POLITECNICO DI TORINO

MASTER's Degree in COMPUTER ENGINEERING

MASTER's Degree Thesis

3D Organoid Classification via Deep Learning for Toxicological EDC Screening

Supervisors

Prof. Francesco PONZIO

Prof. Xavier DESCOMBES

Prof. Santa DI CATALDO

Candidate

Raffaele MARTONE

MARCH 2026

3D Organoid Classification via Deep Learning for Toxicological EDC Screening

Raffaele Martone

Abstract

This thesis addresses the problem of automatic 3D organoid classification within the field of 3D medical image classification, with application to monitoring the effects of endocrine disrupting chemicals (EDCs). A database of 3D volumes (full z-stacks) of organoids exposed to various EDCs was constructed, annotated according to three main phenotypes defined by morphology and cellular organization: cystic, compact, and cauliflower. The objective of this work was to develop and evaluate an end-to-end framework for the automatic classification of these phenotypes, specifically analyzing: the effectiveness of 3D deep learning architectures in distinguishing EDC-induced phenotypes; the comparison between transformer-based models (SwinUNETR, SwinVit) and traditional 3D CNNs (ResNet, DenseNet)—motivated by the growing importance of transformers in computer vision; and the optimal trade-off between model complexity, computational efficiency, and predictive quality. To this end, A pipeline was proposed, combining state-of-the-art 3D models with an adaptive z-stack volume preprocessing module, based on dissimilarity metrics between slices, aimed at reducing dimensionality while preserving relevant information. Experimental results show that 3D transformers can achieve competitive or superior performance compared to 3D CNNs, especially at higher resolutions, albeit with greater computational costs, and that adaptive slice selection enables memory and inference time reduction while maintaining good classification accuracy. The work also investigates model uncertainty, a crucial metric in the medical domain for assessing prediction reliability. The overall aim of the thesis is to identify the best trade-off between accuracy, computational efficiency, and reliability in phenotype classification, thereby advancing cancer research: organoids, as faithful preclinical models of human tumor tissue, play a fundamental role in pharmacological studies, enabling rapid high-throughput screening of new anticancer drugs and understanding therapeutic resistance mechanisms in more realistic contexts than traditional 2D models.

Table of Contents

List of Figures

List of Tables

Acronyms

BN	Batch Normalization.
CLSM	Confocal Laser Scanning Microscopy.
CNN	Convolutional Neural Network.
EDC	Endocrine-Disrupting Chemicals.
FC	Fully Connected.
FLOPS	Floating Point Operations Per Second.
GNN	Graph Neural Network.
GPU	Graphics Processing Unit.
MIP	Maximum Intensity Projection.
MSE	Mean Squared Error.
ReLU	Rectified Linear Unit.
SGD	Stochastic Gradient Descent.
ViT	Vision Transformer.

Chapter 1

Introduction

1.1 Context and motivation

In this chapter, we will address the scientific context and motivations underlying this thesis, focusing on the importance of organoids as in vitro models for studying the effects of environmental contaminants, particularly endocrine-disrupting chemicals (EDCs). The 3D imaging techniques used to acquire organoid data will be discussed, with a focus on confocal microscopy. Next, biomedical image analysis methods will be examined, highlighting the limitations of manual analysis and the potential of automated artificial intelligence-based solutions. Finally, the specific scientific problem addressed in this thesis will be presented, outlining the objectives and main contributions of the work carried out.

1.1.1 Organoids: a revolution in cell biology and regenerative medicine

Organoids are miniaturized, three-dimensional (3D) versions of organs or tissues, cultivated in vitro from cells with stem cell potential. [1, 2] These structures are capable of self-organization and differentiation, recapitulating the anatomy and physiological functions of their in vivo counterparts.
They are considered revolutionary in cell biology and regenerative medicine, offering more realistic models for studying human development, diseases, and pharmacology compared to traditional 2D or animal models. Unlike conventional two-dimensional cell cultures, where cells grow on artificial flat surfaces losing their original morphology, organoids reproduce the spatial organization, cell-cell interactions, and biochemical gradients typical of real tissues. [2]
Although useful and for decades the basis of biomedical research, animal models present significant limitations due to interspecies differences, which can lead to non-translatable results to humans. [1] Human organoids overcome these barriers, enabling more accurate studies of human physiology and specific pathologies. [2] One of the most promising applications of organoids is in personalized medicine. They can be generated from patient-derived tissues (PDO), allowing testing of drug efficacy on an individual basis and studying genetic variability between individuals. [1] Due

to their miniaturized nature, therefore easily manipulable and observable thanks to advanced imaging techniques like confocal microscopy, organoids represent an ideal platform for studies without the confounding complexity of in vivo experiments. [3] Their modeling capacity opens unprecedented prospects for regenerative medicine, as an unlimited source of compatible tissues for transplants, reducing the risk of immune rejection. [1] Moreover, they can be used as drug development platforms, enabling high-throughput screening of pharmacological compounds in a more physiological context than 2D cultures. [2]

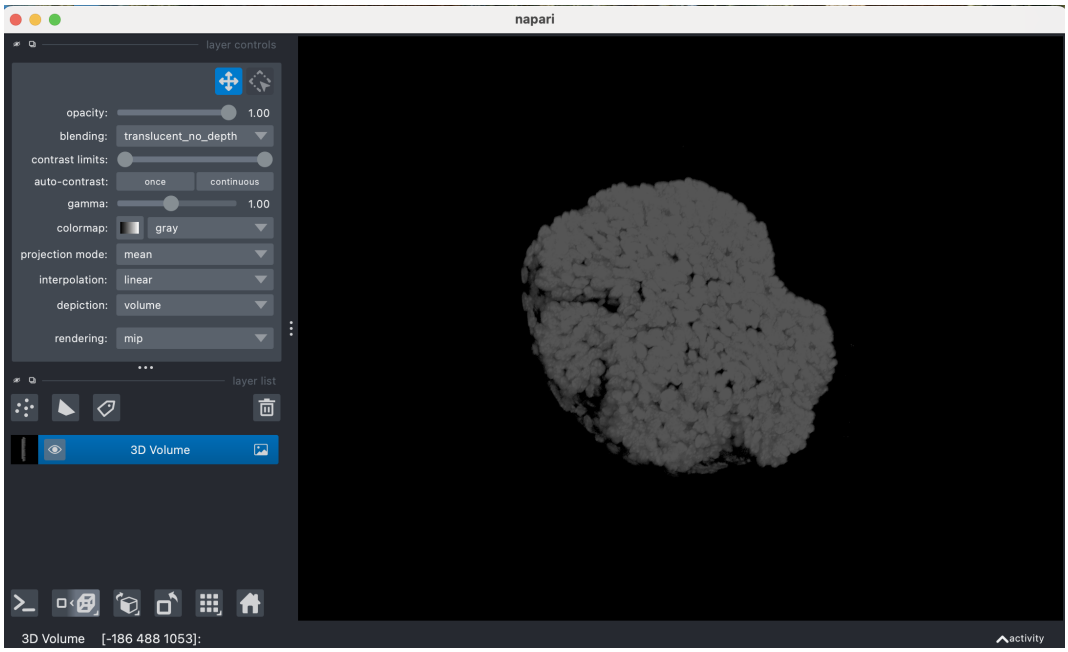

Figure 1.1: Organoid volumetric rendering using Napari.

1.1.2 Endocrine-disrupting chemicals (EDCs) and their impact on health

Endocrine-disrupting chemicals (EDCs) are exogenous substances, external to the organism, that can alter the hormonal pathways of the human body, leading to various harmful health effects. These contaminants interfere with the endocrine system by blocking or mimicking the interaction between natural **hormones** and their receptors. [4]

EDCs are present in numerous everyday products, such as plasticizers for food containers and personal care items (e.g., bisphenol A and phthalates), pesticides (e.g., DDT), heavy metals (e.g., lead and cadmium), and industrial compounds (e.g., PCBs). However, they are not only present in synthetic compounds: some natural substances, such as phytoestrogens found in certain plants, can also act as EDCs. A concerning aspect of EDCs is their ability to accumulate in the environment, given their long half-lives, and in living organisms, leading to chronic exposures even at low doses. Their mechanisms of action are multiple and complex, including modifications in the regulation of hormone availability, interference with signal reception,

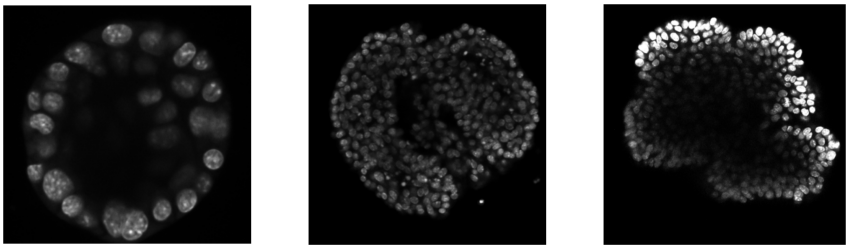

Figure 1.2: Different phenotypes of prostate organoids under EDC exposure.

and epigenetic remodeling. The **alteration** of hormone synthesis, circulation, and metabolism is the primary consequence; for example, Bisphenol A (BPA) is capable of reducing the expression levels of enzymes critical for steroidogenesis, influencing the production of testosterone and other steroids. [4]
The second mechanism is even more sophisticated: EDCs bind directly to hormonal receptors, mimicking natural hormones. This can lead to over-stimulation or blockade of receptor activity, altering normal hormonal signaling. The third level of complexity concerns epigenetics, that is, the mechanisms that control gene expression without altering the DNA itself, silencing or activating specific genes involved in hormonal regulation. These effects translate into a wide range of health problems, including reproductive disorders, metabolic dysfunctions, alterations in neurological development, and increased risk of hormone-dependent tumors.

1.1.3 Impact of EDCs on prostate health and use of organoids as models

In particular, our object of study concerns the impact of EDCs on prostate health, an organ highly sensitive to hormonal variations. Chronic exposure to such substances has been associated with an increased risk of prostate cancer (PCa) and the development of pre-neoplastic lesions. Through the use of organoids, it is possible to observe how EDCs visibly alter tissue morphology based on concentration and type of interference:

- **Cystic Phenotype (Healthy):** Under normal conditions or with adequate DHT levels, organoids present two layers of nuclei, an internal lumen, and the secretion of prostatic fluid that exerts pressure on the walls.
- **Compact Phenotype:** High concentrations of androgen antagonists (such as DDE, a DDT metabolite) disrupt cell differentiation, preventing the formation of the internal cavity and generating dense aggregates of nuclei devoid of fluid.
- **“Cauliflower” Phenotype:** Excessive and uncontrolled cell proliferation produces organoids with irregular and nodular surfaces, a sign of induced tumor growth.

1.2 Biomedical image analysis for organoids

1.2.1 Generation and culture of organoids

Organoids are generated by embedding stem cells in 3D matrices that mimic the extracellular environment. These can derive from embryonic stem cells (ESC), induced pluripotent stem cells (iPSC), adult stem cells (ASC), or patient biopsies (PDO).
The process relies on three key principles:
1. Self-organization: Stem cells spontaneously rearrange into organ-like structures through intrinsic developmental programs, guided by specific growth factors added to the culture medium.
2. 3D matrix support: Unlike 2D cultures, organoids require a three-dimensional scaffold:

- **Matrigel:** Most common (protein mix from mouse tumors)
- **Synthetic hydrogels:** More controlled and reproducible

These matrices provide both structural support and biochemical cues for cell polarization.
3. Culture methods: Organoids at this stage appear as small opaque domes or spheres (0.5-3 mm) embedded in their matrices, ready for confocal microscopy after 7-21 days of culture. In **submerged culture**, organoids are encapsulated within a Matrigel domes fixed to the bottom of culture plates and fully immersed in 1-2 mm of nutrient medium. They appear as translucent domes containing whitish organoid clusters clearly visible against the gel background. The **air-liquid interface (ALI)** method grow on porous membranes with basal side in medium and apical side exposed to air, producing larger (2 mm), stratified structures. In **rotating bioreactors**, organoids float freely in suspension (1-5 mm diameter) with slow rotation (20-40 rpm) preventing sedimentation. Continuous movement ensures uniform nutrient.
At this mature stage (200-1000 μm), organoids exhibit compact, well-defined architecture with stable morphology, making them perfect subjects for 3D confocal imaging. [5]

1.2.2 Confocal microscopy for organoids

Laser scanning confocal microscopy (CLSM) is particularly well suited for acquiring three-dimensional images of organoids, because it can generate high-resolution optical sections without physically cutting the sample. [6]
In CLSM, a laser beam is focused through an objective into a very small illumination spot within the tissue. Fast galvanometric mirrors scan this spot across the field of view along the X and Y axes, so that the image is built point by point. A key element of the system is the **pinhole**, placed in front of the detector, which blocks out-of-focus light and removes the blur typical of conventional widefield microscopy. To capture the full depth of the sample, the microscope steps incrementally along

the Z axis, acquiring a series of optical sections separated by a few micrometers. The resulting set of images, known as a *z-stack*, can be computationally reconstructed into a full three-dimensional volume of the organoid. This volumetric approach is essential to overcome the limitations of traditional 2D cultures, which cannot capture spatial complexity and cell–cell interactions. [3]
Organoids are complex cellular systems that develop internal lumens, apico–basal polarity, and layered architectures that only volumetric imaging can faithfully resolve. CLSM provides superior lateral (200–300 nm) and axial (500–800 nm) resolution compared with conventional microscopy, enabling precise mapping of nuclear biomarkers and the study of dynamic processes within the tissue. [6]

1.2.3 Manual analysis: limitations and variability

Manual analysis of confocal organoid scans follows a structured workflow: biologists first explore 3D renderings and maximum intensity projections (MIP) to assess global morphology (shape, size, regularity), then examine individual 2D sections at different depths to identify internal structures (lumens, buds, crypts), and finally perform semi-quantitative biomarker analysis through cell counting. However, this approach has critical limitations:

- **High time/cost:** Detailed analysis of one organoid takes 15-30 minutes by expert personnel. A medium-scale study (1,000 samples) requires ∼500 hours of work.
- **Observer variability:** Qualitative assessments vary significantly between different biologists (inter-observer) and even by the same biologist at different times (intra-observer).
- **Limited scalability:** Manual methods become impractical beyond hundreds of samples, requiring automation for large-scale studies.

