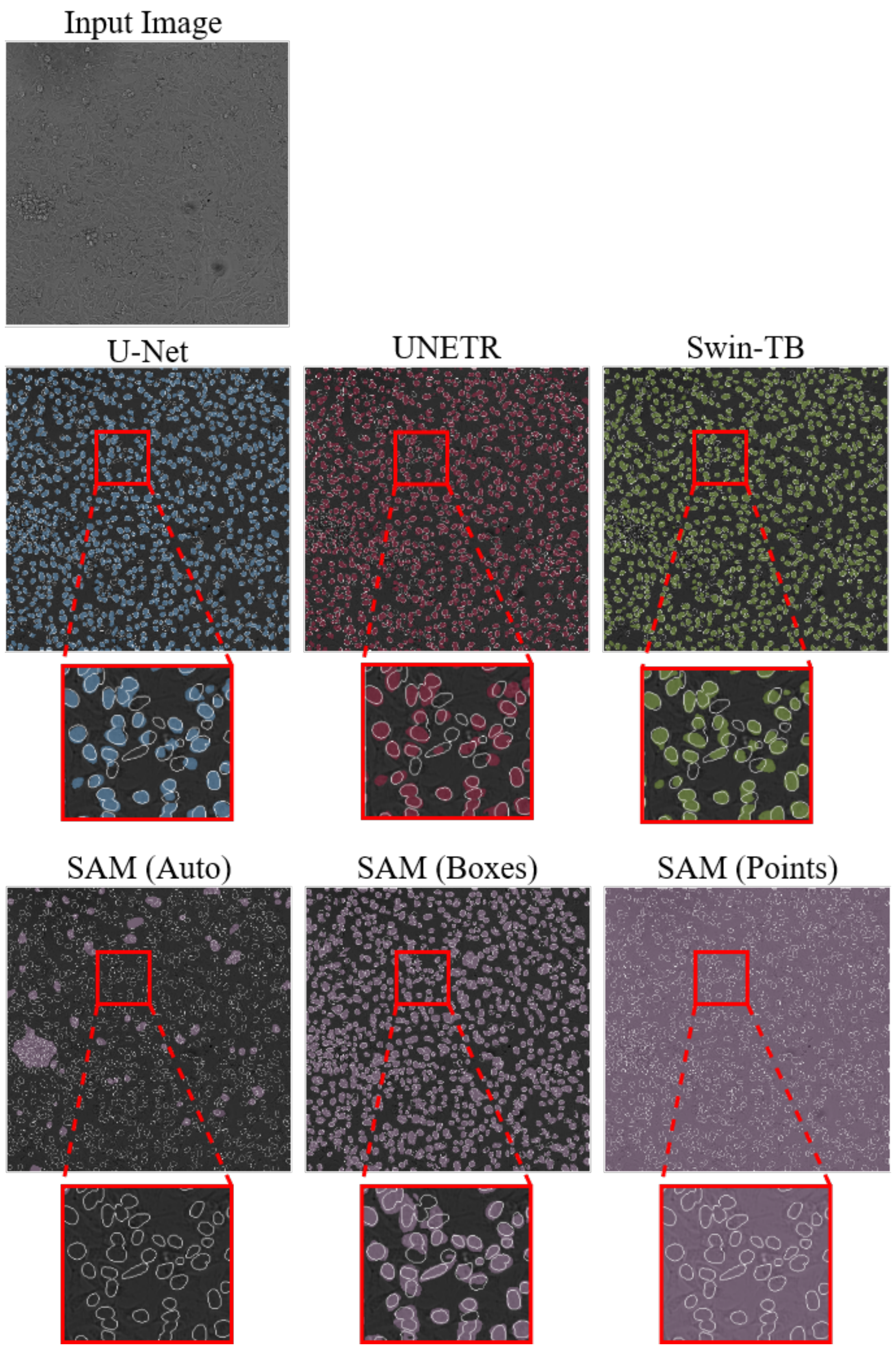

Figure 22. Predicted segmentation masks on the given brightfield image from Seven Cell Lines dataset. The colour in the image represents the predicted masks. Ground truth mask represented as the white contour.

