Biomedical Image Segmentation of Breast Cancer WSIs – A Study of various Supervised and Unsupervised Methods

Shourya Pratap Singh
[shouryapratap.singh2022@vitstudent.ac.in]

This report outlines the work undertaken during a 7-week internship within BRAF at CDAC, Pune, to study the various possible methods to segment a biomedical image in the absence of any labels (unsupervised segmentation) and with multi class labels (supervised segmentation). The respective datasets and the necessary computational infrastructure required to make this study possible were provided by BRAF, CDAC Pune. Herein, are all the necessary details of the project, the datasets used, the methodologies, and a summary and comparison of the works done, and their possible alternatives.

Table of Contents

1) Introduction

Globally, breast cancer is the second most common cancer affecting women, with histopathology considered the definitive method for diagnosing malignancy. However, manual analysis of these images is labour-intensive, subjective, and susceptible to human error. This has driven the need for effective Computer-Aided Diagnostic (CAD) systems. Computational analysis of histopathological whole-slide images (WSIs) has become essential in enhancing the consistency, accuracy, and scalability of breast cancer diagnosis. A significant obstacle in this process arises from the variations in staining methods and tissue preparation across different laboratories, which create domain shifts between datasets. These inconsistencies often limit the ability of supervised models to generalize effectively across different sources. To overcome this issue, recent research has turned to unsupervised domain adaptation techniques, which aim to transfer knowledge from labelled source domains to unlabelled target domains without additional annotation efforts. We undertake efforts to perform the unsupervised segmentation using clustering, graph based traditional techniques and also graph neural networks. We also explore supervised multi class segmentation techniques to understand about the entire problem in multiple ways.

1.1) Dataset Overview

In this study, we utilize a diverse collection of histopathological whole slide images (WSIs) from multiple sources, encompassing a broad range of resolutions, tissue types, and annotation granularities, to support the development and benchmarking of deep learning models for breast cancer tissue segmentation.

TCGA Whole Slide Images (WSIs):

Our primary dataset comprises ultra high-resolution WSIs obtained from The Cancer Genome Atlas (TCGA), stored in the Aperio .svs file format. These slides exhibit dimensions ranging from 33,864 × 30,341 pixels to 155,539 × 99,584 pixels, corresponding to a physical tissue area of 83 mm × 76 mm up to 709 mm × 249 mm. Captured at submicron spatial resolutions of 0.25 µm per pixel, each .svs file encapsulates a multi-resolution image pyramid, typically with levels at 1x, 4x, 16x, and 64x magnification. This facilitates efficient navigation across image scales during both annotation and model training.

TIGER Grand Challenge Subset:

To support experimentation, we also use a small set of four WSIs from the TIGER Grand Challenge. These slides are of a .tif format, and are comparatively smaller in scale, averaging around 50,000 × 30,000 pixels, and correspond to a tissue area of approximately 24 mm × 16 mm. These images serve as a valuable benchmark for elementary segmentation trials, with the use of lesser computation for the ease of experimentation

BCSS Dataset from Kaggle:

To enable a study of supervised segmentation methods, we use a large-scale patch-based collection sourced from Kaggle. This dataset includes histopathological image tiles of dimensions 512 × 512 pixels and 224 × 224 pixels, derived from WSIs via manual or automated patch extraction. The 512×512 version retains finer morphological detail, while the 224×224 version is optimized for GPU memory efficiency and faster model iteration. The 512 version has a 6000 training images, and the 224 version has 30,760 training images.

All patches are encoded in the sRGB color space without embedded gamma correction or DPI metadata. The absence of spatial calibration precludes inference of real-world physical dimensions for the patches. The dataset includes pixel-wise annotations across 22 histological tissue categories, namely: outside_roi, tumor, stroma, lymphocytic_infiltrate, necrosis_or_debris, glandular_secretions, blood, exclude, metaplasia_NOS, fat, plasma_cells, other_immune_infiltrate, mucoid_material, normal_acinus_or_duct, lymphatics, undetermined, nerve, skin_adnexa, blood_vessel, angioinvasion, dcis, other.

This heterogeneous and hierarchically structured dataset supports a wide range of experimentation in tissue classification, semantic segmentation, multi-scale learning, and model generalization across resolutions and sources.

2) Literature Review

Unsupervised segmentation approaches have recently leveraged deep feature learning and graph formalisms to segment histological images without labels. For example, **UnSeGArmaNet**, is an unsupervised framework that extracts patch features using a pretrained Vision Transformer (ViT) and then builds a graph over patches. A Graph Neural Network with auto-regressive moving average (ARMA) filters is trained with a modularity-based loss to partition the images into regions. Remarkably, UnSeGArmaNet achieves performance close to supervised methods on several standard image segmentation benchmarks (ECSSD, DUTS, CUB) and even on challenging medical sets (KVASIR, CVC-ClinicDB, ISIC-2018. This suggests that unsupervised graph-based deep models can segment tissue structures without annotations.

Classical graph algorithms have a long history in histology: Felzenszwalb and Huttenlocher's graph-based method runs in near-linear time and preserves detail in low-variability regions while ignoring detail in high-variability regions. Such methods cluster pixels by intensity/texture, which can yield superpixel-like regions. However, they lack semantic guidance and require manual tuning of scale parameters. Similarly, shape-guided segmentation integrates top-down shape priors into segmentation by combining Bayesian shape models with bottom-up region cues. While not specific to histology, it illustrates how multi-scale and shape information can guide unsupervised grouping of complex structures (e.g. cell clusters or ducts). Another line of work uses heuristic optimization, proposing a "advanced equilibrium optimizer" for multi-level thresholding. Their AEO algorithm uses population-based search to find optimal thresholds for gray-level segmentation, demonstrating outstanding quality (high PSNR/SSIM) on benchmark images. In principle, such thresholding could roughly separate tissue regions (e.g. stroma vs. tumour), but in practice it may struggle with the rich colour and texture variation in H&E slides and often needs careful parameter tuning.

Recent advances also apply self-supervised or clustering approaches on medical volumes. Moriya et al. (2018) extends **unsupervised deep clustering** to 3D pathology (lung micro-CT). They train a CNN in an unsupervised fashion by alternately clustering its latent features (using Joint Unsupervised Learning, JULE) and updating the network based on those cluster labels. Applied to micro-CT of lung cancer, they could segment each 3D image into tumour (invasive vs. in situ) and normal tissue without manual labels. This two-phase approach (learn features, then k-means cluster) shows the potential of unsupervised CNNs to discover tissue categories directly from the data, though it was demonstrated on micro-CT and not yet on optical histology. Likewise, **CUTS** is an unsupervised deep framework that generates *multi-scale* segmentations. CUTS first embeds each image using a contrastive/self-reconstruction loss, then partitions the embedding at multiple granularities based on topology. On retinal and brain MRI, CUTS yielded coarse-to-fine segmentations and improved Dice scores by ~10% over prior unsupervised methods. Notably, CUTS' performance rivals supervised models like SAM (Segment-Anything) trained on billions of labelled images, suggesting that unsupervised medical segmentation can approach supervised quality. However, CUTS has not yet been applied to breast WSIs, and its diffusion of scales may need adaptation for the heterogeneous tissue regions in histopathology.