These constraints necessitate robust computational tools for objective, reproducible 3D phenotyping.

1.2.4 Automated approaches: from classical methods to AI

The current methodological landscape for organoid phenotyping has evolved from manual expert analysis toward computational approaches. Classical computer vision techniques extract geometric descriptors (volume, sphericity, eccentricity) and texture features (Haralick matrices, Local Binary Patterns), which are then classified using machine learning algorithms like Random Forest or SVM. These methods provide objective quantification but struggle to capture complex biological patterns. Deep Learning approaches include CNNs processing 2D projections (2.5D) and Graph Neural Networks (GNNs) that model organoids as geometric graphs where cells are nodes and edges encode spatial relationships, achieving data compression while preserving tissue topology.

1.3 Scientific problem and thesis positioning

1.3.1 Limitations of existing approaches

Despite their advances, these methods present significant limitations:

- **Information loss:** 2D projections discard 3D spatial relationships essential for organoid architecture.
- **Feature dependency:** Classical methods rely on hand-crafted descriptors missing subtle biological patterns.
- **High inference time:** GNNs require extensive preprocessing to construct graphs from volumetric data, often taking minutes per sample.
- **Limited scalability:** High-throughput studies generate data volumes beyond current 2D/GNN capabilities.

These constraints necessitate dedicated 3D analysis frameworks.

1.3.2 Limited Prior Work

The scientific work presented in this thesis addresses a research area that remains relatively unexplored in the existing literature. A key challenge is the scarcity of large-scale public datasets of 3D organoids acquired through confocal microscopy with detailed annotations on experimental conditions and EDC-induced phenotypes. While recent advances in automated organoid analysis using deep learning have shown promising results [7, 8], most approaches focus on:

- **Segmentation and tracking** of organoids in 2D brightfield images [7, 9]
- **3D reconstruction** from multiple slices for visualization purposes [9, 8]
- **Morphological analysis** and quantification of geometric features [10, 11, 8]

In contrast, **automatic classification** of 3D organoids based on complete z-stack volumes for toxicological applications remains significantly understudied. A systematic review of PubMed, IEEE Xplore, and Google Scholar (2020-2025, keywords: *"organoid classification deep learning 3D"*, *"EDC organoid screening automated"*) revealed no studies that simultaneously address:

1. Classification (rather than segmentation) of 3D organoids
2. Utilization of entire z-stack volumes (not 2D projections or MIP)
3. Specific application to endocrine-disrupting chemical (EDC) screening

While transformer architectures for 3D computer vision have achieved remarkable success in biomedical segmentation tasks, particularly for organs and tumors [12, 11], capturing long-range spatial dependencies in complex medical volumes, their application to **classification** (as opposed to segmentation) of dynamic biological structures such as chemically-perturbed organoids remains largely unexplored.

1.3.3 Research Questions and Objectives

This thesis addresses the following research questions:

1. Can 3D deep learning architectures effectively classify organoids based on their exposure to different EDCs using full z-stack volumes?
2. How do transformer-based architectures (Swin-UNETR) compare to traditional 3D CNNs (ResNet, DenseNet) for this classification task?
3. What is the optimal balance between model complexity and computational efficiency for organoid phenotype classification?
4. How does spatial resolution affect CNN vs Transformer performance, inference speed, and storage requirements?
5. What is the best strategy for preprocessing this kind of data without losing important information?
6. Can adaptive z-stack slice selection maintain classification performance while reducing computational costs?

To address these questions, the **main objective** of this thesis is to develop and evaluate an end-to-end framework for automatic classification of 3D organoid images exposed to endocrine-disrupting chemicals, comparing transformer-based and convolutional architectures.

Key contributions include:

- A novel application of 3D transformer architectures (Swin-UNETR) to organoid classification for toxicological screening
- Comprehensive comparison of state-of-the-art 3D architectures (Swin-UNETR, ResNet50, DenseNet201) on a custom organoid dataset
- Development of an adaptive z-stack slice selection method based on dissimilarity metrics to reduce computational overhead
- Analysis of attention mechanisms and their interpretability for understanding EDC-induced phenotypic changes
- Open-source implementation and reproducible experimental framework

1.4 Thesis outline

The remainder of this thesis is organized as follows:

Chapter 2 reviews the relevant literature on organoid imaging, deep learning for 3D medical image analysis, and transformer architectures.

Chapter 3 describes the dataset, experimental setup, and the proposed classification framework, including data preprocessing, model architectures, and training procedures.

Chapter 4 presents the experimental results, comparing the performance of different architectures and analyzing the impact of the adaptive slice selection method.

Chapter 5 discusses the findings, limitations, and implications for toxicological screening applications.

Chapter 6 concludes the thesis and outlines directions for future research.

Chapter 2

Related works

2.1 2D and 3D Image Classification: Conceptual Foundations

2.1.1 2D Image Classification

Two-dimensional image classification represents the fundamental paradigm of modern computer vision. [13] Conceptually, the process transforms a pixel grid $I \in \mathbb{R}^{H \times W \times C}$ into a probability distribution over K predefined classes $p(y \in \{1, \ldots, K\}|I)$.

The standard workflow follows four main phases:

- **Preprocessing**: Normalization, resizing to fixed dimensions (typically 224×224), color channel standardization
- **Feature Extraction**: Application of CNNs to extract pattern hierarchies (low levels: edges/textures; high levels: complex objects)
- **Classification Head**: Fully-connected layers with Softmax activation:

$$\hat{y} = \arg\max_k \left(\sigma(f_\theta(I))_k\right), \quad \sigma(z)_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

The loss function guides optimization during training via backpropagation, updating the network parameters θ to minimize distanze between predicted and true labels. Data augmentation techniques (rotations, flipping, color jittering) are essential for mitigating overfitting and improving generalization.

2.1.2 3D Image Classification

Three-dimensional classification extends the 2D paradigm by preserving complete volumetric spatial relationships. Unlike 2D projections $I \in \mathbb{R}^{H \times W \times C}$, a 3D volume $V \in \mathbb{R}^{H \times W \times D \times C}$ encodes full geometric information across all spatial dimensions. [13]

In 3D computer vision, data representation choice fundamentally determines the algorithmic paradigm for feature extraction and spatial modeling. [14] The main

approaches—voxels ($H \times W \times D$), point clouds ($N \times 3$), and meshes—each require specialized processing strategies.

Voxel-based representations leverage 3D CNNs (ResNet3D, DenseNet3D) that extend 2D convolutions across three axes, but suffer cubic memory growth and sparsity issues addressed by sparse convolutions or octree hierarchies that process only occupied voxels.

Point clouds, being unordered sets, demand permutation-invariant operators. PointNet uses pointwise MLPs with global max-pooling, while DGCNN captures local geometry via dynamic kNN graphs. Recent 3D Transformers model long-range dependencies on arbitrary point sets, overcoming convolutional receptive field limitations.

Meshes provide detailed surface geometry (vertices, edges, faces) but require specialized MeshCNN/GraphCNN architectures or surface sampling to convert into processable point clouds. The choice between voxelization, point sampling, or multi-view projections ultimately balances morphological detail against computational feasibility.

Table 2.1: Main data representations for 3D classification

Representation	**Dimensionality**	**Typical Networks**
Voxels	$H \times W \times D$	3D CNN (ResNet3D, DenseNet3D)
Point Clouds	$N \times 3$	PointNet, DGCNN
Meshes	Vertices+Faces	MeshCNN, GraphCNN

However, extending from 2D to 3D introduces significant computational challenges. The curse of dimensionality escalates complexity from $O(HW)$ to $O(HWD)$ parameters, while 3D convolutions process K^3 weights per voxel versus K^2 in 2D, requiring approximately 10× more memory and FLOPs. This rapidly saturates GPU memory even at moderate resolutions (128^3).

Moreover, 2D projections lose critical z-axis spatial relationships essential for understanding true 3D geometry, while pre-trained ImageNet models ($\sim 14M$ samples) fail to capture $H \times W \times D$ volumetric correlations. In stark contrast, 3D datasets like ShapeNet provide only $\sim 50K$ volumes, creating a data scarcity bottleneck that necessitates transfer learning and extensive augmentation strategies.

2.2 Convolutional Neural Networks

2.2.1 Convolutional Neural Networks: Core Principles

Convolutional Neural Networks (CNNs) represent the dominant paradigm in modern computer vision, thanks to their ability to automatically learn spatial feature hierarchies, overcoming the limitations of traditional methods based on manual feature engineering or flattened pixel vectors.

The fundamental principle lies in the discrete convolution operation, where small

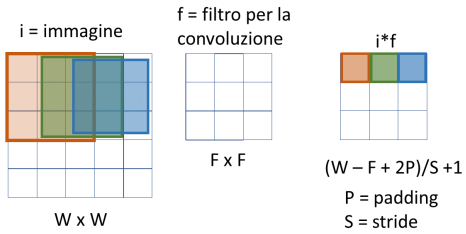

Figure 2.1: Convolution Operation

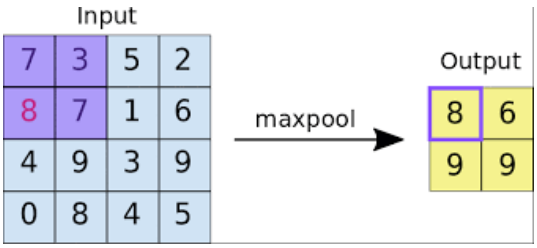

Figure 2.2: Max Pooling Operation

kernels ($K \times K$, typically 3×3) slide over the input image $I \in \mathbb{R}^{H \times W \times C}$ to generate feature maps:

$$F_{i,j,c'} = (I * K_{c'})_{i,j,c'} = \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} \sum_{c=0}^{C-1} I_{i+m,j+n,c} \cdot K_{c',c,m,n} + b_{c'}$$

The sequential architecture organizes layers in a hierarchical progression: shallow layers extract elementary local patterns (edges, textures, gradients), while deeper layers synthesize complex semantic representations by combining information from previous feature maps. The VGGNet-standardized approach [15], based on systematically repeated 3×3 kernels, has demonstrated that controlled depth maximizes expressive capacity while maintaining computational elegance.

Dimensionality reduction through Max Pooling (2×2, stride 2) performs nonlinear selection of the most salient activations, conferring invariance to small translations and reducing overfitting risk.

Non-linearities constitute an essential element of CNNs, introduced through pixel-wise activation functions applied after each convolution. ReLU ($\max(0, x)$) has become the standard due to its computational simplicity and mitigation of the vanishing gradient problem, although it suffers from the "dying neurons" phenomenon. Variants like Leaky ReLU ($\max(\alpha x, x)$ with $\alpha \approx 0.01$) and PReLU address this limitation by allowing gradient flow for negative inputs. These non-linearities transform the feature maps space into a non-linear manifold, ensuring the network's universal expressive capacity needed to model complex high-level functions.

The intrinsic weight sharing of convolutions guarantees drastic parametric efficiency compared to fully-connected layers: a CNN with L convolutional layers requires $O(K^2 C_{in} C_{out})$ parameters per layer, versus $O(HWC_{in}C_{out})$ for FC layers.

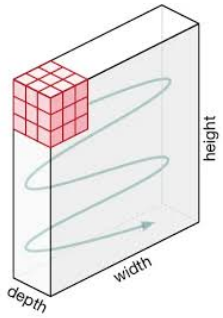

Figure 2.3: 3D Convolution Operation

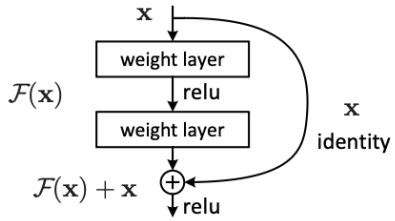

Figure 2.4: Residual Block Structure

Architectural innovations have addressed the limits of extreme depth. ResNet's residual connections ($x_l = F(x_{l-1}, W_l) + x_{l-1}$) preserve gradient flow during backpropagation, enabling networks beyond 100 layers without performance degradation.

To extend these principles to the 3D volumetric domain, 2D kernels $K \times K$ are replaced with $K \times K \times K$, operating on tensors $I \in \mathbb{R}^{H \times W \times D \times C}$. While they fully preserve three-dimensional spatial relationships, 3D CNNs exponentially amplify computational complexity ($O(K^3)$ FLOPs per voxel), requiring transfer learning strategies from pre-trained volumetric datasets to overcome biomedical annotation scarcity.