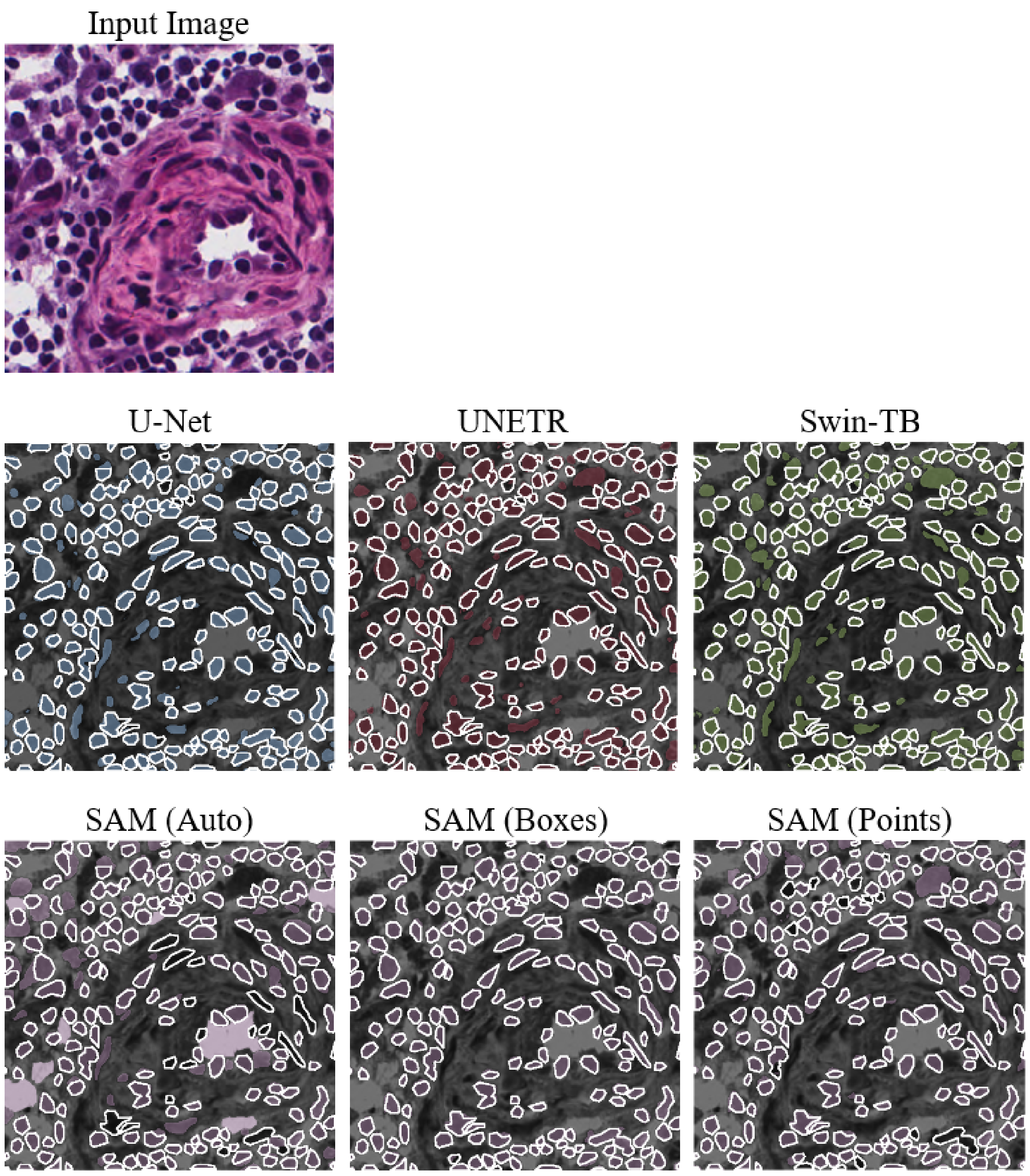

Figure 23. Predicted segmentation masks on the given histopathology image from MoNuSeg dataset. The colour in the image represents the predicted masks. Ground truth mask represented as the white contour. The input image on the background of each prediction is converted to grayscale for the visualisation purposes.