Semi-supervised methods in histology address annotation scarcity by combining labelled patches with unlabelled data. Nguyen *et al.* (2025) propose Feature-Diversified Collaborative Learning (FDCL) for histopathology segmentation, which co-trains two networks with a loss that forces their feature representations to differ. This combats the common problem in co-training where two models reinforce each other's mistakes. FDCL achieves state-of-the-art segmentation on the GlaS colon histology dataset using only 10% of the labels, demonstrating that encouraging diverse features can improve semi-supervised performance. Such methods are directly relevant to multi-class WSI segmentation (e.g. tumour vs. stroma vs. fat), where exhaustive annotation is impractical.

Supervised deep learning dominates current whole-slide image (WSI) analysis. Many systems perform patch-level classification or object detection using convolutional neural networks (CNNs). For example, Hameed *et al.* (2022) trained an Xception-based CNN on a multi-class histology dataset (normal, benign, in situ, invasive) and achieved ~98% accuracy ($\kappa \approx 0.969$). They extracted features from intermediate layers of Xception and showed that even simple stain-normalization yielded near-perfect AUC (>0.99) on each class. This indicates that deep CNNs can very accurately distinguish breast histology categories when trained on curated image crops. Similarly, Sneider *et al.* (2023) used a multi-class CNN to segment breast tissue into classes like tumour cells, ducts, fat, various collagen types, etc. They then correlated those segmentation maps with mechanical stiffness measurements of the same tissues. They found that the percentage of straight collagen (identified by the CNN) correlates strongly with tissue stiffness, whereas overall breast density (radiographic) does not. This study highlights that CNN-based segmentation of micro-anatomical features can reveal new biomarkers (e.g. straight vs. wavy collagen) in breast cancer.

In the context of WSI-level detection, Zhang *et al.* (2023) proposed a multi-stage pipeline for tumour localization. They normalize whole-slide H&E images with a CycleGAN, then run a deep polyphase network (DPN) to generate heatmaps, and finally a Swin-Transformer based classifier to detect cancer regions. This "fused" model achieved strong tumour localization performance, but at the cost of high complexity and long processing time. Such approaches show promise for unsupervised detection (using heatmaps as a cue), but the multi-stage design and reliance on synthetic transformations can be cumbersome.

Unsupervised deep segmentation is maturing (UnSeGArmaNet, CUTS) but has yet to be fully validated on histological WSIs, which pose unique colour and scale challenges. Semi-supervised methods like FDCL help reduce annotation needs, but often rely on extensive pretraining and still need some ground truth. Supervised deep learning yields high accuracy on curated patches, yet WSIs bring issues of data heterogeneity and label imbalance (e.g. sparse tumour regions). Classical methods (graph or thresholding) offer speed but lack semantic understanding. Finally, existing work often treats WSI-level biomarkers (proliferation, survival) separately; integrating fine-grained tissue segmentation with these outcomes is still rare.

3) Experiments

3.1) Unsupervised Methods

3.1.1) VaDE

Reference paper: https://arxiv.org/abs/1611.05148
[Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering]

Data Preparation

The VaDE model was evaluated on image datasets, specifically utilizing large collections of medical images using python's DeepZoom Generator offering, for training and testing. The ImageDataset class was implemented to efficiently load and preprocess images from specified directories. Images were resized to 64×64 pixels and normalized to a range of [−1,1] using standard torchvision.transforms for consistent input to the neural network.

Model Configuration

The VaDE model was configured with the following hyperparameters:

- **Input Dimensions:** 3×64×64=12288, representing the flattened RGB image pixels.
- **Latent Dimensions:** 10, determining the dimensionality of the learned latent space.
- **Number of Clusters:** 3-5, indicating the number of distinct clusters the model is expected to identify.
- **Batch Size:** 32, for training iterations.
- **Epochs:** 50, for the training duration.
- **Learning Rate:** 1×10−4, for the Adam optimizer.

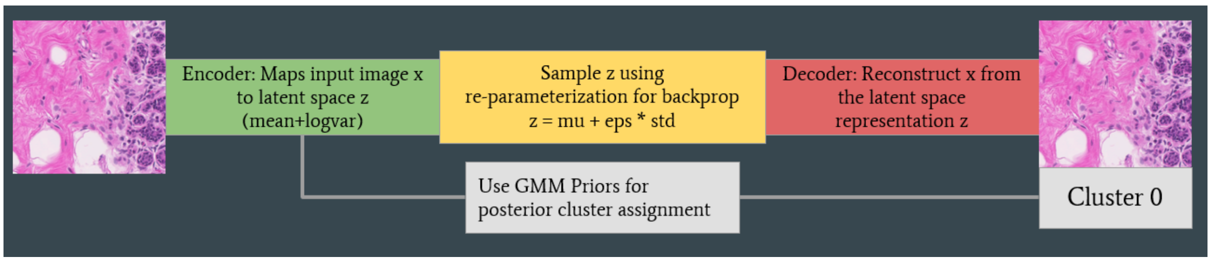
Training and Results

The model architecture consisted of fully connected layers for both the encoder and decoder. The encoder mapped the input to a 500-dimensional hidden layer, then to another 500-dimensional layer, before outputting the mean (μ_z) and log-variance ($\log(\sigma_z)^2$) of the latent distribution. The decoder mirrored this structure, reconstructing the input from the latent space. Crucially, the Gaussian Mixture Model (GMM) parameters (π_k, μ_k, $\log \sigma_k^2$) were initialized based on K-Means clustering applied to a PCA-reduced representation of the training data's raw pixels, providing a robust starting point for the GMM optimization.

Training was performed using the Adam optimizer, minimizing the Negative Evidence Lower Bound (ELBO) as defined by the elbo loss function. This loss comprises a reconstruction term (Mean Squared Error between original and reconstructed images) and a KL divergence term. The KL divergence ensures that the approximate posterior q(z|x) aligns with the GMM prior p(z) and prevents posterior collapse. The GMM parameters were updated along with the encoder and decoder weights during backpropagation, allowing the model to jointly learn meaningful latent representations and cluster assignments. Training progress was monitored by tracking the total loss per epoch.

After training, the VaDE model was used for:

- **Cluster Prediction:** The predict cluster function infers the most probable cluster for a given image by computing the GMM posterior probabilities p(y|z) and selecting the cluster with the highest probability. This was demonstrated for individual images and applied to an entire test folder, with results saved to a CSV file.
- **Cluster Visualization:** To qualitatively assess the learned clusters, the latent space representations (z) of the training data were extracted and reduced to two dimensions using t-SNE. A scatter plot of these 2D embeddings, coloured by their assigned clusters, was generated to visualize the separation and coherence of the learned clusters. This visualization aids in understanding the structure imposed by the GMM prior on the latent space.