2.2.2 Residual Networks (ResNet)

The introduction of Residual Networks (ResNet) revolutionized the training of deep convolutional networks, solving the degradation problem observed when the number of layers exceeds a certain threshold. Traditionally, deeper networks did not necessarily guarantee better performance due to optimizers' difficulty in converging toward identity functions. ResNet introduces *residual learning*[16], where each block learns a residual mapping $F(x) = H(x) - x$ rather than the direct mapping $H(x)$:

$$x_l = F(x_{l-1}, W_l) + x_{l-1}$$

The *shortcut connections* (or skip connections) propagate the original input without adding parameters, facilitating gradient flow during backpropagation and enabling stable training of networks with over 100 layers.

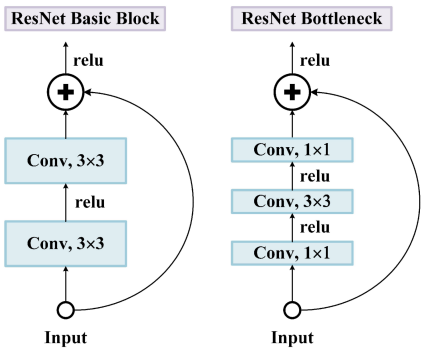

Figure 2.5: ResNet-18 basic block vs ResNet-50 bottleneck block

2.2.3 ResNet-18 vs ResNet-50: Design and Performance

ResNet-18 uses the *basic block* with two 3×3 convolutions, ideal for networks of moderate depth. Its structural simplicity ($11.2M$ parameters) ensures rapid convergence even without residual connections, although they significantly accelerate early training phases. Its efficiency makes it preferable for applications with limited computational resources or moderate-sized datasets. [16]

Table 2.2: ResNet-18 vs ResNet-50 Comparison

Feature	**ResNet-18**	**ResNet-50**	**Ratio**
Block	Basic	Bottleneck	-
Depth	18 layers	50 layers	2.8×
Params	$11.2M$	$25.6M$	2.3×
FLOPs (ImageNet)	$1.8G$	$4.1G$	2.3×
Top-1 Acc (ImageNet)	69.8%	76.1%	+6.3%

ResNet-50 introduces the *bottleneck block* to scale depth while maintaining computational efficiency. The sequence 1×1 (dimensionality reduction), 3×3 (main processing), 1×1 (dimensionality restoration) dramatically reduces parameters compared to three equivalent 3×3 convolutions. Shortcut connections become indispensable, as alternative projections would linearly increase complexity.

In 3D biomedical applications, ResNet-50 pre-trained on volumetric datasets like Med3D [17] demonstrates clear superiority on complex tasks (e.g., lung segmentation: Dice 93.3% vs 87.3% of ResNet-18), thanks to its ability to extract sophisticated morphological patterns. However, it requires $\sim 2.3\times$ more memory and FLOPs, making ResNet-18 preferable when the accuracy/efficiency tradeoff is critical.

2.2.4 DenseNet Architectures

Dense Connected Convolutional Networks (DenseNet) introduce an innovative connectivity paradigm compared to residual architectures, connecting *each layer directly to all subsequent layers* within a Dense Block.
While a traditional network with L layers has only L connections, DenseNet imple-

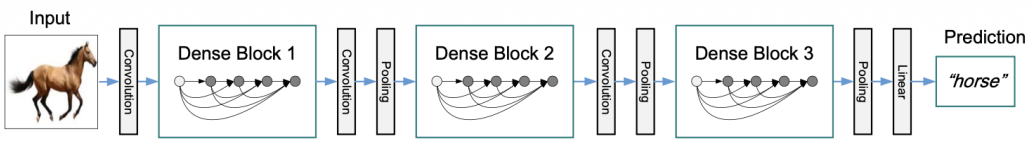

Figure 2.6: Dense Block Structure

ments $\frac{L(L+1)}{2}$ direct connections:

$$x_l = H([x_0, x_1, \ldots, x_{l-1}]),$$

where $[\cdot]$ denotes *concatenation* along the channel dimension (not summation as in ResNet), and $H(\cdot)$ represents 3×3 convolutions + BN + ReLU.

The *growth rate* parameter (k) regulates the number of new channels generated by each layer ($k = 32$ standard), balancing expressive capacity and dimensional growth. The structure alternates *Dense Blocks* with *Transition Layers* (1×1 conv + 2×2 avg pooling), controlling spatial resolution.

DenseNet's dense connectivity yields three fundamental advantages that justify its effectiveness in complex classification tasks:

Dense connectivity creates a global "knowledge state" where each layer l receives the concatenation of all previous feature maps $[x_0, x_1, \ldots, x_{l-1}]$. This *feature reuse* enables layers to add only novel information (controlled by growth rate k) without relearning redundant features extracted earlier, as confirmed by filter weight heatmaps showing deep layers actively leveraging early-stage features.

The dense structure establishes extremely short gradient paths from the loss function to every layer, providing *implicit deep supervision* akin to Deeply Supervised Networks but achieved more efficiently. This direct gradient access effectively mitigates vanishing gradient issues without auxiliary classifiers.

Finally, *parametric efficiency* dramatically reduces parameter counts: DenseNet-121 (∼8M parameters) matches ResNet-50 (25M), while DenseNet-BC models require only $\sim \frac{1}{3}$ of ResNet parameters for equivalent accuracy. Notably, a 100-layer DenseNet (0.8M parameters) rivals a 1001-layer ResNet pre-activation (10M+ parameters), making DenseNets ideal feature extractors for sparse 3D biomedical data where computational efficiency is paramount.

However, progressive concatenation causes *memory explosion* in deep Dense Blocks (hundreds of channels), requiring optimized implementations for 3D medical imaging where volumes amplify the problem.

The *DenseNet-BC* variants (Bottleneck + Compression) apply initial 1×1 convolutions and channel compression in transition layers, maintaining efficiency even on complex volumetric datasets.

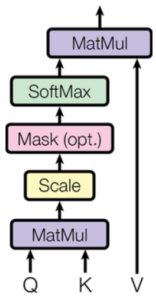

Figure 2.7: Scaled Dot-Product Attention Mechanism

2.3 Transformer-Based Architectures

2.3.1 Foundations of Attention

2.3.1.1 Scaled Dot-Product Attention

The evolution toward Transformers, formalized in the paper "Attention is All You Need" [18], completely abandons recurrent (RNN/LSTM) and convolutional (CNN) mechanisms in favor of an architecture based solely on attention. An attention function operates by mapping a query $Q \in \mathbb{R}^{N \times d_k}$ to a set of key-value pairs $K, V \in \mathbb{R}^{N \times d_v}$, producing an output as a weighted sum of the values V. The weight assigned to each value is computed by measuring the similarity between the query and the corresponding key via their dot product.

The core mechanism of the Transformer is the *Scaled Dot-Product Attention*, which computes the dot products between Q and all keys K, normalizing them by the scaling factor $1/\sqrt{d_k}$ to prevent softmax saturation when operating with large d_k values:

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

Geometrically, the attention matrix $\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \in \mathbb{R}^{N \times N}$ represents a similarity map determining how much each input token influences the others. Unlike CNNs, limited by local receptive fields (3×3, 5×5), self-attention captures long-range dependencies by integrating the entire global context of the sequence in $O(N^2)$ operations, regardless of spatial distance between tokens. This non-local modeling capability is particularly valuable for computer vision tasks, where complex semantic relationships (e.g., occlusions, global context) transcend the locality constraints imposed by convolutions [19].

However, since attention operates on sets of elements and is inherently permutation-invariant, it lacks a natural notion of spatial or sequential order. To preserve the topological structure of the original data, it is essential to inject positional information through *positional embeddings*, typically sinusoidal or learned.

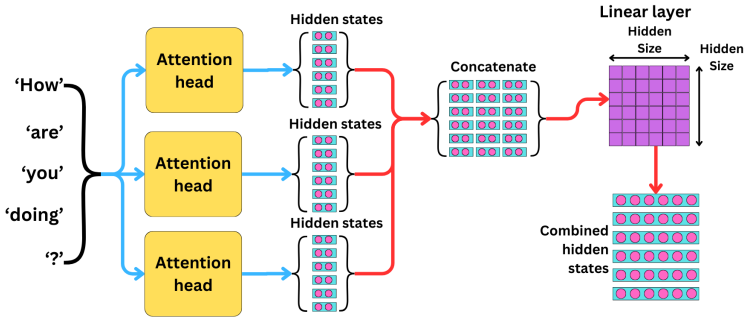

Figure 2.8: Multi-Head Self-Attention Mechanism

2.3.1.2 Multi-Head Self-Attention (MSA)

To enhance representational capacity, *Multi-Head Attention* linearly projects Q, K, V through h parallel subspaces ("heads"), allowing the model to attend simultaneously to different spatial and semantic relationships:

$$\text{MSA}(Q,K,V) = \text{Conc}(\text{head}_1, \ldots, \text{head}_h)W^O, \quad \text{head}_i = \text{Att}(QW_i^Q, KW_i^K, VW_i^V)$$

where $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_k}$ are learned projection matrices for the i-th head, and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ projects the concatenated output back to the original dimension. Each head independently explores distinct subspaces of the representation, while the final concatenation integrates multiple perspectives into a single enriched representation.

This multi-head parallelization overcomes the limitations of single-head attention, which tends to focus on dominant patterns while neglecting subtle relations. Empirically, configurations with $h = 8$–16 heads and $d_k = d_v = d_{\text{model}}/h$ balance expressive power and computational efficiency, making MSA the fundamental engine of all Transformer architectures.

2.3.2 Transformer Block Architecture and Composition

The original Transformer architecture introduced by Vaswani et al. [18] is fundamentally *encoder-decoder*, designed for sequence-to-sequence tasks such as machine translation. However, modern applications have adapted this architecture to three principal configurations based on task requirements:

- **Encoder-Decoder:** Classical Transformer with symmetric encoder and decoder stacks. The encoder processes input tokens into contextual representations; the decoder generates output tokens autoregressively using both self-attention and cross-attention to encoder outputs.
- **Encoder-Only:** Stacks of encoder blocks only (as in BERT, ViT). Used for representation learning, classification, and dense prediction where a single forward pass suffices.

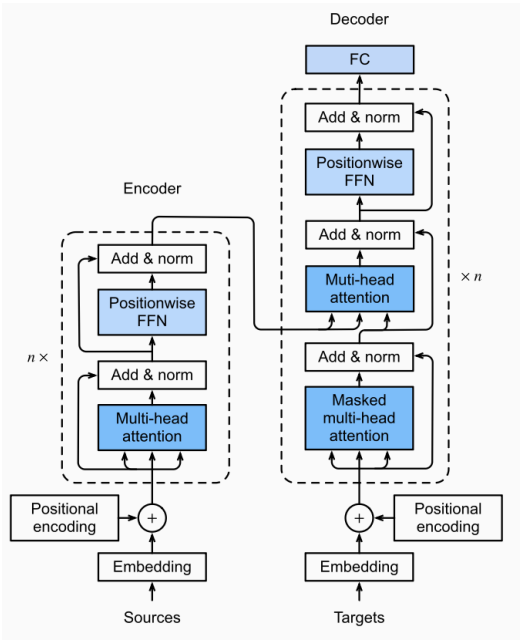

Figure 2.9: Transformer Encoder-Decoder Block Architecture

- **Decoder-Only:** Stacks of causal decoder blocks only (as in GPT). Used for autoregressive generation where each token attends only to previous positions.

Vision Transformers (ViT, Swin) and medical imaging architectures (UNETR, Swin UNETR) predominantly employ the *encoder-only* paradigm, where a stack of identical encoder blocks transforms input patch tokens into hierarchical feature representations. Each encoder block comprises two principal sub-layers: a Multi-Head Self-Attention (MSA) module and a position-wise Feed-Forward Network (FFN), surrounded by residual connections and normalization operations.

2.3.2.1 Positional Encodings.

Since the Transformer lacks recurrent or convolutional structure, it is inherently permutation-invariant and lacks an innate understanding of token ordering or spatial layout. To remedy this, *positional encodings* are added to the initial token embeddings:

$$\tilde{z}_0 = z_0 + \mathrm{PE}(i)$$

where z_0 is the embedded token and $\mathrm{PE}(i)$ encodes position i.

2.3.2.2 Position-wise Feed-Forward Network (FFN).

Following the MSA module, each token is processed through an identical, position-wise FFN consisting of two linear transformations separated by a non-linear activation function:

$$\mathrm{FFN}(x) = \mathrm{Activation}(xW_1 + b_1)W_2 + b_2$$

where $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ projects to an intermediate dimensionality d_{ff} (typically $4 \times d_{\text{model}}$), and $W_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$ projects back to the model dimension. The activation function is commonly GELU (Gaussian Error Linear Unit) or ReLU. Critically, this FFN operates independently and identically on each token position, introducing spatial invariance while expanding and compressing feature representations to enable non-linear feature mixing.