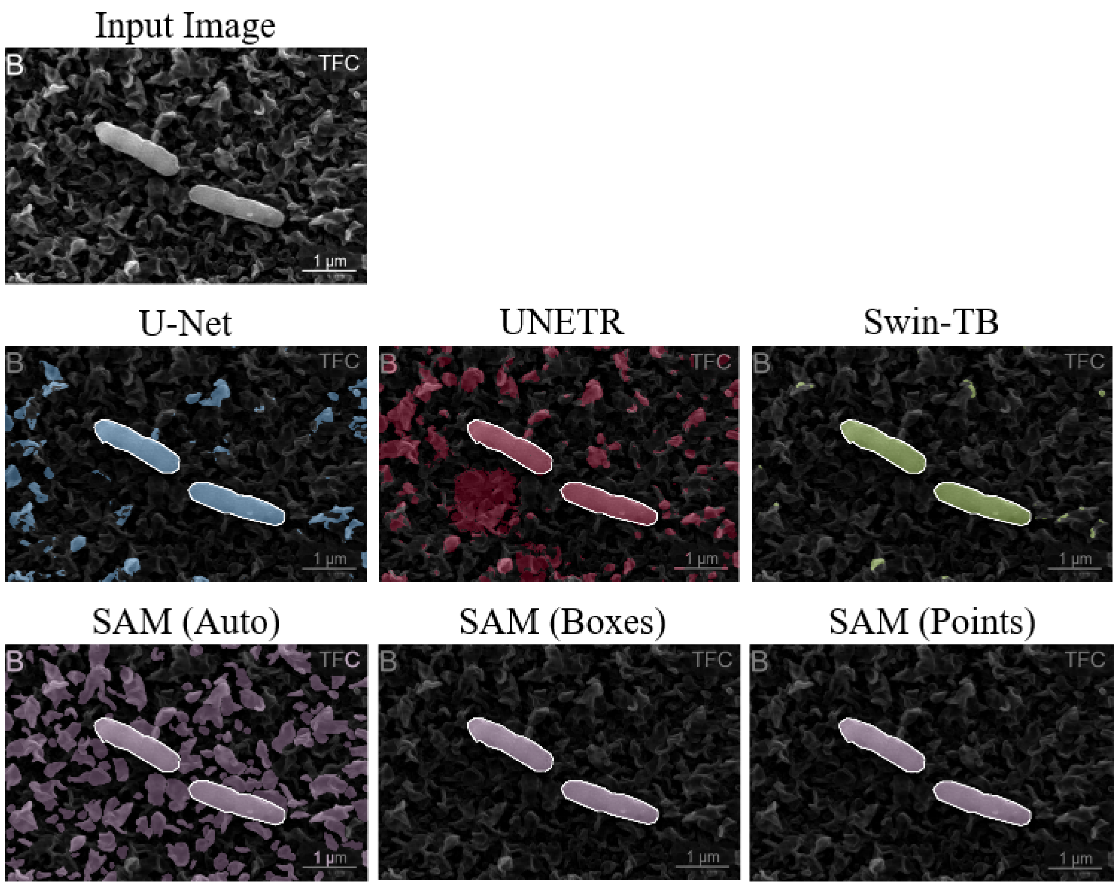

Figure 24. Predicted segmentation masks on the given electron microscopy image from Electron Microscopy dataset. The colour in the image represents the predicted masks. Ground truth mask represented as the white contour.

5 Discussion

Our experiments explored the capabilities of transformer-based models, such as UNETR, SAM, and various versions of the Swin Transformer in recognising and segmenting objects within biomedical images across different modalities.

The results displayed in Table 4 indicate that U-Net remains a robust model and a good choice for semantic segmentation challenges in biomedical images. While UNETR and Swin Transformer (Swin-small) perform on par, they still fall behind on datasets such as Seven Cell Lines and Electron Microscopy. SAM delivered the best results by utilising bounding boxes as the input prompt to the model, yet they achieved worse results than other transformers. It is also worth noting that SAM requires user-generated prompts to operate on images, demanding additional effort from the user. SAM also tended to segment some unnecessary areas incorrectly, showing it lacked class awareness, especially when automatic segmentation was activated (see Figures 24, 22, 23).

While the default version of Swin Transformer did not show promising results compared to U-Net, our designed architectural modification, utilising convolutional blocks, skip connection, and extended encoder, yielded impressive improvements compared to the original architecture. Table 5 and Figures 22, 21 demonstrate that Swin-TB modification achieved higher IoU and F1 scores across all datasets by better handling complex spatial relationships. While the increase in performance is not drastic, it is still noteworthy and shows that there is still room for improvement even in well-established models like Swin Transformer.

5.1 Limitations

One primary limitation is the dependency on the quality and diversity of the datasets used. Despite efforts to include a variety of image modalities, the results of our findings may be limited to the types and characteristics of the datasets employed. For instance, we did not include datasets with fluorescent microscopy images, which could provide different insights and challenges compared to the modalities we used.