Encoder: Maps input image x to latent space z (mean+logvar)
Sample z using re-parameterization for backprop z = mu + eps * std
Decoder: Reconstruct x from the latent space representation z
Use GMM Priors for posterior cluster assignment
Cluster 0

VaDE Model

3.1.2) UnSegARMANet

Reference Paper: https://bmvc2024.org/proceedings/922/

[UnSeGArmaNet: Unsupervised Image Segmentation using Graph Neural Networks with Convolutional ARMA Filters]

Patching and Filtering

The initial step in processing whole slide images (WSIs) for unsupervised segmentation involves decomposing them into smaller, manageable units. This is achieved by systematically extracting dense, overlapping patches of a fixed size (96×96 pixels) with a defined stride (64 pixels) across the entire image. To focus on relevant tissue regions and exclude background noise, a simple yet effective tissue filtering mechanism is applied. Each extracted patch is converted to grayscale, and its mean pixel intensity is compared against a predefined threshold (e.g., 10). Patches with a mean intensity above this threshold are deemed "tissue patches" and retained for further analysis, while background or low-information patches are discarded. The coordinates of these tissue patches are also stored to facilitate spatial reconstruction later.

Feature Extraction

To obtain rich, discriminative representations of the tissue patches, a pre-trained DINO-ViT (Vision Transformer) model, the dino-vitb16, is utilized. This self-supervised learning model is chosen for its ability to learn robust visual features without requiring labelled data, making it suitable for unsupervised segmentation tasks. The AutoImageProcessor handles the necessary preprocessing (e.g., resizing, normalization) for the DINO-ViT model. Patches are processed in mini-batches to leverage GPU acceleration. For each patch, the DINO-ViT model extracts a high-dimensional embedding (e.g., 384 dimensions for vitb16) from the last hidden state corresponding to the [CLS] token, effectively capturing semantic information. These embeddings serve as the node features for the subsequent graph construction.

Graph Construction

The next step in this unsupervised segmentation pipeline is the construction of a graph where each node represents a tissue patch, and edges signify relationships between them. The graph is built by considering both semantic similarity and spatial proximity between patches.

- Semantic Similarity: Cosine similarity is computed between the DINO-ViT embeddings of all tissue patch pairs. An edge is established between two patches if their cosine similarity exceeds a predefined similarity threshold (0.8). This ensures that only patches with similar visual content are connected.
- Spatial Proximity: In addition to semantic similarity, a spatial threshold (200 pixels) is applied. An edge is only formed if two patches are also within this specified Euclidean distance from each other in the original image space. This incorporates the crucial spatial context often lost in purely feature-based clustering. The resulting connections form an adjacency matrix, which is then converted into an edge index tensor suitable for PyTorch Geometric objects, along with the DINO-ViT embeddings as node features (x). This Data object encapsulates the constructed graph.

Graph Neural Network

A custom GNN model, inspired by the principles of Graph Convolutional Networks (GCNs) and using ARMAConv for its performance in spectral graph convolutions, is designed for unsupervised clustering on the constructed graph.

- **Architecture:** The GNN comprises two main components:
    - **ARMAConv Layer:** A ARMAConv layer processes the node features and graph structure. As per the reference, a multi-stacked, multi-layered ARMAConv setup can be employed (e.g., two stacks, four layers), aggregating information from neighbouring nodes. ReLU activation is applied after this layer.
    - **Fully Connected Network (FCN):** Two linear layers follow the ARMAConv layer. The first FCN maps the aggregated features (e.g., 384 dimensions) to a reduced dimension (e.g., 64), and the second FCN projects these features to the number of desired output clusters (k, e.g., 2 for binary segmentation like cancer/non-cancer).
- **Output:** The final output of the GNN is a soft cluster assignment matrix C, where each row represents a node (patch) and each column corresponds to a cluster,

with values indicating the probability of a node belonging to a particular cluster. This is achieved by applying a softmax activation on the last FCN layer's output.

Modularity Loss

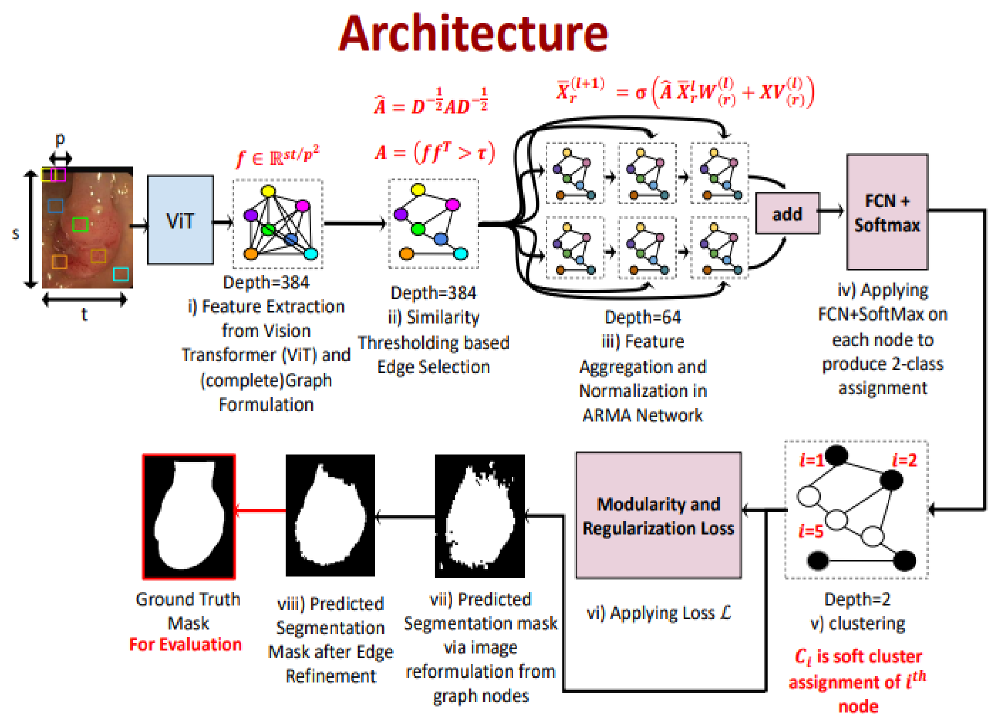
Unlike supervised segmentation that relies on pixel-wise ground truth, this unsupervised approach utilizes a modularity-based loss function to guide the GNN training. The Modularity Loss class implements a variant of the modularity maximization objective, directly optimizing for a good clustering structure in the graph. The loss function, as described in the paper's Equation 8, aims to:

- Maximize the number of edges within clusters and minimize edges between clusters (first term, based on the Modularity Matrix B).
- Encourage balanced cluster sizes (second term, penalizing highly imbalanced cluster assignments). The dense adjacency matrix of the graph, along with the soft cluster assignments C from the GNN, are used to compute this loss.

Training and Visualisation

The GNN model is trained by minimizing the Modularity Loss using the Adam optimizer. During training, the GNN learns to adjust its parameters such that the generated cluster assignments C leads to higher modularity in the graph. After training for a set number of epochs (100), the model outputs the final soft cluster assignments. Hard cluster labels are obtained by taking the argmax of these soft assignments.