2.3.2.3 Residual Connections and Layer Normalization.

The fundamental design principle governing Transformer stability is the use of residual (skip) connections paired with layer normalization. Each sub-layer is wrapped in a pre- or post-normalization scheme:

$$\text{Sub-layer Output} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

where x is the input to the sub-layer and $\text{Sublayer}(\cdot)$ denotes either MSA or FFN. Modern architectures—including Vision Transformers (ViT) and Swin Transformers—favor *pre-normalization* (Pre-Norm), in which normalization precedes the sub-layer:

$$\text{Sub-layer Output} = x + \text{Sublayer}(\text{LayerNorm}(x))$$

This ordering improves gradient flow, stabilizes training dynamics, and enables larger learning rates without divergence, particularly important when training deep networks with 12+ encoder blocks.

2.3.2.4 Patch Merging and Hierarchical Downsampling.

While classical Transformers maintain uniform spatial resolution throughout the encoder, hierarchical variants (e.g., Swin Transformer) introduce *patch merging* layers between encoder stages. These layers concatenate features from 2×2 (or in 3D, $2 \times 2 \times 2$) adjacent patches and apply a linear transformation:

$$\text{Merge}(\mathbf{z}) = \text{Linear}(\text{Concat}(\mathbf{z}_{2i,2j}, \mathbf{z}_{2i+1,2j}, \mathbf{z}_{2i,2j+1}, \mathbf{z}_{2i+1,2j+1}))$$

This operation halves spatial resolution while doubling the feature dimension, mimicking the hierarchical feature extraction of convolutional networks and enabling multi-scale representation learning.

2.3.2.5 Overall Block Composition.

An individual Transformer encoder block can be formally written as:

$$\mathbf{z}'_\ell = \text{MSA}(\text{LayerNorm}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}$$
$$\mathbf{z}_\ell = \text{FFN}(\text{LayerNorm}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell$$

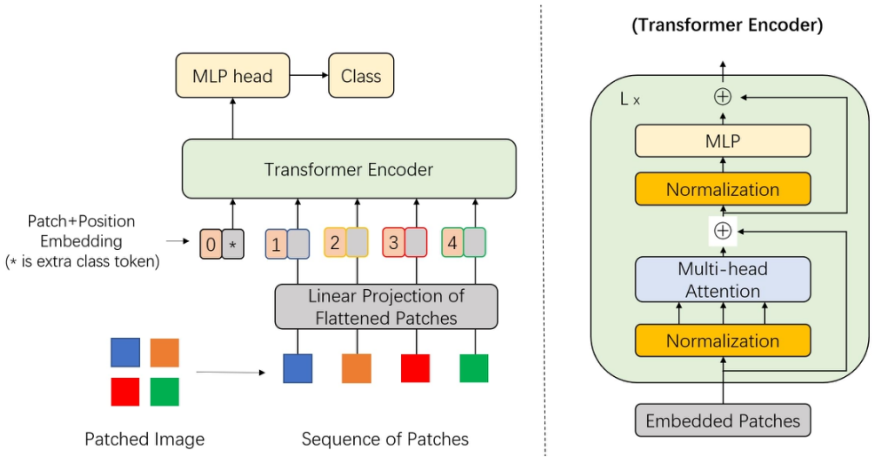

Figure 2.10: Vision Transformer architecture consisting of patch tokenization, positional encoding, stacked Transformer encoders, and the classification head.

Stacking L such blocks creates a deep non-linear feature hierarchy, where early layers extract low-level token patterns and later layers integrate global semantic relationships. The deliberate design—combining attention (for capturing dependencies), feed-forward networks (for non-linear feature mixing), residual connections (for gradient propagation), and normalization (for training stability)—enables Transformers to scale to billions of parameters while remaining computationally efficient and generalizable across diverse modalities (text, vision, volumetric data).

2.4 Vision Transformers (ViT-Style)

The advent of Vision Transformers (ViT) has marked a paradigm shift in computer vision and computational biology, extending the power of self-attention—originally devised for natural language processing (NLP)—to visual tasks [18, 20]. Unlike convolutional neural networks (CNNs), ViTs treat the input image not as a spatially correlated grid but as a sequence of tokens, allowing the model to capture long-range dependencies across the entire image domain.

2.4.1 Patch Projection and Tokenization

The processing pipeline begins by decomposing a 2D image into a sequence of non-overlapping patches of fixed size (typically 16×16 pixels) [19, 21]. Each patch is flattened and passed through a linear projection layer to obtain a patch embedding—a dense vector representation of local visual information [22]. Formally, for an input image $x \in \mathbb{R}^{H \times W \times C}$, the patch embedding $z_p \in \mathbb{R}^{N \times D}$ is computed as:

$$z_p = [x_p^1 E; x_p^2 E; \ldots; x_p^N E] + E_{\text{pos}}$$

where $E \in \mathbb{R}^{(P^2 C) \times D}$ is the learnable projection matrix, P is the patch size, $N = HW/P^2$ the number of patches, and E_{pos} the positional embedding matrix.

2.4.2 Positional Encoding and Spatial Awareness

Since self-attention is permutation-invariant, ViTs lack an intrinsic notion of spatial arrangement. To encode information on the 2D structure of the original image, positional embeddings are added to each patch representation.
These embeddings can be:

- **Fixed sinusoidal functions**, inherited from the original Transformer, which inject periodic positional bias;
- **Learned positional embeddings**, optimized jointly with the model parameters; or
- **2D relative positional encodings**, designed specifically for images to model relationships between neighboring patches.

Empirical evidence indicates that learnable or hybrid forms outperform purely sinusoidal ones in vision tasks by capturing spatial deformations more flexibly. [20]

2.4.3 Hierarchical Window-Based Attention

The fundamental limitation of classical Vision Transformers lies in the quadratic computational complexity $O(N^2)$ of global self-attention, which becomes prohibitive for dense prediction tasks and high-resolution or volumetric data [23]. This constraint has motivated the development of *hierarchical* and *locally-scoped* attention mechanisms that preserve global context while scaling linearly with input size.

2.4.3.1 Window-Based Multi-Head Self-Attention (W-MSA)

The Swin Transformer introduces **Window-based Multi-head Self-Attention** (W-MSA), a localized variant of standard MSA that restricts attention computation to non-overlapping spatial windows. Formally, the input feature map is partitioned into a grid of local windows, each containing $M \times M$ patches (or in 3D, $M \times M \times M$ voxels). Attention is computed independently within each window:

$$\text{W-MSA}(\hat{z}) = \text{Concat}(\text{head}_1^{\text{win}}, \ldots, \text{head}_h^{\text{win}})W^O$$

where $\text{head}_i^{\text{win}} = \text{Attention}(\hat{Q}_i W_i^Q, \hat{K}_i W_i^K, \hat{V}_i W_i^V)$ and $\hat{Q}, \hat{K}, \hat{V}$ denote queries, keys, and values scoped to a single window.

By constraining attention to local regions, the computational complexity is reduced from $O(N^2)$ to $O(M^2 \cdot \frac{N}{M^2}) = O(N)$, achieving linear scaling relative to the number of tokens N. This dramatic reduction enables processing of high-resolution images and volumetric datasets without prohibitive memory overhead.

However, W-MSA introduces a critical limitation: isolated windows lack cross-window connections, preventing the model from capturing long-range spatial dependencies and weakening its representational power.

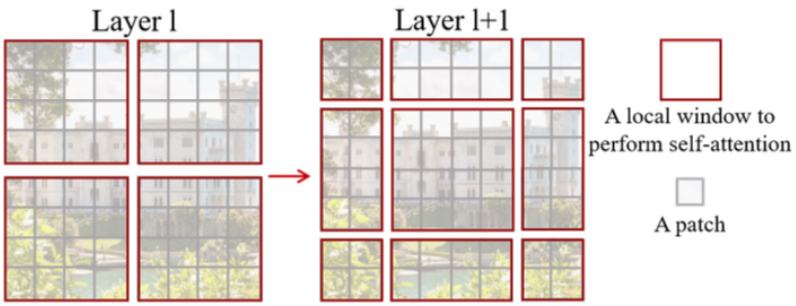

Figure 2.11: Partitions the input into non-overlapping windows, computing attention independently within each window.

2.4.3.2 Shifted and Masked Window Multi-Head Self-Attention (SW-MSA)

To overcome window isolation, the Swin Transformer alternates between standard window partitioning and **Shifted Window Multi-head Self-Attention** (SW-MSA) across consecutive encoder blocks. In a W-MSA layer, the feature map is divided into a regular grid of non-overlapping windows of size $M \times M$ (or $M \times M \times M$ in 3D), and self-attention is computed independently within each window. In the subsequent SW-MSA layer, the same feature map is first *cyclically shifted* by $\lfloor M/2 \rfloor$ pixels along the spatial dimensions, so that the new windows straddle the boundaries of the previous partitioning:

$$\text{Shift}(\mathbf{z}) = \text{Roll}(\mathbf{z}, \Delta h, \Delta w) \quad \text{with} \quad \Delta h = \Delta w = \left\lfloor \frac{M}{2} \right\rfloor,$$

(and analogously a shift Δd along the depth axis in 3D).

Geometrically, this shift ensures that tokens which belonged to distinct windows in W-MSA are now re-grouped into the same window, creating communication channels between adjacent regions without resorting to global attention. To maintain constant window dimensions after the cyclic shift, the tensor is "wrapped" at the borders, generating windows containing tokens that are not truly adjacent in the original spatial layout.

To prevent these artificial correspondences from degrading local modeling, the Swin Transformer introduces **Masked Window Multi-Head Self-Attention** (Mask-MSA). Prior to attention computation, a binary mask (or infinite penalty mask) is constructed that nullifies contributions between tokens belonging to regions that were not contiguous in the unshifted partitioning; at the implementation level, this mask is added to the attention logits before the softmax:

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}} + \mathbf{M}\right)\mathbf{V},$$

where $\mathbf{M}$ is the *attention mask* that assigns very negative values to disallowed token pairs within the same effective window. Thus, SW-MSA achieves an effective compromise: shifted windows establish contact between previously disjoint regions,

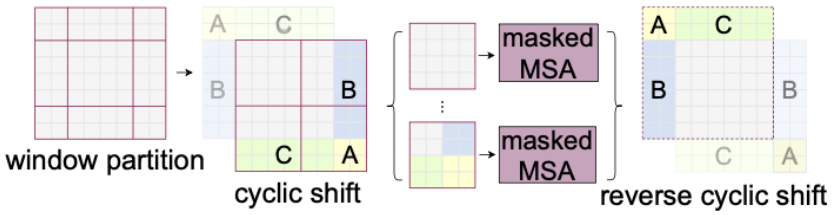

Figure 2.12: Shifted Window Multi-Head Self-Attention (SW-MSA) applies a cyclic shift to the window grid, enabling cross-window connections and enhancing long-range dependency modeling.

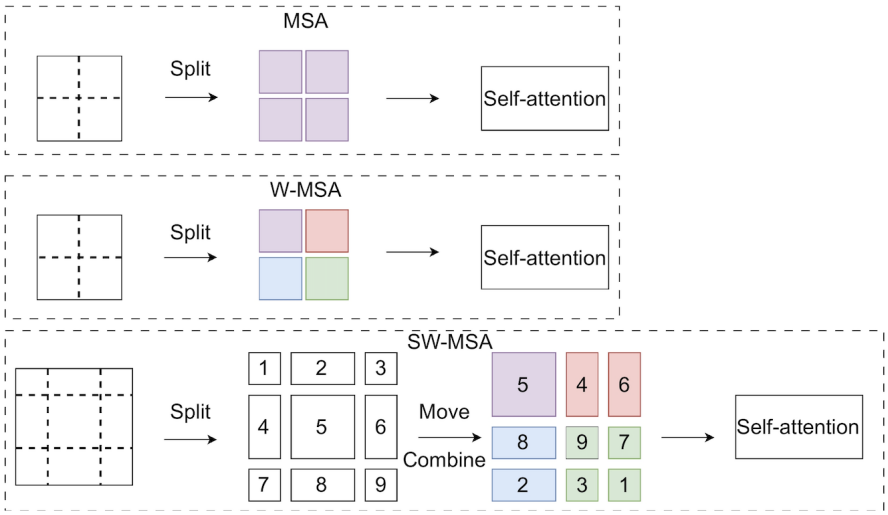

Figure 2.13: Comparison between Global MSA, W-MSA, and SW-MSA mechanisms.

while masking preserves computational locality and ensures complexity remains proportional to the number of tokens.

The regular alternation between W-MSA and SW-MSA blocks

$$[\text{W-MSA} \rightarrow \text{SW-MSA} \rightarrow \text{W-MSA} \rightarrow \text{SW-MSA} \ldots]$$

guarantees that:

1. Each token can exchange information with an increasingly wide spatial neighborhood as depth progresses through the stack.
2. Computational complexity remains linear with respect to the total number of tokens, thanks to the masking constraint within fixed-size windows.
3. The network implicitly builds a hierarchical representation, where long-range dependencies emerge from the composition of multiple layers of locally-shifted attention.

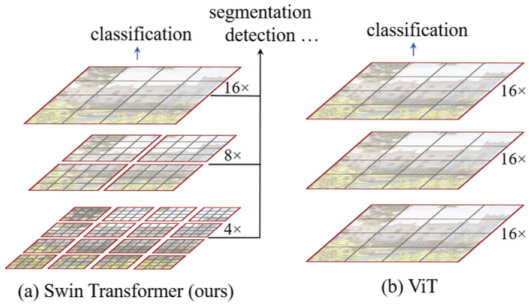

Figure 2.14: Robust multi-scale feature learning through hierarchical window-based attention in Swin Transformers.