Another limitation arises from the computational resources required for training and evaluating complex models like UNETR and Swin Transformer (see Table 7 for number of parameters and FLOPs for each model). Such requirements may not be feasible in all research environments, potentially limiting the broader adoption of these advanced models.

Lastly, the scope of the thesis was restricted to 2D image segmentation. While many microscopy imaging tasks are effectively handled in two dimensions, there are numerous applications, particularly in volumetric imaging, where 3D models might be necessary. For instance, studying 3D models of cysts and tumours requires capturing and analysing volumetric data to fully understand their structure and progression. Therefore, the results and conclusions drawn from this study may not directly translate to 3D applications, necessitating further research and validation in those areas.

6 Conclusion

This work provides a thorough comparison between U-Net and modern transformers - SAM, UNETR, and Swin Transformer in the framework of microscopy image analysis. Our results illustrate that while modern transformer-based models perform comparably to the robust U-Net, they can be improved. Particularly, in this work we proposed and implemented a number of architectural modifications to the Swin Transformer. Experiments that we have conducted have shown that these modifications notably enhance its performance. To this end, our version of the Swin model, called Swin-TB, has surpassed the U-Net and other alternative approaches across all tested microscopy datasets with respect to the IoU metric. This improvement has come at the cost of a higher computational burden, which might render the application of our proposed model problematic. Therefore, future research should focus on optimising the proposed transformer model for real-world settings, improving its computational efficiency, and integrating it into various microscopy image analysis workflows.

7 Acknowledgements

I want to express my sincere thanks to my supervisors, Dmytro Fishman and Pavel Chizhov for their tremendous help, guidance, and support. I am also very grateful to the Biomedical Computer Vision Lab and its members for their continuous support. Thank you all for your encouragement and assistance.

References

[1] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, June 1994.

[2] Ayman Al-Kababji, Faycal Bensaali, and Sarada Prasad Dakua. Scheduling techniques for liver segmentation: Reducelronplateau vs onecyclelr. 2022.

[3] Mohammed A. S. Ali, Oleg Misko, Sten-Oliver Salumaa, Mikhail Papkov, Kaupo Palo, Dmytro Fishman, and Leopold Parts. Evaluating very deep convolutional neural networks for nucleus segmentation from brightfield cell microscopy images. *SLAS DISCOVERY: Advancing the Science of Drug Discovery*, 26(9):1125–1137, 2021. PMID: 34167359.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2014.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[6] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[7] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[8] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. 2023.

[9] Pieter-Tjerk de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method, February 2005.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. 2020.

[12] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[13] Christoffer Edlund, Timothy R. Jackson, Nabeel Khalid, Nicola Bevan, Timothy Dale, Andreas Dengel, Sheraz Ahmed, Johan Trygg, and Rickard Sjögren. Livecell—a large-scale dataset for label-free live cell segmentation, August 2021.

[14] Dmytro Fishman, Sten-Oliver Salumaa, Daniel Majoral, Tõnis Laasfeld, Samantha Peel, Jan Wildenhain, Alexander Schreiner, Kaupo Palo, and Leopold Parts. Practical segmentation of nuclei in brightfield cell images with neural networks trained on fluorescently labelled samples, June 2021.

[15] Dmytro Fishman, Sten-Oliver Salumaa, Daniel Majoral, Tõnis Laasfeld, Samantha Peel, Jan Wildenhain, Alexander Schreiner, Kaupo Palo, and Leopold Parts. Practical segmentation of nuclei in brightfield cell images with neural networks trained on fluorescently labelled samples, June 2021.

[16] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. 2017.

[17] Ali Hatamizadeh, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett Landman, Holger Roth, and Daguang Xu. Unetr: Transformers for 3d medical image segmentation. 2021.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv*, 2014.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015.

[20] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. 2017.

[21] Pavel Iakubovskii. Segmentation models pytorch, 2019.

[22] Grammarly Inc. Grammarly. https://www.grammarly.com/, 2024.

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. 2015.

[24] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015.

[25] Xin Jin and Jiawei Han. *K-Means Clustering*, pages 563–564. Springer US, Boston, MA, 2010.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014.