- **Pseudo-mask Generation:** The predicted cluster labels for each tissue patch are then mapped back to their original spatial coordinates on the image canvas, creating a pseudo-mask that visually represents the segmented regions (e.g., cancer vs. non-cancer). This pseudo-mask is saved as a grayscale image.
- **Quantitative Evaluation:** For evaluation, if a ground truth mask is available, the generated pseudo-mask is quantitatively compared against it. This involves:
    - Resizing the true mask to match the image dimensions.
    - Binarizing both the predicted and true masks.
    - Calculating common segmentation metrics such as Dice Similarity Coefficient and Jaccard Index (IoU). Since the clustering is unsupervised, the assignment of cluster labels (e.g., which cluster corresponds to "cancer") is ambiguous; therefore, metrics are calculated for both possible assignments, and the better-performing assignment is chosen as the final result.
- **Qualitative Visualization:** Visual comparisons between the original image, true mask, and the predicted pseudo-mask are performed to provide qualitative insights into the segmentation performance. Overlays of the cluster map on the original image are also generated to intuitively show the identified regions.


Architecture
$\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
$\bar{X}_r^{(l+1)} = \sigma\left(\hat{A}\,\bar{X}_r^l W_{(r)}^{(l)} + X V_{(r)}^{(l)}\right)$
$f \in \mathbb{R}^{st/p^2}$
$A = (ff^T > \tau)$
p
s
t
ViT
Depth=384
i) Feature Extraction from Vision Transformer (ViT) and (complete)Graph Formulation
Depth=384
ii) Similarity Thresholding based Edge Selection
Depth=64
iii) Feature Aggregation and Normalization in ARMA Network
add
FCN + Softmax
iv) Applying FCN+SoftMax on each node to produce 2-class assignment
i=1
i=2
i=5
Depth=2
v) clustering
C_i is soft cluster assignment of i^{th} node
Modularity and Regularization Loss
vi) Applying Loss $\mathcal{L}$
vii) Predicted Segmentation mask via image reformulation from graph nodes
viii) Predicted Segmentation Mask after Edge Refinement
Ground Truth Mask
For Evaluation

UnSegArmaNet Architecture

3.1.3) Felzenszwalb

Reference paper 1: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/Borenstein06.pdf
[Shape Guided Object Segmentation]
Reference paper 2: https://cs.brown.edu/people/pfelzens/papers/seg-ijcv.pdf
[Efficient Graph-Based Image Segmentation]

Multiscale Image Segmentation

The initial phase of the proposed methodology involves a bottom-up segmentation approach to decompose the input image into a hierarchy of regions. This is achieved by employing the Felzenszwalb algorithm at multiple scales. The Felzenszwalb function from scikit-learn image segmentation is applied to the grayscale input image with a range of scale parameters (e.g., 100,200,...,12800). Each scale parameter generates a distinct segmentation, where larger scales result in coarser segmentations and smaller scales yield finer, more detailed segments. This process effectively creates a set of nested segmentations, forming the foundation of the hierarchical structure.

A directed graph (G) is then constructed to represent this segmentation hierarchy. Nodes in the graph correspond to individual segments at different levels (scales), identified by a unique label. Edges are established between a fine segment (from scale k) and a coarse segment (from scale k+1) if there is a significant overlap between them. The overlap is quantified using the Jaccard index (Intersection over Union), computed efficiently on GPU using cupy for performance. An edge's weight is set to this similarity score. Each node in the graph is also initialized with a saliency attribute, representing an initial estimate of its importance or distinctiveness. This hierarchical graph provides a structural representation of the image, capturing relationships between regions at varying levels of detail.

Hierarchical Prior Probability Computation

Following the construction of the segmentation hierarchy, prior probabilities of a segment belonging to a specific class (0 or 1) are computed within this hierarchical framework. This computation propagates information from coarser levels down to finer levels. The process begins at the coarsest segmentation level (the highest level in the hierarchy). Segments at this level are initially assigned uniform prior probabilities (0.5 for each class). For subsequent finer levels, the prior probability of a segment belonging to a class X_i is influenced by its parent segments at the immediately coarser level (X_j). The conditional probability $P(X_i|X_j, \text{saliency})$ is modelled, which biases towards agreement ($X_i = X_j$) when saliency is low and towards disagreement when saliency is high. The prior computation function iteratively calculates these probabilities for each node in the graph, weighted by the similarity of the segment to its parent. This top-down propagation of prior information provides a contextual understanding of segment likelihoods based on their enclosing larger regions.

Template Matching

To incorporate specific object knowledge or user-defined cues, a top-down mechanism is introduced through template detections. This mechanism allows for the integration of high-level information into the segmentation process. A template (a known mask of a specific tissue type) is matched against regions in the image. The matching of templates is done by calculating a likelihood score for how well a given template aligns with a region within the image. This score is based on the Hamming distance between the binarized template and the image region, normalized against a uniform distribution, providing a measure of match quality. These individual detections, comprising a region, a proposed labelling (from the template), and a match score, are then aggregated into a continuous soft mask via the aggregation of templates. This creates a weighted average of the labels from all overlapping detections. For each pixel, the contributions from multiple detections are summed, weighted by their respective match scores, and then normalized by the total sum of weights. This results in a floating-point top down mask, where values near 0 or 1 indicate strong evidence for one class or another, respectively. This process effectively translates high-level template knowledge into pixel-level evidence.

Final Segmentation

The final segmentation is obtained by combining the bottom-up hierarchical priors with the top-down aggregated evidence using a Bayesian inference framework. For each pixel (i, j) in the image, the posterior probability of it belonging to class 0 or 1 is calculated. This is achieved by multiplying the likelihood of the pixel given the top-down soft mask with its corresponding prior probability derived from the hierarchical graph. The likelihoods are modelled using exponential functions, where $L_0 = \exp(-\lambda * y)$ and $L_1 = \exp(-\lambda * (1-y))$, with y being the value from the soft mask and λ acting as a scaling parameter. This formulation penalizes discrepancies between the pixel's soft mask value and the assumed class. The final classification for each pixel is then determined by comparing its calculated posterior probabilities for class 0 and class 1. The pixel is assigned to the class with the higher posterior probability, resulting in a binary segmentation mask. This step effectively fuses the general structural knowledge from the multi-scale segmentation hierarchy with specific evidence from template matching to produce the final, refined segmentation.