2.4.3.3 Hierarchical Feature Aggregation and Efficiency Gains

Beyond the W-MSA / SW-MSA alternation, Swin-style architectures incorporate **patch merging** operations between stages, progressively downsampling the spatial resolution while increasing the feature dimensionality. This hierarchical structure mirrors traditional CNNs, combining coarse contextual information with fine-grained detail in an efficient manner.

The empirical benefits are substantial:

- **Linear Scaling**: Memory and computation scale linearly with image/volume size, enabling processing of 4K images and high-resolution 3D scans.
- **Reduced Parameters**: Hierarchical designs with local windows typically require fewer parameters than dense global-attention models, facilitating deployment on resource-constrained devices.
- **Robust Multi-Scale Learning**: The progressive downsampling captures features at multiple scales, improving robustness to variations in object size and contextual clues.

2.4.4 Swin Transformer: Hierarchical Vision Backbone

Introduced by Liu et al. [23] , its core innovation lies in constructing *hierarchical feature representations* through a shifted window partitioning scheme, bridging the gap between global attention models and efficient convolutional backbones.

2.4.4.1 Stage-wise Hierarchical Architecture

The Swin Transformer is organized into *four sequential stages* that progressively generate multi-resolution feature maps, closely mimicking the behavior of hierarchical CNN backbones like ResNet.
The processing pipeline unfolds as follows:

1. **Patch Partitioning.** The input RGB image is initially divided into non-overlapping 4×4 patches, each linearly projected into C dimensional embeddings to form the initial token sequence.

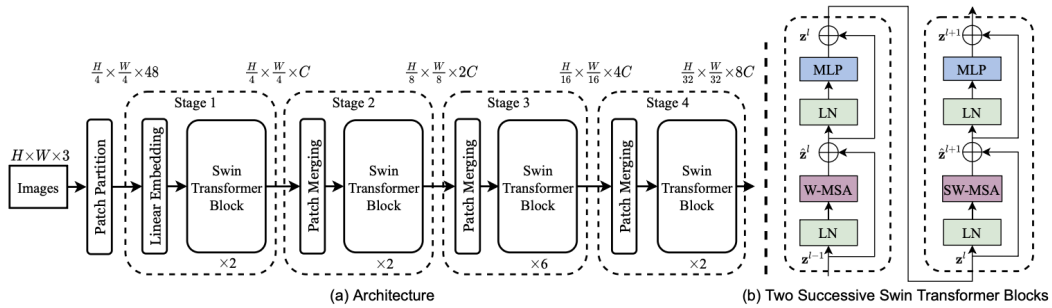

Figure 2.15: Swin Transformer hierarchical stages showing patch merging and alternating W-MSA/SW-MSA blocks.

2. **Swin Transformer Blocks.** Each stage consists of multiple identical Swin Transformer blocks, where standard Multi-Head Self-Attention (MSA) is replaced by window-based attention modules:

$$[\text{W-MSA} + \text{FFN}] \times 2 \quad \rightarrow \quad [\text{SW-MSA} + \text{FFN}] \times 2$$

3. **Patch Merging Layers.** Between stages, *patch merging* reduces spatial resolution while expanding channel dimensionality. Adjacent 2×2 patch features are concatenated and passed through a linear layer:

$$\text{Merge}(\mathbf{z}) = \text{Linear}(\text{Concat}(\mathbf{z}_{i,j}, \mathbf{z}_{i+1,j}, \mathbf{z}_{i,j+1}, \mathbf{z}_{i+1,j+1})) \in \mathbb{R}^{\frac{H}{2}\times\frac{W}{2}\times 2C}$$

This produces the characteristic Swin feature pyramid: $[56 \times 56 \times C, 28 \times 28 \times 2C, 14 \times 14 \times 4C, 7 \times 7 \times 8C]$.

2.4.4.2 Linear Computational Complexity

Unlike the original ViT's global self-attention with quadratic complexity $O((HW)^2)$, Swin Transformer restricts attention computation to local $M \times M$ windows (typically $M = 7$). For a feature map of spatial size $H \times W$ containing $N = HW$ tokens partitioned into $N_w = HW/M^2$ windows, the per-window attention complexity becomes:

$$\text{Complexity} = N_w \cdot M^4 = \frac{HW}{M^2} \cdot M^4 = HW \cdot M^2 = O(HW)$$

This *linear scaling* enables processing of high-resolution inputs essential for dense prediction tasks (semantic segmentation, object detection) without prohibitive memory overhead.

2.4.4.3 Shifted Window Mechanism and Cross-Window Connectivity

The key innovation enabling long-range modeling is the *shifted window partitioning* that alternates between consecutive blocks: - **W-MSA blocks**: Regular non-

overlapping window grid - **SW-MSA blocks**: Windows cyclically shifted by $\lfloor M/2 \rfloor$ pixels

This shift creates new window configurations that straddle boundaries from the previous partitioning, establishing communication channels between previously isolated regions. A relative position bias table $\mathbf{B}_{ij}$ is added to attention logits to encode spatial relationships within windows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + B_{ij} + M\right)V$$

where M denotes the shift-induced attention mask.

2.4.4.4 Internal Block Structure

Each Swin Transformer block follows the Pre-Norm residual pattern:

$$\hat{\mathbf{z}}^l \leftarrow \mathbf{z}^{l-1} + \text{W-MSA/SW-MSA}(\text{LN}(\mathbf{z}^{l-1}))$$
$$\mathbf{z}^l \leftarrow \hat{\mathbf{z}}^l + \text{FFN}(\text{LN}(\hat{\mathbf{z}}^l))$$

The FFN employs two-layer MLP with GELU activation and expansion ratio $4:1$, identical to ViT.

2.4.4.5 Unified Backbone Capabilities

Thanks to its hierarchical design and efficient local attention, Swin Transformer serves as a drop-in replacement for CNN backbones across vision tasks:

- **Image Classification:** Competes with EfficientNet/RegNet on ImageNet-1K
- **Object Detection:** Outperforms prior SOTA when paired with Mask R-CNN
- **Semantic Segmentation:** Achieves top results on ADE20K with UPerNet decoder

The architecture's flexibility—linear complexity, multi-scale features, and hardware-friendly window computations—has established Swin Transformer as the de facto standard for hierarchical vision backbones, spawning 3D extensions (Swin UNETR) for medical imaging applications.

2.4.5 From 2D to 3D Vision Transformers

The extension of the Vision Transformer (ViT) paradigm from bidimensional (2D) to tridimensional (3D) domains represents a crucial advancement in modeling volumetric data, such as Magnetic Resonance Imaging (MRI) or Computed Tomography (CT) scans [24, 25]. In the original ViT framework, a 2D image is divided into non-overlapping patches of size $P \times P$, which are subsequently flattened and projected into a sequence of tokens. The shift to 3D, by contrast, involves *volumetric tokenization*, where the input is no longer a grid of pixels but a voxelized volume decomposed

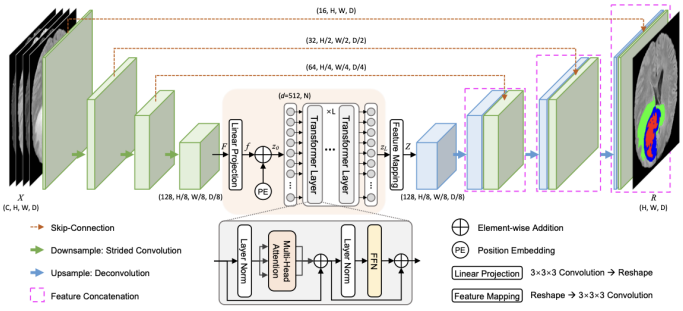

Figure 2.16: Hybrid CNN–Transformer architecture in TransBTS, combining 3D convolutional feature extraction with Transformer-based global context modeling.

into cubic (or parallelepiped) patches, enabling the self-attention mechanism to model spatial dependencies jointly across the height, width, and depth dimensions (H, W, D).

2.4.5.0.1 Hybrid CNN–Transformer Architectures. Approaches such as **TransBTS** and related models integrate the strong local feature extraction capabilities of 3D CNNs with the global relational modeling of Transformers. In these architectures, a 3D convolutional backbone performs initial downsampling and feature extraction, producing compact feature maps that are serialized and processed by a Transformer encoder. This hybrid design effectively reduces computational costs while preserving spatial coherence at multiple resolutions.

2.4.5.0.2 Pure Volumetric Transformers. Fully Transformer-based architectures, such as **UNETR** (Transformers for 3D Medical Image Segmentation), operate directly on a sequence of 3D patches. [24] The architecture proceeds by decomposing the 3D input volume into non-overlapping cubic patches, projecting them into a fixed-dimensional embedding space, and processing the resulting token sequence through stacked Transformer blocks augmented with learnable positional embeddings. Unlike hybrid CNN–Transformer designs, UNETR eliminates the need for a convolutional backbone by directly tokenizing the raw volume, thus capturing global context from the outset. The Transformer encoder outputs are then fused with a convolutional U-Net-style decoder through multi-scale skip connections, enabling the model to combine global semantic understanding with fine-grained spatial detail recovery. This hybrid architecture—Transformer encoder coupled with CNN decoder—leverages the strengths of both paradigms: the ability of self-attention to model long-range volumetric dependencies unconstrained by local receptive fields, and the spatial reconstruction efficiency of transposed convolutions.

2.4.5.0.3 3D Positional Embeddings. A core technical component of this transition is the adaptation of positional encodings. While 2D ViTs typically employ 1D absolute or 2D learned embeddings, their 3D counterparts must encode spatial coordinates along all three axes to ensure topological consistency across depth. Several

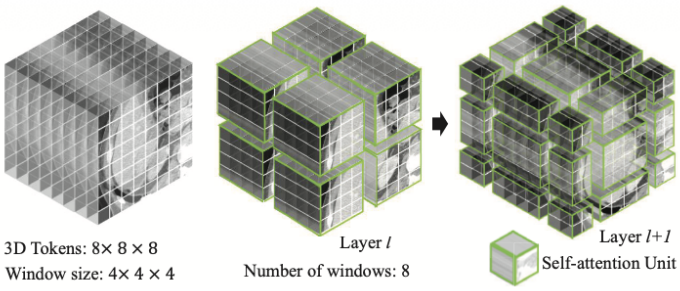

Figure 2.17: Decomposing the 3D input volume into non-overlapping cubic patches

strategies have been proposed:

- **Factorized 3D Positional Embeddings:** Combine separable learnable encodings along the height, width, and depth dimensions (H, W, D) independently, reducing parameter count while maintaining expressiveness. This approach is computationally efficient and has shown effectiveness in volumetric medical imaging [22].
- **Continuous Trivariate Sinusoidal Embeddings:** Extend the classical 1D sinusoidal positional encoding to three dimensions using trivariate frequency bases. This method provides translation-equivariant properties and does not require learnable parameters, though it may sacrifice fine-grained spatial precision compared to learned variants.
- **Relative and Rotary Positional Encodings (RoPE):** Encode spatial relationships relative to neighboring tokens rather than absolute positions, preserving translation consistency and rotational symmetries between volumetric tokens. This approach has recently gained prominence in 3D vision tasks due to its ability to generalize to variable input sizes and improved geometric robustness [22].

The choice of positional encoding strategy significantly impacts model performance: learned embeddings typically outperform fixed sinusoidal encodings for domain-specific tasks, while relative encodings excel in scenarios requiring generalization to unseen resolutions or irregular volumetric sampling patterns.

Despite these advances, 3D Transformers remain highly data-hungry, as they inherit the lack of intrinsic inductive biases (e.g., translation invariance) that characterize CNNs. Current research mitigates this limitation through large-scale pre-training, self-supervised contrastive schemes, and knowledge distillation from CNN-based teachers—crucial strategies to stabilize training and enhance generalization in medical imaging contexts.

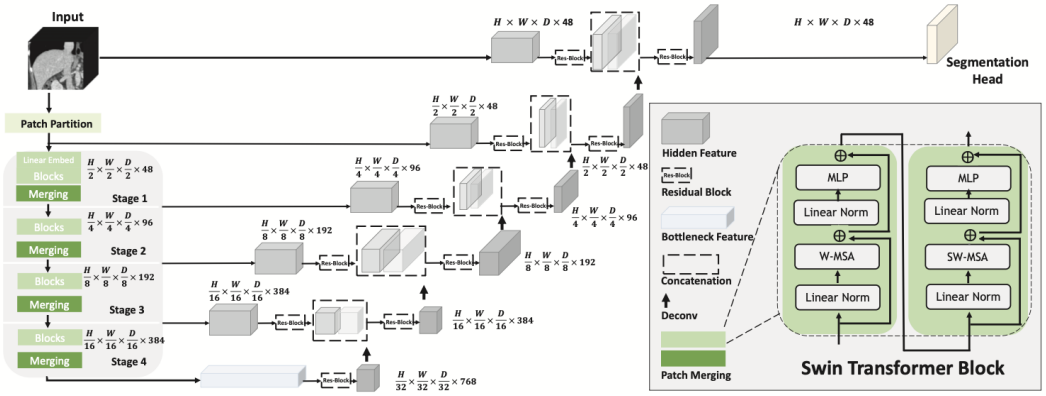

Figure 2.18: Swin Transformer U-Net encoder-decoder architecture. The hierarchical Swin encoder produces multi-scale features fused via skip connections to the decoder, which reconstructs full-resolution segmentation maps.