[27] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[29] Neeraj Kumar, Ruchika Verma, Deepak Anand, Yanning Zhou, Omer Fahri Onder, Efstratios Tsougenis, Hao Chen, Pheng-Ann Heng, Jiahui Li, Zhiqiang Hu, Yunzhi Wang, Navid Alemi Koohbanani, Mostafa Jahanifar, Neda Zamani Tajeddin, Ali Gooya, Nasir Rajpoot, Xuhua Ren, Sihang Zhou, Qian Wang, Dinggang Shen, Cheng-Kun Yang, Chi-Hung Weng, Wei-Hsiang Yu, Chao-Yuan Yeh, Shuang Yang, Shuoyu Xu, Pak Hei Yeung, Peng Sun, Amirreza Mahbod, Gerald Schaefer, Isabella Ellinger, Rupert Ecker, Orjan Smedby, Chunliang Wang, Benjamin Chidester, That-Vinh Ton, Minh-Triet Tran, Jian Ma, Minh N. Do, Simon Graham, Quoc Dang Vu, Jin Tae Kwak, Akshaykumar Gunda, Raviteja Chunduri, Corey Hu, Xiaoyang Zhou, Dariush Lotfi, Reza Safdari, Antanas Kascenas, Alison O'Neil, Dennis Eschweiler, Johannes Stegmaier, Yanping Cui, Baocai Yin, Kailin Chen, Xinmei Tian, Philipp Gruening, Erhardt Barth, Elad Arbel, Itay Remer, Amir Ben-Dor, Ekaterina Sirazitdinova, Matthias Kohl, Stefan Braunewell, Yuexiang Li, Xinpeng Xie, Linlin Shen, Jun Ma, Krishanu Das Baksi, Mohammad Azam Khan, Jaegul Choo, Adrián Colomer, Valery Naranjo, Linmin Pei, Khan M. Iftekharuddin, Kaushiki Roy, Debotosh Bhattacharjee, Anibal Pedraza, Maria Gloria Bueno, Sabarinathan Devanathan, Saravanan Radhakrishnan, Praveen Koduganty, Zihan Wu, Guanyu Cai, Xiaojie Liu, Yuqin Wang, and Amit Sethi. A multi-organ nucleus segmentation challenge. *IEEE Transactions on Medical Imaging*, 39(5):1380–1391, 2020.

[30] Neeraj Kumar, Ruchika Verma, Sanuj Sharma, Surabhi Bhargava, Abhishek Vahadane, and Amit Sethi. A dataset and a technique for generalized nuclear segmentation for computational pathology. *IEEE Transactions on Medical Imaging*, 36(7):1550–1560, 2017.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition, 1998.

[32] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. 2016.

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. 2017.

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. 2021.

[35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. 2017.

[36] OpenAI. Chatgpt. Software tool, 2023. Available from OpenAI: `https://www.openai.com/chatgpt`.

[37] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979.

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. 2015.

[40] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain., 1958.

[41] Sebastian Ruder. An overview of gradient descent optimization algorithms. 2016.

[42] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors, October 1986.

[43] Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay. 2018.

[44] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In M. Jorge Cardoso, Tal Arbel, Gustavo Carneiro, Tanveer Syeda-Mahmood, João Manuel R.S. Tavares, Mehdi Moradi, Andrew Bradley,

Hayit Greenspan, João Paulo Papa, Anant Madabhushi, Jacinto C. Nascimento, Jaime S. Cardoso, Vasileios Belagiannis, and Zhi Lu, editors, *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248, Cham, 2017. Springer International Publishing.

[45] T. Tieleman and G. Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude, 2012. COURSERA: Neural Networks for Machine Learning.

[46] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. 2016.

[47] University of Tartu. High performance computing center, institute of computer science, 2018.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[49] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, June 1991.

[50] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. 2018.

[51] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019.

[52] Batuhan Yildirim and Jacqueline M. Cole. Bayesian particle instance segmentation for electron microscopy image quantification, March 2021.

[53] Ying Yu, Chunping Wang, Qiang Fu, Renke Kou, Fuyu Huang, Boxiong Yang, Tingting Yang, and Mingliang Gao. Techniques and challenges of image segmentation: A review, March 2023.

[54] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H. S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. 2020.

Appendix

I. Complexity

Model	Params (M)	FLOPs (G)
Swin-TB	82.1	224.9
UNETR	111.7	83.2
Swin-small	81.1	49.0
U-Net	24.4	6.0
SAM	93.7	—

Table 7. Model Parameters and FLOPs. We calculated the number of FLOPs by passing the 3-channel image of size 224×224 to the model. We cannot provide FLOPs for the SAM model, as it depends on the number of prompts passed to the model, which can vary.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public