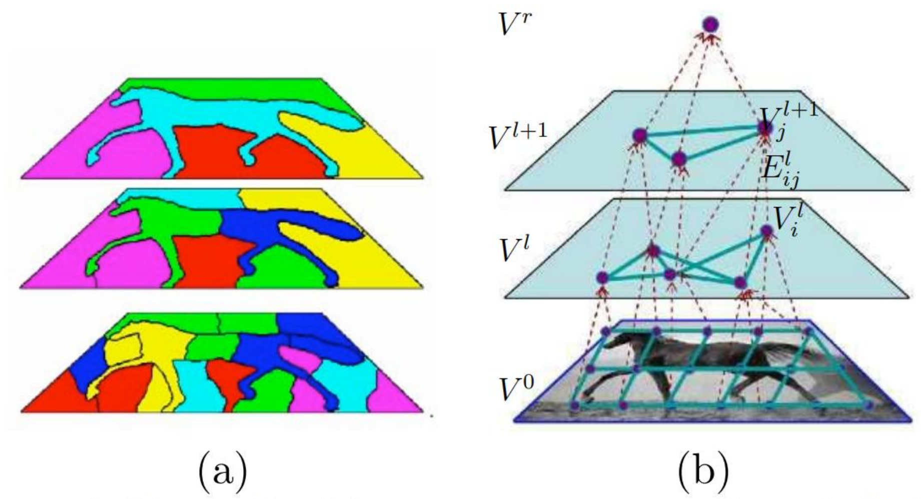


Figure 3. Hierarchical bottom-up segmentation of the image. Segments at multiple scales, identifying salient homogenous regions (a), are represented by a graph $G = (V, E)$ (b). The graph is constructed recursively: Segments V^l at a given level l are used to form larger segments V^{l+1} at the next level, as indicated by their connecting edges E.

A general representation of our method

3.2) Supervised Methods

3.2.1) Transfer Learning based on Full Fine Tuning

Data loading and Augmentation

We use a custom PyTorch Dataset class that handles loading image-mask pairs.

- **Image-Mask Loading:** Images are loaded using PIL and converted to RGB. Masks are loaded as grayscale images.
- **Transformations:** The albumentations library is used for extensive data augmentation, crucial for improving model generalization and robustness, especially with limited medical image datasets. This includes resizing to an encoder-specific input size; geometric augmentations such as Horizontal, Vertical flips etc; colour augmentations changing the hue saturation and other colour characteristics; Normalizing the RGB values to between 0 and 1; and converting the NumPy arrays to PyTorch tensors

Model

The model used is a SegFormer (from the PyTorch segmentation models library).

SegFormer: SegFormer is a recent and powerful semantic segmentation architecture that leverages Transformers. Key features of SegFormer include:

- **Hierarchical Transformer Encoder:** It extracts multi-scale features without requiring positional encodings, making it robust to varying input resolutions.
- **Lightweight MLP Decoder:** It uses a simple Multi-Layer Perceptron (MLP) decoder to aggregate information from different encoder layers, combining local and global attention for powerful representations. This design is known for its efficiency and effectiveness in semantic segmentation.
- **Dropout:** A Dropout layer is added after the base SegFormer model's output. Dropout is a regularization technique that randomly sets a fraction of input units to zero at each update during training. This helps prevent overfitting by forcing the network to learn more robust features and preventing co-adaptation of neurons.

Losses, Optimizer, Scheduler

A Combined Loss is defined, which sums two common loss functions for semantic segmentation.

CrossEntropyLoss: This is a standard classification loss that measures the performance of a classification model whose output is a probability distribution over C classes. For semantic segmentation, it's applied pixel-wise, treating each pixel's classification as an independent task. It is well-suited for multi-class problems and encourages the model to output a high probability for the correct class.

DiceLoss: The Dice Loss is derived from the Dice Coefficient, a measure of overlap between two samples. For segmentation, it quantifies the similarity between the predicted segmentation mask and the ground truth mask. Dice Loss is typically 1−Dice. It is particularly effective for highly imbalanced datasets (common in medical imaging where regions of interest are small) because it focuses on region-based overlap rather than just pixel-level accuracy. The mode=multiclass ensures it is computed for each class and then averaged.

Combination: By combining these two losses, the model benefits from both the pixel-level accuracy driven by Cross-Entropy and the region-level overlap optimization from Dice Loss, leading to more robust and accurate segmentations.

Optimizer: Adam (Adaptive Moment Estimation) is an adaptive learning rate optimization algorithm that computes individual adaptive learning rates for different parameters from the estimates of first and second moments of the gradients. It's used due to its efficiency and good performance in practice. Here, LR=5e-4 and weight decay=1e-4 (L2 regularization to prevent overfitting) are set.

Learning Rate Scheduler: This scheduler dynamically adjusts the learning rate based on a monitored validation metric. The learning rate will be reduced when the monitored quantity (validation loss in our case) has stopped decreasing. The learning rate will be reduced by a set factor (0.5 in our case). The learning rate will be reduced if the validation loss does not improve for a set number of epochs (1 in our case). The scheduler sets a lower bound on the learning rate. This scheduler helps the model converge better by allowing a higher learning rate initially and then reducing it when the training loss plateaus, allowing for finer adjustments in the optimization landscape.

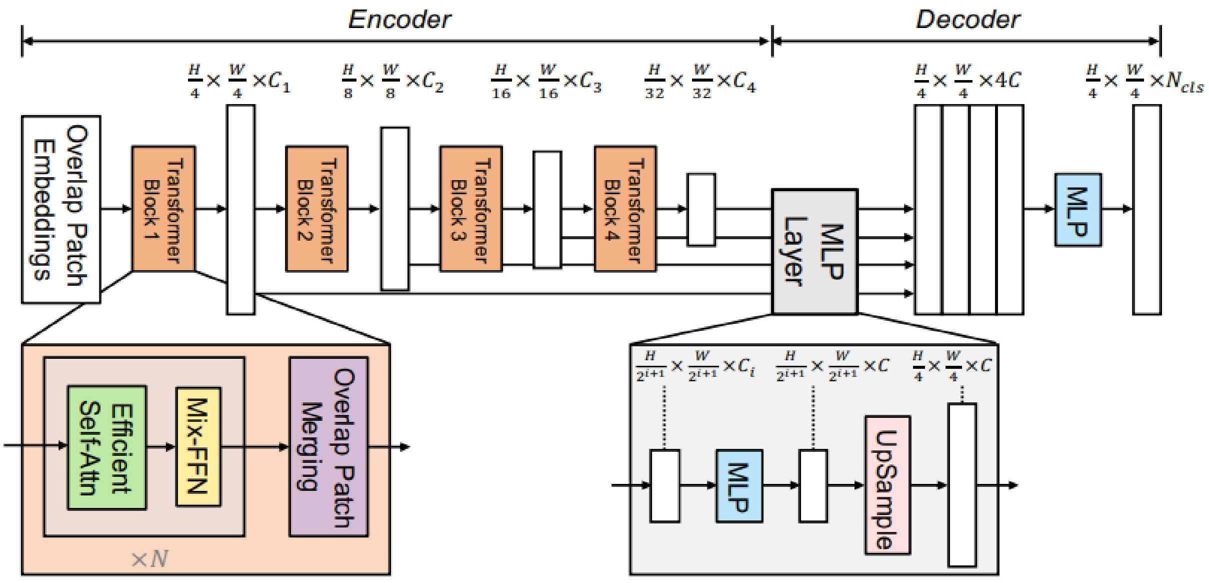
Full Fine-Tuning Approach

Full fine-tuning involves training all parameters of a pre-trained model on a new, target dataset and task. In our work, while the SegFormer model is used, we use pre-trained weights trained on the ImageNet dataset.