2.4.6 SwinUNETR: Hierarchical Encoder-Decoder for 3D Medical Imaging

The integration of Swin Transformer's hierarchical attention mechanism with the U-Net encoder-decoder topology yields a powerful architecture for volumetric medical image segmentation. This framework—generically termed **Swin UNETR**—replaces the global-attention encoder of vanilla UNETR with a four-stage Swin Transformer backbone that progressively downsamples spatial resolution while expanding feature dimensionality [25].

2.4.6.1 Architecture and Computational Efficiency

The encoder consists of four hierarchical stages:

$$\text{Stage}_i : \left(\frac{H}{2^i}, \frac{W}{2^i}, \frac{D}{2^i}, C_i\right), \quad i = 1, 2, 3, 4$$

reaching $(H/16, W/16, D/16, 384)$ at maximum compression. Each stage alternates between **W-MSA** (Window-based Multi-head Self-Attention) and **SW-MSA** (Shifted Window Multi-head Self-Attention) blocks, enabling volumetric self-similarity learning at multiple scales.

Encoder outputs are connected via skip connections to corresponding decoder levels, enabling recovery of both coarse semantic context and fine anatomical detail. The critical advantage over standard UNETR is *linear complexity*: window-based attention scales as $O(HW)$ rather than quadratic, reducing memory requirements and enabling processing of larger 3D volumes without degradation.

2.4.6.2 Domain-Specific Optimization: 3D-Organoid-SwinNet

While the base Swin UNETR architecture generalizes across medical imaging tasks (achieving state-of-the-art on BTCV multi-organ and brain tumor segmentation), its application to high-content organoid profiling has motivated domain-specific

refinements. **3D-Organoid-SwinNet** represents such an optimization, maintaining the hierarchical Swin encoder but introducing a specialized decoder design for precise nuclear boundary delineation in densely packed multicellular structures [26].

The key architectural innovation is the replacement of standard convolutional upsampling layers with *multi-scale MLP blocks* for progressive feature reconstruction:

$$\text{Decode}_i = \text{MLP}_i(\text{Upsample}(z_i) \oplus \text{Skip}_i)$$

This design yields several practical advantages:

- **Parameter Efficiency:** Reduces total parameters to 21 million (vs. 131M for standard Swin UNETR), enabling deployment on resource-constrained imaging workstations.
- **Computational Scalability:** MLP-based reconstruction avoids redundant convolution operations, reducing inference time critical for high-throughput organoid screening.
- **Boundary Precision:** Fine-grained feature preservation enables precise nuclear boundary delineation in challenging scenarios: dense nuclear packing, morphological variability across differentiation stages, and confocal z-stack artifacts.

The combined loss function integrates cross-entropy and soft Dice components to balance boundary precision with volumetric accuracy:

$$\mathcal{L}_{\text{combined}} = \alpha \cdot \mathcal{L}_{\text{CE}} + (1 - \alpha) \cdot \mathcal{L}_{\text{Dice}}$$

2.4.6.3 Performance and Clinical Impact

3D-Organoid-SwinNet achieves a Dice coefficient of 94.91 on high-content organoid datasets, demonstrating unprecedented efficacy in resolving tightly packed nuclei within complex multicellular colonies. This lightweight yet high-performing architecture has enabled practical deployment for real-time morphogenesis profiling and precision pharmacological assessment—applications where both accuracy and computational efficiency are essential.

The success of domain-specific decoder optimization in organoid segmentation illustrates a broader principle: while hierarchical Swin Transformer encoders provide a powerful general-purpose foundation for 3D medical imaging, task-specific decoder designs can substantially improve both parameter efficiency and boundary precision without architectural restructuring.

2.5 Comparative Analysis and Advanced Techniques

2.5.1 Comparative Analysis: Convolutional Neural Networks versus Transformer Architectures

The evolution of visual recognition has been marked by a fundamental architectural shift—from the local, hierarchical feature extraction of convolutional neural networks (CNNs) to the global, self-attention-based reasoning of Transformers. Understanding this transition requires examining the technical and conceptual distinctions that shape modern computer vision systems.

2.5.1.1 Receptive Field and Long-Range Dependency Modeling

CNNs operate through kernels of fixed spatial extent (typically 3×3, 5×5, or 7×7), building increasingly large receptive fields through stacking and pooling. This design excels at extracting low-level features—edges, textures, local patterns—but captures long-range spatial relationships only implicitly through deep stacking, incurring substantial latency.

Transformers, by contrast, employ self-attention to compute relationships between *all* patch pairs globally, capturing long-range dependencies and contextual relationships within a single layer [18, 20]. For organoid analysis, this capacity is particularly valuable: nuclear morphology, cell-cell contact patterns, and spatial organization—features spanning multiple scales—are directly accessible to early transformer layers without requiring sequential depth.

2.5.1.2 Inductive Biases and Data Requirements

CNNs encode strong inductive biases: locality (features depend primarily on spatially proximate neighborhoods) and translation invariance (feature detection is independent of spatial position). These biases dramatically reduce sample complexity, enabling effective learning on moderate-sized datasets like ImageNet (1.2M images).

Vision Transformers possess minimal inductive biases, treating spatial structure as learned rather than imposed. This flexibility confers superior representational capacity but at a cost: ViTs are notoriously *data-hungry*, typically underperforming CNNs until trained on massive datasets (JFT-300M, ImageNet-21K). Knowledge distillation techniques (DeiT) partially mitigate this limitation, enabling competitive performance on standard-scale datasets, but the fundamental data dependency remains.

2.5.1.3 Computational Complexity and Scalability

Convolutional networks exhibit linear complexity $O(n)$ with respect to input size n (number of pixels/voxels), maintaining stable memory usage and inference latency even at high resolutions.

Classical Transformers' global self-attention scales quadratically $O(n^2)$, rendering them impractical for high-resolution 2D images and prohibitively expensive for volumetric 3D data. Innovations like the Swin Transformer introduce window-based attention (W-MSA), reducing complexity to linear $O(n)$ while preserving global context through shifted window partitioning (SW-MSA) [23]. This breakthrough has enabled Transformer deployment across dense prediction tasks previously dominated by CNNs.

2.5.1.4 Robustness and Adversarial Resilience

Empirical evidence indicates that Transformers exhibit superior robustness to natural perturbations, noise, and adversarial attacks compared to CNNs [14, 19]. This resilience stems from self-attention's ability to integrate global context: rather than fixating on high-frequency features susceptible to noise, attention mechanisms implicitly balance local detail with global semantic understanding.

In medical imaging—where image quality, motion artifacts, and intensity inhomogeneities are endemic—this robustness translates to more reliable segmentation across diverse acquisition protocols and pathological presentations.

2.5.1.5 Three-Dimensional Data Processing

Processing volumetric data presents unique challenges for each paradigm.

CNNs: Operate on regular voxel grids via 3D kernels, incurring cubic complexity $O(n^3)$ relative to linear dimension n. For a 256^3 volume, this translates to 16 billion voxel operations per forward pass, severely limiting practical resolution and batch size.

Transformers: Decompose volumes into non-overlapping cubic patches, linearizing the 3D structure into a 1D sequence. This reformulation decouples spatial complexity from architectural depth, enabling volumetric reasoning at computational cost comparable to 2D image processing.

Furthermore, Transformers' intrinsic permutation-invariance makes them naturally suited to unordered data like point clouds, a flexibility CNNs fundamentally lack.

2.5.1.6 Parameter Efficiency and Transfer Learning

Dense connections (DenseNet) improve CNN parameter efficiency through feature reuse; however, Transformer-based models demonstrate superior transfer learning capacity, particularly in medical imaging.
Frameworks like Med3D pre-train Transformer encoders on multi-domain medical datasets (3DSeg-8, multi-organ CT/MRI), enabling initialization that dramatically accelerates convergence and improves accuracy relative to ImageNet pre-training—a gap reflecting the substantial domain shift between natural images and biomedical volumes.

2.5.1.7 Hybrid Paradigm: The Future Frontier

Recent architectural innovations have converged on *hybrid* designs combining the complementary strengths of both paradigms:

- **UNETR:** Transformer encoder for global context aggregation, convolutional decoder for fine-grained spatial reconstruction via skip connections.
- **Swin UNETR:** Hierarchical Swin encoder with linear-complexity attention, convolutional decoder preserving sub-voxel anatomical precision.
- **3D-Organoid-SwinNet:** Swin Transformer hierarchical encoder, MLP-based decoder optimized for dense nuclear segmentation in challenging multicellular structures.

These hybrid frameworks leverage Transformers' global reasoning for semantic understanding while exploiting convolutions' locality bias for precise boundary delineation—a synergy that has established state-of-the-art performance across medical imaging benchmarks (BTCV, brain tumor segmentation, organoid profiling).

2.5.1.8 Synthesis and Implications for Biomedical Imaging

The CNN-to-Transformer evolution is not a strict replacement but rather an expansion of the architectural design space. In medical imaging specifically, the combination of Transformer encoders for capturing complex anatomical relationships (morphology, spatial organization, multi-scale pathological signatures) with convolutional decoder mechanisms for preserving fine boundary detail represents the current frontier. This hybrid paradigm has proven particularly effective in challenging 3D segmentation tasks—such as nuclear delineation in dense organoid colonies—where both global morphogenetic context and pixel-precise boundary localization are essential for clinical utility.

2.5.2 Transfer Learning and Pre-training in Medical Image Analysis

Transfer learning—the practice of initializing models with weights learned on large source datasets before fine-tuning on smaller target tasks—has emerged as a critical methodology in medical imaging, where annotated datasets are often limited. Rather than training deep architectures from random initialization ("from scratch"), leveraging pre-trained weights substantially accelerates convergence, improves generalization, and achieves superior final accuracy with substantially fewer target-domain samples.

2.5.2.1 Pre-training Paradigms: Natural Images versus Medical Domains

Historically, medical imaging models have been initialized with weights from ImageNet pre-training—weights learned on millions of natural photographs. However, the substantial visual domain gap between natural RGB photographs and grayscale/multi-channel biomedical volumes (CT, MRI, microscopy) raises a fundamental question:

does ImageNet pre-training remain beneficial when the source and target domains differ drastically?

Recent empirical work addresses this directly. Comparative experiments—training identical architectures from scratch versus from ImageNet pre-training versus from medical-domain pre-training—reveal a consistent pattern [27, 28]:

- **From Scratch:** Convergence is slow, requiring 2–3× more training iterations to reach equivalent accuracy. Final performance plateaus at suboptimal levels due to insufficient gradient signal from limited target data.
- **ImageNet Pre-training:** Provides meaningful acceleration. Even though the visual domain differs, low-level features learned on natural images (edges, textures, oriented patterns) transfer partially, yielding 10–20% accuracy gains over scratch initialization.
- **Medical Domain Pre-training:** Delivers superior performance. Pre-training on large-scale medical imaging corpora (e.g., Med3D on 3D-segmentation datasets, or cross-modality pre-training on multi-organ CT/MRI collections) yields 5–15% additional accuracy over ImageNet initialization.

The performance hierarchy reflects a principle from learning theory: *domain similarity directly influences transfer efficacy.* Features learned on medical data are more relevant to medical downstream tasks than features learned on photographs.

2.5.2.2 Challenges of 3D Pre-training: The Volumetric Data Scarcity Problem

While 2D medical imaging benefits from large-scale pre-training datasets like CheXpert ($224K$ chest X-rays) and MIMIC-CXR ($377K$ radiographs), the transition to **3D volumetric data** introduces significant challenges [28].

- **Exponentially Larger Annotation Cost:** Annotating a single 3D CT volume requires 100–1000× the time investment of a 2D slice, yielding datasets orders of magnitude smaller (e.g., MSD: 750 volumes vs. ImageNet: $1.2M$ images).
- **Computational Barriers:** Training 3D models incurs cubic scaling in memory and compute relative to linear dimension, limiting pre-training scale even for well-resourced institutions.
- **Limited Public 3D Corpora:** Unlike 2D, where standardized benchmarks like MedMNIST v2 provide diverse pre-training opportunities, 3D pre-training remains fragmented across institution-specific datasets (BTCV, MSD, TotalSegmentator) with limited public availability.

MedMNIST highlights this disparity through standardized 3D benchmarks (e.g., OrganMNIST3D, FractureMNIST3D), which demonstrate that 3D models trained

from scratch underperform their 2D counterparts by 10–20% Dice score, underscoring the acute need for effective 3D transfer learning strategies [28].

Consequently, organoid segmentation practitioners must often rely on **cross-dimensional transfer** (2D ImageNet → 3D organoids) or **cross-anatomy transfer** (abdominal CT → organoid microscopy), introducing domain gaps that degrade performance relative to 2D scenarios. This scarcity motivates development of 3D-specific pre-training frameworks and careful fine-tuning protocols to maximize knowledge transfer from limited source domains.

2.5.2.3 Fine-tuning Strategies

Once initialized with pre-trained weights, several fine-tuning strategies exist, each trading off flexibility against overfitting risk:

- **Full Fine-tuning:** All layers are trainable. Computationally expensive but enables substantial weight adaptation to target-specific data distributions. Recommended when target datasets exceed 10^3 annotated volumes.
- **Partial Fine-tuning:** Early encoder layers remain frozen; only late layers and decoder are optimized. Preserves low-level feature generality while allowing high-level adaptation. Often preferred in low-data regimes (< 500 annotated volumes).
- **Layer-wise Learning Rate Decay:** Assigns smaller learning rates to early encoder layers (which generalize broadly) and larger rates to later layers and decoder. Balances preservation of transferred features with target-specific learning.