The rationale for using a pre-trained model is rooted in transfer learning. Training a deep neural network from scratch on a specialized dataset like BCSS be computationally expensive and may lead to overfitting, especially if the dataset is not as vast as general-purpose image datasets. Pre-trained models have already learned a rich hierarchy of features (edges, textures, shapes, etc.) from diverse natural images. These low-level and even some high-level features are often transferable to new domains, including medical images, even though the specific content differs significantly.

The fine-tuning process involves:

- **Forward Pass:** Input images are fed through the entire SegFormer network.
- **Loss Calculation:** The output logits are compared against the ground truth masks using a Combined Loss function, which sums CrossEntropyLoss (for pixel-wise classification accuracy) and DiceLoss (for region-based overlap, robust to class imbalance).
- **Backpropagation:** The computed loss is backpropagated through all layers of the network. This calculates gradients for every trainable parameter in the model.
- **Parameter Update:** The Adam optimizer uses these gradients to update the weights of all layers.


Encoder
Decoder
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
$\frac{H}{4}\times\frac{W}{4}\times 4C$
$\frac{H}{4}\times\frac{W}{4}\times N_{cls}$
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Efficient Self-Attn
Mix-FFN
Overlap Patch Merging
$\times N$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C_i$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C$
$\frac{H}{4}\times\frac{W}{4}\times C$
MLP
UpSample

SegFormer

3.2.2) Transfer Learning based on Feature Extraction

The data loading and augmentation is similar to what has been previously described in the Full Fine-Tuning Section.

Model

The core of the segmentation model is a U-Net architecture implemented via PyTorch. The U-Net is a well-established convolutional neural network known for its effectiveness in biomedical image segmentation, characterized by its symmetric encoder-decoder structure with skip connections that help preserve spatial information.

The U-Net model is initialized with a Resnet34 encoder. The encoder part of the U-Net leverages weights is pre-trained on the large ImageNet dataset, allowing it to benefit from features learned from general photographic images. To manage computational resources and focus learning on the segmentation task with limited medical data, a feature extraction strategy is adopted:

The parameters of the encoder are explicitly frozen. This ensures that the pre-trained weights of the encoder remain fixed during training, and only the decoder and potentially the final classification layer learn to adapt to the specific segmentation task. This approach is generally preferred when the target dataset is relatively small or has limited annotations, as it prevents the model from overfitting by retaining robust, pre-learned features. The model is configured to handle 3 input channels (RGB images) and output 21 classes, corresponding to the distinct tissue types in the BCSS dataset.

Training and Inference

The model training is configured with the following components:

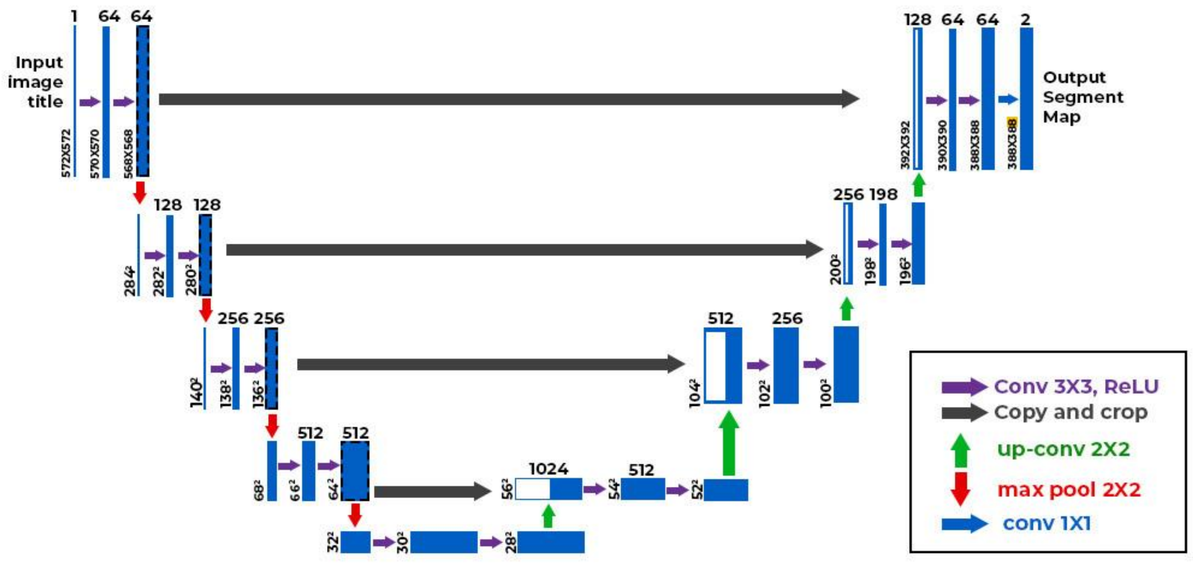
- **Loss:** Cross Entropy Loss is chosen as the loss function. This is a standard and effective loss for multi-class classification tasks, applied pixel-wise across the segmentation mask, encouraging the model to predict correct class probabilities for each pixel.
- **Optimizer:** Adam is used for optimization, with a learning rate of 1e-4 and a weight decay of 0.0001. Adam is a robust and widely used optimizer known for its adaptive learning rates for different parameters.
- **Early Stopping:** Early Stopping is implemented to prevent overfitting and save computational resources. It monitors the validation loss and stops training if the loss does not improve by a delta of 1e-4 for patience of 20 consecutive epochs. The best model weights (based on the lowest validation loss) are saved to a best model .pth file.

The training loop iterates for a maximum of EPOCHS = 100. Inside each epoch:

- The model is first set to training mode. It processes batches of images and masks, calculates the loss, performs backpropagation, and updates the trainable parameters. Training accuracy is calculated using python provided accuracy metric function with reduction="micro" for multiclass segmentation.

- The model is set to evaluation mode for validation. It processes validation batches without gradient computation, calculating validation loss and accuracy.

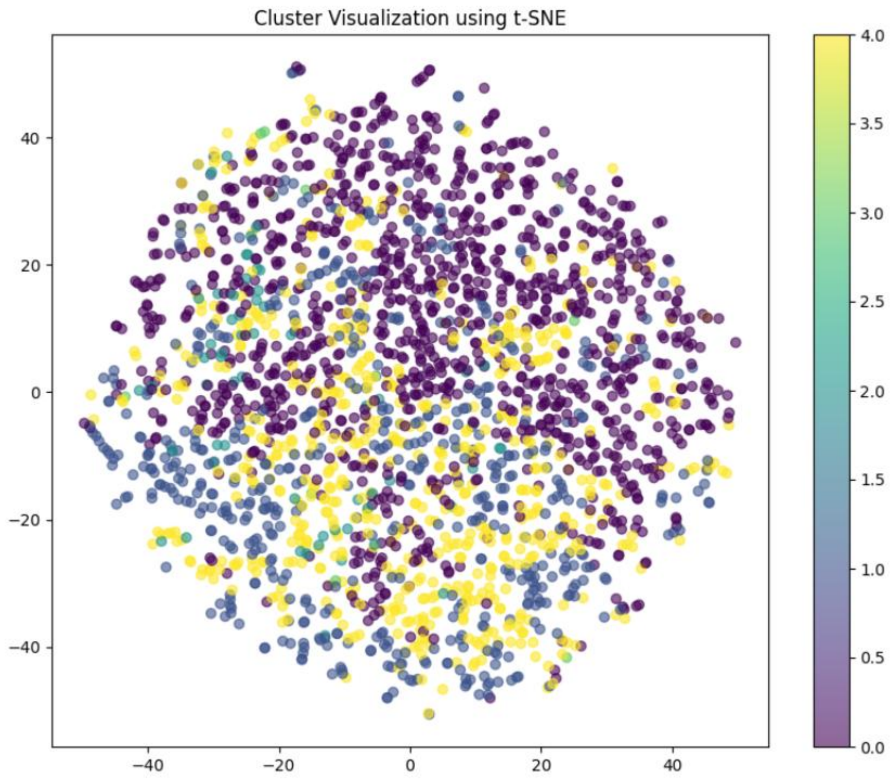
After training, the saved best model checkpoint is loaded for inference. The prediction function is designed to load the best-performing model, set the model to evaluation mode, process the images from a separate test data loader, and for each image, it performs a forward pass, applies softmax to convert logits to probabilities, and then takes the argmax along the channel dimension to get the predicted class for each pixel. The predicted masks are collected and concatenated.



U-Net architecture

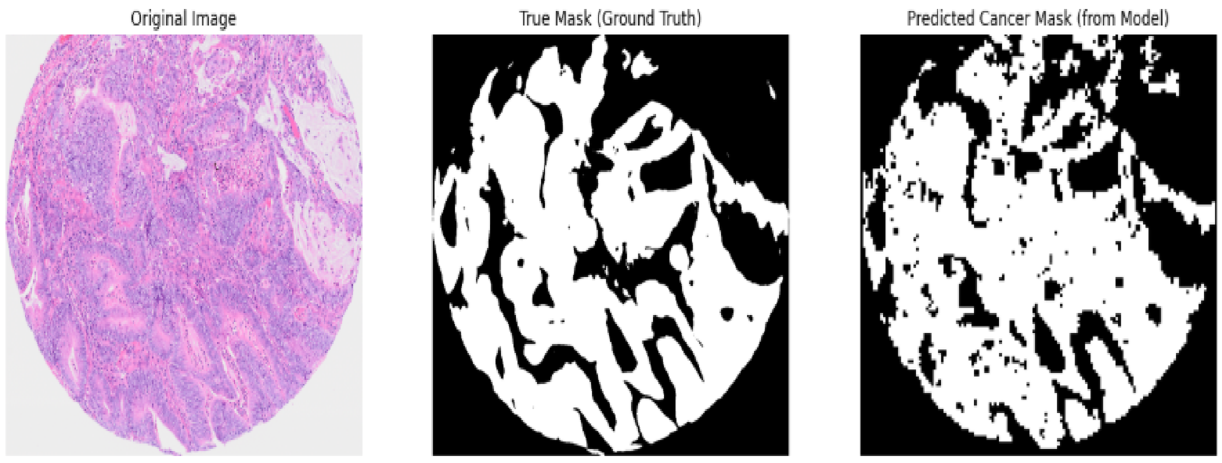
4) Results and Conclusion

Results from VaDE:

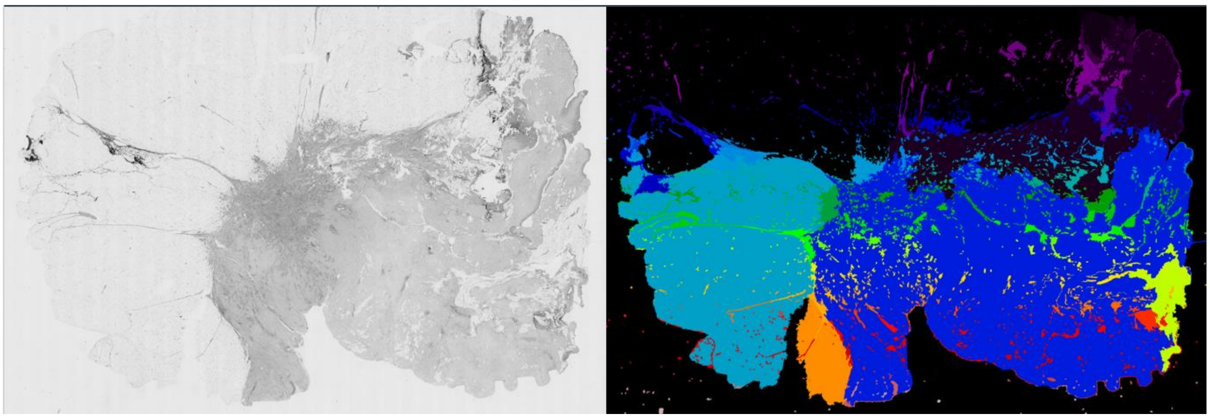


Results from UnSegArmaNet:

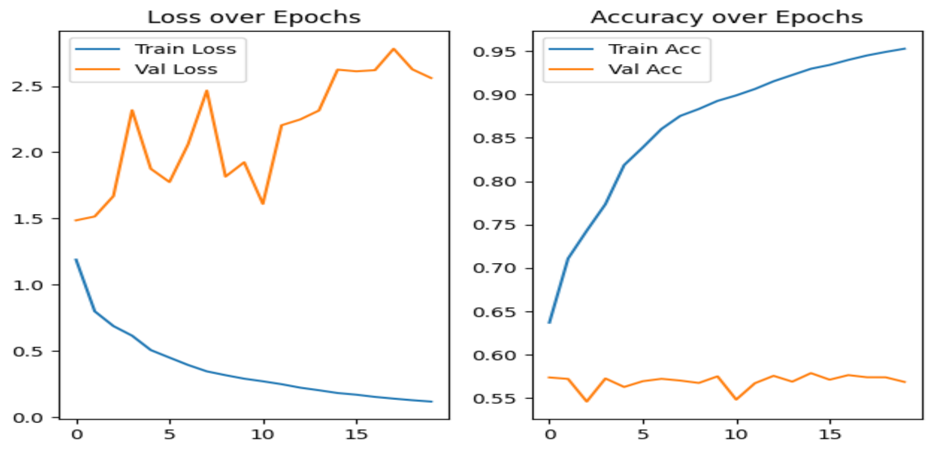
```
--- Evaluation Metrics ---
Assignment: Cluster 1 is assumed to be Cancer (this assignment yields higher Dice/IoU).
Dice Similarity Coefficient: 0.8341
Jaccard Index (IoU): 0.7155
```