For organoid segmentation—where annotated 3D volumes remain scarce relative to broader medical imaging—partial fine-tuning or layer-wise decay typically outperform full fine-tuning, preventing catastrophic forgetting of learned feature hierarchies while adapting to organoid-specific nuclear morphology and imaging artifacts.

2.5.3 Monte Carlo Dropout: Uncertainty Quantification in Deep Learning

Clinical deployment of deep learning models demands not only high accuracy but also *calibrated uncertainty estimates*—a measure of model confidence that reflects the true likelihood of prediction errors. **Monte Carlo (MC) Dropout** provides a principled approach to uncertainty quantification by leveraging dropout—a regularization technique—as a mechanism for Bayesian approximation. Unlike traditional dropout, which is disabled at test time, MC dropout remains active during inference, enabling the model to generate a distribution of predictions from which uncertainty can be estimated.

2.5.3.1 Theoretical Foundations: Dropout as Bayesian Approximation

The theoretical basis for MC dropout rests on an elegant connection between dropout and approximate Bayesian inference [29, 30].

A neural network with dropout applied before each weight layer is mathematically equivalent to an approximate posterior over a deep Gaussian Process (GP). During training, dropout implicitly performs *variational inference*, minimizing the Kullback-Leibler (KL) divergence between a learned approximate posterior and the true posterior:

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{task}} + \frac{1-p}{2N\lambda}\|\mathbf{W}\|_2^2$$

where p is the dropout rate, N is dataset size, λ is a weight decay coefficient, and the regularization term emerges naturally from the variational formulation. This interpretation recasts dropout—often viewed as ad-hoc regularization—as a principled approximation to Bayesian model averaging over an exponential family of sub-networks.

During inference, each stochastic forward pass samples a different sub-network (by randomly zeroing units according to the dropout mask). The ensemble of these samples approximates the posterior predictive distribution:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) \approx \frac{1}{T}\sum_{t=1}^{T} p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{W}^{(t)})$$

where $\mathbf{W}^{(t)}$ denotes the weights of the t-th sampled sub-network and T is the number of MC samples.

2.5.3.2 Implementation in Deep Learning Models

MC dropout is straightforward to implement in practice and requires minimal architectural modification:

1. **Training Phase (unchanged):** Train the network with dropout enabled at specified layers (typically before dense layers or after convolutional blocks), using standard loss functions and optimization.

2. **Test Phase (modified):** For a given input $\mathbf{x}^*$, perform T independent forward passes with dropout *enabled*:

   $$\hat{\mathbf{y}}_t^* = f(\mathbf{x}^*; \mathbf{W}^{(t)}), \quad t = 1, \ldots, T$$

   where each forward pass receives a different stochastic dropout mask.

3. **Predictive Mean and Variance:** Compute the mean prediction and estimate uncertainty:

   $$\hat{\mathbf{y}}^* = \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{y}}_t^*, \quad \sigma^2 = \frac{1}{T}\sum_{t=1}^{T}(\hat{\mathbf{y}}_t^* - \hat{\mathbf{y}}^*)^2$$

4. **Computational Efficiency:** Empirical studies show that beyond approximately $T = 20$ Monte Carlo samples, marginal improvements in uncertainty estimates plateau [31]. This permits practical inference with modest computational overhead (20–50× that of single deterministic pass).

2.5.3.3 Sources of Uncertainty

MC dropout estimates capture two conceptually distinct uncertainty sources:

- **Aleatoric Uncertainty (Data Uncertainty):** Noise and randomness inherent in the data—e.g., image acquisition noise, anatomical variability, overlapping class statistics. This uncertainty cannot be reduced by acquiring more data and reflects irreducible randomness in the problem domain.
- **Epistemic Uncertainty (Model Uncertainty):** Reflects the model's lack of knowledge—regions of input space far from training data, conflicting training examples, or insufficient model capacity. This uncertainty decreases as more relevant data is acquired.

MC dropout provides a marginal estimate of predictive uncertainty that conflates both sources. For clinical applications, distinguishing between them is valuable: high aleatoric uncertainty suggests intrinsic problem difficulty, while high epistemic uncertainty suggests the model requires additional training data or capacity.

2.5.3.4 Applications in Medical Imaging

In clinical settings—where false confidence in predictions can have catastrophic consequences—MC dropout provides multiple benefits [31, 32]:

2.5.3.4.1 Test-Retest Repeatability. A fundamental clinical requirement is *reproducibility*: identical images of the same anatomical region should yield consistent diagnoses across independent model evaluations. Traditional deterministic networks often exhibit boundary instability—near decision boundaries, small image variations cause dramatic prediction swings. MC dropout's ensemble averaging reduces this instability substantially, improving test-retest agreement by 16 percentage points (in terms of 95 limits of agreement) on real clinical datasets [31].

2.5.3.4.2 Class Boundary Calibration. Clinical classification often involves ordinal categories (e.g., normal → dysplasia → cancer) or binary decisions near severity thresholds. At class boundaries, predictions are inherently uncertain. MC dropout explicitly quantifies this uncertainty, enabling clinicians to flag ambiguous cases for expert review rather than forcing unreliable binary decisions.

2.5.3.4.3 Probability Calibration. Output probabilities from deep networks are often poorly calibrated—a prediction of 95% confidence may not correspond to 95%

true success rate. MC dropout naturally produces better-calibrated probabilities: the mean prediction and its variance jointly characterize the posterior, aligning predicted confidence with actual error likelihood.

2.5.3.4.4 Robustness to Image Artifacts. Medical images are corrupted by acquisition artifacts, patient motion, and scanning parameter variation. Models trained without uncertainty awareness often exhibit overconfidence on noisy data. MC dropout-enabled models gracefully handle degraded inputs: uncertainty estimates rise appropriately, flagging potentially unreliable predictions.

Chapter 3

Methodology

3.1 Dataset and Experimental Protocol

3.1.1 Organoid Dataset Description

The dataset is a **structured collection of 960 3D confocal microscopy images of mouse prostate organoids**, developed for phenotypic classification of tumor and non-tumor conditions. It was assembled over four years (2022–2025) through a collaboration between the IPMC laboratory (Nice) and the Metatox team (Université Paris Cité), totaling approximately ∼850 GB of raw data.

Table 3.1: Distribution of the dataset by laboratory and phenotypic class.

Laboratory	**Cauliflower**	**Compact**	**Cystiques**	**Total**
Metatox (Paris)	106 (40.8%)	1 (0.4%)	153 (58.8%)	260
IPMC (Nice)	379 (54.1%)	42 (6.0%)	279 (39.9%)	700
Total	**485**	**43**	**432**	**960**

Images were acquired using confocal microscopy with 20× and 40× objectives and a single grayscale channel. Samples were organized into three distinct sets based on the observed phenotype. Each sample is stored as a stack of images in `.tif` format, with original resolutions of 512×512, 1024×1024, or 2048×2048 pixels per 2D slice and variable depth ranging from 38 to 666 slices per 3D volume, in either `uint8` or `uint16` formats.

Table 3.2: Distribution of the dataset by phenotypic class and magnification/resolution. Cells report N (%) relative to the row total (magnification).

	512×512	**1024×1024**	**2048×2048**	**Total**
Cauliflower 20x	2 (0.6%)	328 (98.2%)	40 (1.2%)	370
Cauliflower 40x	0 (0.0%)	106 (100%)	0 (0.0%)	106
Compact 20x	1 (2.3%)	41 (95.3%)	0 (0.0%)	42
Compact 40x	0 (0.0%)	1 (100%)	0 (0.0%)	1
Cystiques 20x	1 (0.4%)	74 (30.7%)	214 (68.9%)	289
Cystiques 40x	2 (1.4%)	136 (97.1%)	0 (0.0%)	138
Total	**6**	**686**	**254**	**946**

The heterogeneous nature of the files reflects the different data sources (laboratories and experimental setups) and varying acquisition conditions. Some volumes contain multiple organoids per 3D image, while others show only a single organoid; additionally, part of the samples suffers from blur, background noise, or imaging artifacts, making the classification task more challenging.

Table 3.3: Distribution of the dataset by phenotypic class and magnification/format. Cells report N (%) relative to the class total.

Class	**20x uint8**	**20x uint16**	**40x uint8**	**Total**
Cauliflower	31 (6.5%)	348 (73.1%)	106 (22.3%)	485
Compact	42 (97.7%)	0 (0.0%)	1 (2.3%)	43
Cystiques	85 (19.9%)	209 (48.9%)	138 (32.3%)	432
Total	**158**	**557**	**245**	**960**

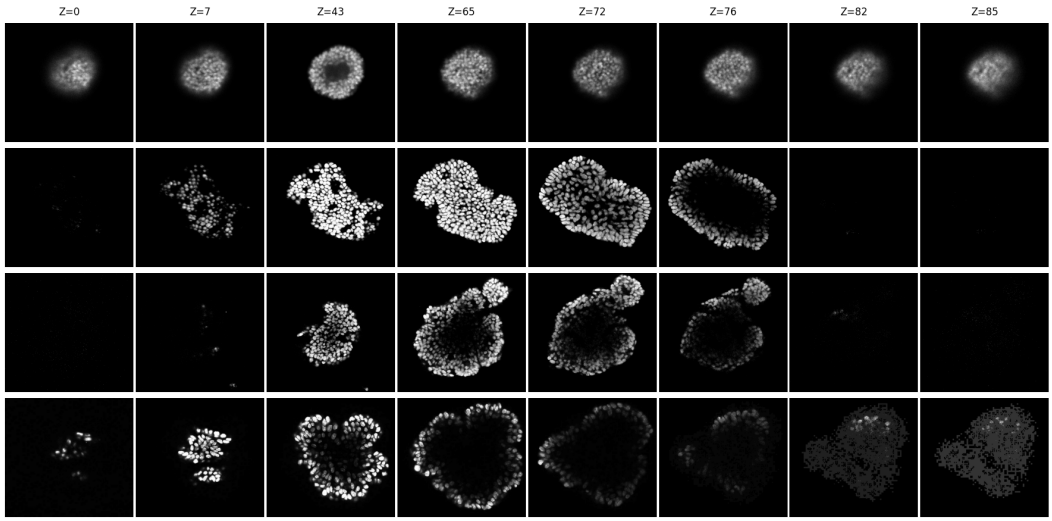

Figure 3.1: Visualization of samples slices.

3.1.2 Preprocessing Pipeline

To address the dataset heterogeneity, a standardized preprocessing pipeline was implemented, consisting of the following sequential steps:

1. **Quality filtering**: Removal of problematic samples affected by blur, excessive noise, or inhomogeneities.
2. **Format normalization**: Conversion of all samples to `uint8` format for consistent bit depth.
3. **Manual cropping**:
    - Background removal through precise manual segmentation to maximize data quality retention.
    - Isolation of individual organoids from multi-organoid volumes.

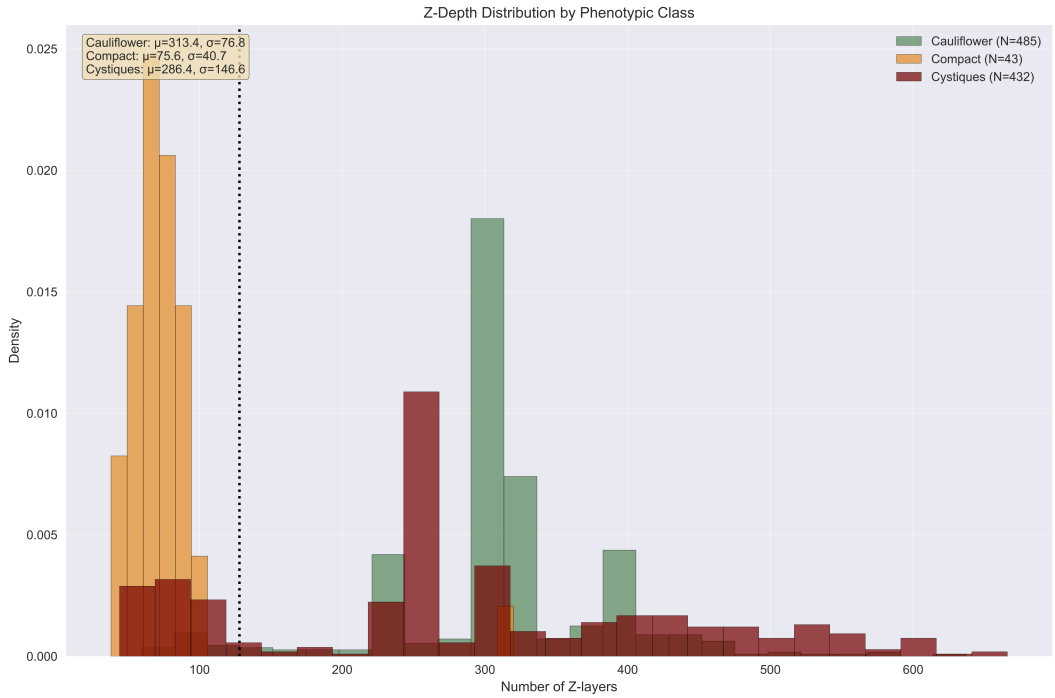

Figure 3.2: Distribution of phenotypic classes before Z-depth standardization.