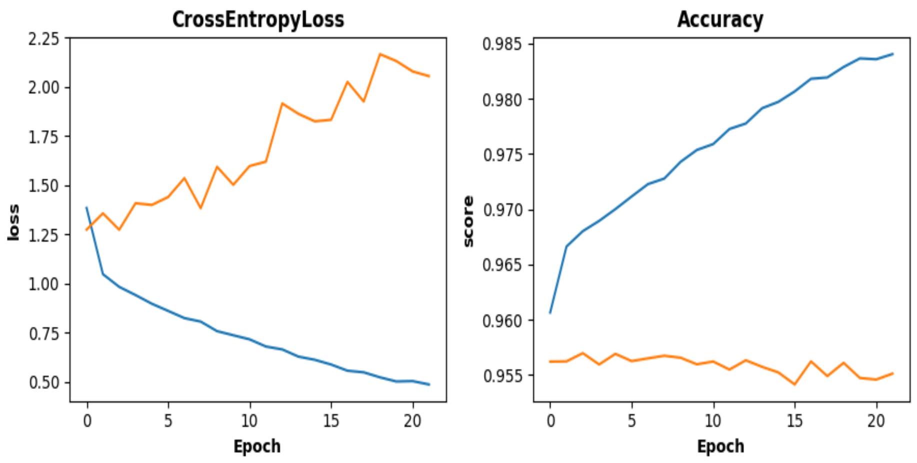


Results from Felzenszwalb:

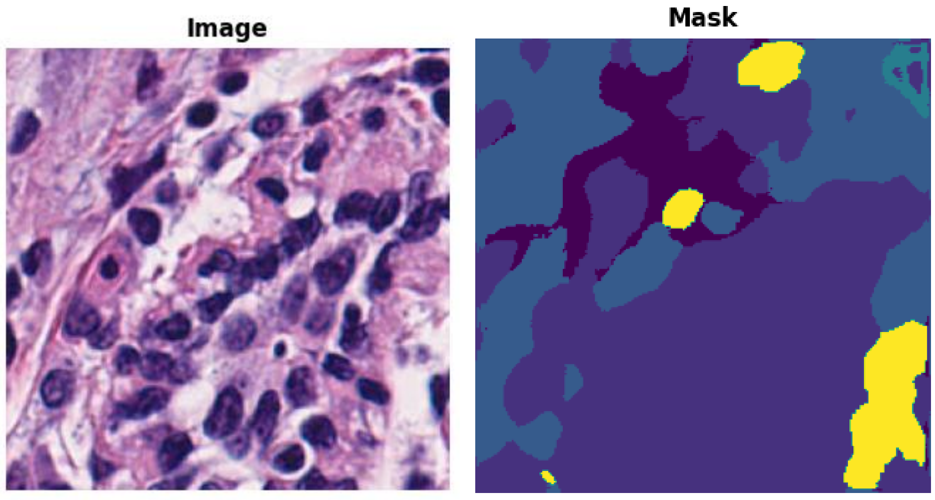


Metrics from Transfer Learning (Full Fine-Tuning and Feature Extraction):





Visualisation of results from Transfer Learning (Full Fine-Tuning and Feature Extraction):



This study investigated supervised and unsupervised methods for breast cancer WSI segmentation. Unsupervised approaches included VaDE for generative clustering, UnSegARMANet leveraging GNNs on DINO-ViT features, and Felzenszwalb for hierarchical segmentation with template matching. These methods demonstrated potential for segmenting unlabelled data by discovering inherent tissue structures. Supervised learning, particularly full fine-tuning of a pre-trained SegFormer model on the multi-class BCSS dataset, achieved high accuracy. Robust training protocols, utilizing combined Cross-Entropy and Dice losses, extensive data augmentation, and adaptive optimization, ensured strong performance. The optimal segmentation strategy depends on data annotation availability and computational constraints. Future research could explore hybrid approaches, combining the strengths of both paradigms.

5) References

1. Aresta, G., et al. BACH: Grand challenge on breast cancer histology images. *Medical Image Analysis, 56*, 122–139. doi.org/10.1016/j.media.2019.05.010

2. Borenstein, E., & Malik, J. (2006). Shape-Guided Object Segmentation.
3. Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision, 59*(2), 167–181.
4. Hameed, Z., Garcia-Zapirain, B., & Aguirre, J. J. (2022). Multiclass classification of breast cancer histopathology images using multilevel features of deep convolutional neural network. *Scientific Reports, 12*, 15600.
5. Habeeb, Z. Q., Vuksanovic, B., & Al-Zaydi, I. Q. (2020). Breast cancer histopathology image classification using deep neural networks: Ensemble of CNN models. *Sensors, 20*(16), 4373.
6. Hu, P., Han, Y., Zhang, Z., Chu, S.-C., & Pan, J.-S. (2022). A multi-level thresholding image segmentation algorithm based on equilibrium optimizer. *Scientific Reports, 12*, 1073.
7. Reddy, K. S. G., Saran, B., Adityaja, A. M., Shigwan, S. J., Kumar, N., & Mukherjee, S. (2024). UnSeGArmaNet: Unsupervised image segmentation using graph neural networks with convolutional ARMA filters. *arXiv preprint* arXiv:2410.06114.
8. Moriya, T., Roth, H. R., Nakamura, S., Oda, H., Nagara, K., Oda, M., & Mori, K. (2018). Unsupervised segmentation of 3D medical images based on clustering and deep representation learning. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018* (pp. 574–582). Springer, Cham.
9. Nguyen, T.-H., Vu, N. L. V., Nguyen, H.-T., Dinh, Q.-V., Li, X., & Xu, M. (2025). Semi-supervised histopathology image segmentation with feature-diversified collaborative learning. [To appear].
10. Paeng, K., et al. (2016). A unified framework for tumor proliferation score prediction in breast histopathology. *arXiv preprint* arXiv:1612.07180.
11. Prasad, K. R., & Udupa, C. B. K. (2023). Breast histopathological image analysis using image processing techniques for diagnostic purposes: A methodological review. [Journal details pending].
12. Sneidera, A., et al. (2023). Deep learning identification of stiffness markers in breast cancer. [Details pending publication].
13. Tembhare, K. (2024). Multi-ensemble machine learning framework for omics data integration: A case study using breast cancer samples. [Details pending publication].
14. The Cancer Genome Atlas Network. (2012). Comprehensive molecular portraits of human breast tumours. *Nature, 490*(7418), 61–70.
15. Veta, M., Huisman, A., Madabhushi, A., Reuz, A. J., Sánchez, O., et al. (2019). Predicting breast tumor proliferation from whole-slide images: The TUPAC16 challenge. *Medical Image Analysis, 54*, 219–232.
16. Zhang, X., Liu, C., Li, T., & Zhou, Y. (2023). The whole slide breast histopathology image detection based on a fused model and heatmaps. [Details pending publication].
17. CUTS: A deep learning and topological framework for multigranular unsupervised medical image segmentation.
18. Kaggle:

18.1) https://www.kaggle.com/datasets/whats2000/breast-cancer-semantic-segmentation-bcss

18.2) *https://www.kaggle.com/code/amspsingh04/bcss-dataset-supervised-seg*

18.3) *https://www.kaggle.com/code/amspsingh04/bcss-seg*