Manual cropping was specifically chosen over automated approaches to preserve maximum dataset quality. Automatic methods often introduce boundary artifacts, partial organoid cropping, or suboptimal background removal that can compromise downstream model performance—issues avoided through human supervision.

4. **Spatial normalization**: Resizing of all 2D slices to a uniform 512×512 resolution.

5. **Z-depth standardization**: Uniform volume depth of 128 slices per sample:

   - Linear downsampling along the Z-axis for volumes exceeding 128 slices.
   - Zero-padding with empty slices (all zeros) for volumes with fewer than 128 slices.

This pipeline ensures maximum data quality preservation while achieving uniformity. Storage requirements were reduced from over 850 GB to 80 GB (91% compression) without loss of phenotypic information.

Table 3.4: Preprocessing pipeline impact on dataset characteristics.

Metric	Raw Dataset	Processed Dataset
Total samples	960	806
Storage	∼850 GB	80 GB
Slice resolution	Mixed	512×512
Z-depth	38–666 slices	128 slices
Format	uint8/uint16	uint8

Table 3.5: Dataset statistics after preprocessing pipeline.

Metric	**Class 0**	**Class 1**	**Class 2**	**Total**
Number of samples	391	43	369	803
Percentage (%)	48.69	5.35	45.95	100.00

Table 3.6: Multi-resolution dataset variants and relative memory footprint.

Resolution	Memory (GB)	Relative size (%)	Reduction factor
Original raw	860	100.0	1.00x
$512 \times 512 \times 128$	54.0	6.28	16.00x
$256 \times 256 \times 128$	14.0	1.63	61.43x
$128 \times 128 \times 128$	3.3	0.38	260.61x
$32 \times 32 \times 128$	0.22	0.03	3904.00x

3.1.3 Multi-Resolution Downsampling

To test the robustness of the proposed models at different spatial resolutions, downsampled versions of the pre-processed dataset were created at lower resolutions. Specifically, dataset versions at 256×256, 128×128, and 32×32 pixels per slice were generated, while maintaining the standardized Z-depth of 128 slices. This allows us to assess the impact of spatial resolution on classification performance and computational efficiency.

3.1.4 Data Augmentation Strategy

To improve generalization and robustness to appearance and pose variations, we adopt a 3D data augmentation pipeline applied on-the-fly during training to approximately 70% of the mini-batch samples. The augmentation operates on volumetric inputs and includes both geometric and intensity-based transformations, while preserving the underlying anatomical semantics of the organoids.

From a geometric perspective, we apply random flips along each spatial axis independently, i.e. depth, height, and width, with a probability of 0.5 per axis. In addition, we use random 90-degree rotations in 3D space (up to three successive quarter turns) around specified spatial axes, which allows the model to learn invariances to discrete orientation changes without introducing interpolation artefacts. These transformations expose the network to a wide range of plausible spatial configurations of the same underlying structure.

On the intensity side, we employ a stochastic one-of strategy that samples a single intensity transform per volume from a predefined set. Concretely, we randomly apply either additive Gaussian noise, contrast adjustment with a gamma factor sampled in a predefined range, or an intensity shift, each selected according to specified weights. This design encourages robustness to acquisition-related variability such as noise levels, contrast differences, and baseline intensity shifts, while keeping the augmented samples physiologically realistic.

3.1.5 Train/Validation/Test Split

The dataset is first divided into training and test sets using a 90% / 10% split, ensuring that the held-out test set remains fixed and is not used during model selection.

Table 3.7: Train-Test split statistics (90%-10% split).

Dataset	Class 0	Class 1	Class 2	Total
Train set (89.8% of original)				
Number of samples	351	38	332	721
Percentage (%)	48.68	5.27	46.05	100.00
Test set (10.2% of original)				
Number of samples	40	5	37	82
Percentage (%)	48.78	6.10	45.12	100.00

The training subset (90% of the data) is then further partitioned into training and validation sets with an 80% / 20% ratio, which is used to tune hyperparameters and perform early stopping.

Within this second-stage split (train/validation), we consider different splitting strategies to investigate their impact on model performance and class balance:

- **Random split**: the samples are assigned to training and validation sets uniformly at random, without explicitly enforcing constraints on the class distributions.
- **Stratified split**: the split preserves the original class proportions in both training and validation sets, ensuring that each subset is representative of the global distribution.
- **Balanced split**: the classes are forced to have an equal proportion in each subset, i.e. approximately 33.33% of the samples per class, to mitigate class imbalance effects, but using only 129 samples in total (43 per class) for training and validation.
- **Percentage-based split**: the class proportions in the subsets follow a predefined ratio (e.g. 4:2:4 across the three classes), allowing controlled reweighting of specific classes while keeping the total size fixed. Specifically, the number of samples per percentage unit is determined by the minimum class size in the training set.

3.1.6 Pre-training Datasets for Backbones

For the SwinViT-based architectures, we leverage the BraTS 2021 (BRaTS21) dataset, which comprises multi-parametric MRI scans from over 2,000 patients with brain gliomas, totaling approximately 8,000 volumes across four modalities (T1, T1CE, T2, FLAIR) at 1 mm^3 isotropic resolution. This large-scale, heterogeneous collection captures diverse tumor subregions (enhancing tumor, edema, necrosis) and imaging

protocols from multiple institutions, providing robust pre-training for 3D transformer-based models targeting volumetric medical image analysis.

For ResNet backbones, we adopt the Med3D pre-trained weights from the 3DSeg-8 dataset, an aggregated collection of 2,000 3D volumes spanning multiple modalities (CT and MRI) and anatomical regions including brain, heart, liver, prostate, spleen, and pulmonary structures. The heterogeneous nature of 3DSeg-8—derived from public segmentation challenges—enables learning of generalizable 3D medical features via co-training across domains, which transfer effectively to downstream tasks like organoid classification.

3.2 Proposed Classification Pipeline

3.2.1 Overall Architecture

The proposed 3D organoid classification architecture is composed of three main components: a spatial preprocessing layer (Spatial Abstractor/Slice Selector), a feature extraction backbone, and a final classification head. The data pipeline begins with a standardized 3D volume input, which is first resized to ensure compatibility with the selected backbone. This volume is then processed by the spatial preprocessing layer, responsible for reducing both the in-plane resolution (e.g., from 512×512 to 256×256) and the axial depth (e.g., from 128 to 64 slices) through specific downsampling strategies tailored to the input characteristics. The preprocessed volume is subsequently partitioned into patches—either overlapping or non-overlapping—which are independently forwarded through the chosen backbone to extract multi-scale feature representations. These feature maps are then aggregated (or fused in the case of overlapping patches) and converted into a global feature embedding via 3D Adaptive Average Pooling, followed by flattening. To evaluate the impact of different feature extraction paradigms, several backbone architectures are considered. Specifically, ResNet18 and ResNet50 pretrained on the Brains18 dataset (MedicalNet) are employed to exploit domain-adapted convolutional representations, whereas the SwinViT model, pretrained on BraTS21, provides a transformer-based alternative that captures long-range dependencies in volumetric data. DenseNet121 is trained from scratch due to the absence of suitable medical pretraining datasets, while the so-called SwinUNETR leverages the encoder component of the original SwinViT architecture combined with its subsequent convolutional Encoder10 block, effectively integrating transformer-based context modeling with localized convolutional refinement. The resulting feature vector is passed to the classification head, which in its baseline configuration consists of a single fully connected layer with softmax activation, yielding class probabilities for the three organoid categories. An alternative configuration includes the use of NoaH, the attention-based classification head. Between the backbone and classification head, dropout layers provide regularization to mitigate overfitting while enabling Monte Carlo Dropout-based uncertainty quantification, explored in detail in the dedicated uncertainty section.

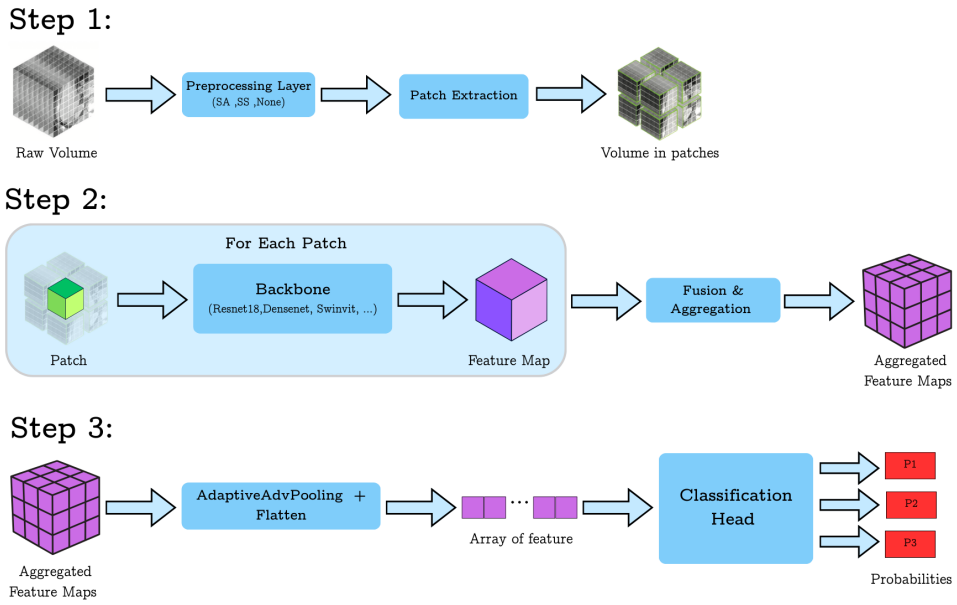

Figure 3.3: Overall Architecture of the Proposed 3D Organoid Classification Pipeline. The pipeline consists of a spatial preprocessing layer (Spatial Abstractor/Slice Selector), a feature extraction backbone (e.g., ResNet or Swin Transformer), and a classification head. The data flow is illustrated from input volume to final class probabilities.

3.2.2 Spatial Abstracter: Volumetric In-Plane Downsampling

Spatial dimensionality in the $H \times W$ dimensions is reduced through the `SpatialAbstracter` module, a lightweight preprocessing layer designed to standardize input volumes to a target resolution (e.g., 128×128) compatible with downstream backbones. This module operates on tensors of shape $[B, C, D, H, W]$, preserving axial depth D while downsampling each D-stack independently via bilinear interpolation.

The forward pass converts the input tensor to NumPy on CPU for compatibility with scikit-image's `resize` function:

```
resized[b, c] = resize(x_np[b, c], (D, target\_H, target\_W),
                       order=1, preserve\_range=True, anti\_aliasing=True)
```

The result is returned as a PyTorch tensor on the original device.

`skimage.transform.resize` performs N-dimensional interpolation for resizing, crucial for downsampling to prevent aliasing artifacts. Bilinear interpolation (`order=1`) computes each output pixel $I_{\text{out}}(i, j)$ as a weighted average of the four nearest input neighbors:

$$I_{\text{out}}(i, j) = \sum_{m,n \in \{0,1\}} w_m w_n \cdot I_{\text{in}}(i + m, j + n),$$

where weights $w_0 = 1 - \delta_x$, $w_1 = \delta_x$ derive from fractional offsets $\delta_x, \delta_y \in [0, 1)$. This ensures smooth sub-pixel continuity, ideal for organoid textures.

Gaussian anti-aliasing ($\sigma \approx (s-1)/2$, $s =$ downsampling factor) pre-filters inputs, attenuating Nyquist frequencies to avoid moiré patterns in sparse structures.

This configuration yields a resized volume, e.g., $[B, C, D, 128, 128]$, reducing computational load by $\sim 16\times$ (from 512^2) while mitigating aliasing in sparse organoid structures.

3.2.3 Slice Selector: Adaptive Axial Dimensionality Reduction

Axial and spatial redundancies in 3D organoid volumes are efficiently pruned by the `SliceSelector` module, a non-parametric algorithm that greedily selects the most representative 2D slices along each axis (D, H, W) to achieve target dimensions (e.g., $64 \times 128 \times 128$) with minimal information loss. Unlike uniform downsampling, this method prioritizes morphological diversity, reducing slices by up to 75% (e.g., 128→32) in $\mathcal{O}(N^2)$ time while preserving key organoid structures.

The module supports the `'feature_variance'` method, processing volumes $[D, H, W]$ sequentially along each axis. For a given axis, slices are extracted as 2D views via `np.moveaxis`, yielding N_{axis} projections.

1. Per-Slice Feature Extraction. Each 2D slice is characterized by an 8D descriptor vector:

- Intensity statistics: mean, std, median, IQR (25^{th}-75^{th} percentiles).
- Gradient features: mean/magnitude/std/max of $\nabla I = \sqrt{(\partial_x I)^2 + (\partial_y I)^2}$.
- Information content: Shannon entropy $H = -\sum p_i \log_2(p_i + \epsilon)$ from normalized histogram ($bins = 256$).

Features are standardized via `StandardScaler` for metric invariance.

2. Similarity Modeling. A symmetric affinity matrix $\mathbf{S} \in^{N \times N}$ is computed from Euclidean distances in feature space:

$$d_{ij} = ||\mathbf{f}_i - \mathbf{f}_j||_2, \quad S_{ij} = 1 - \frac{d_{ij}}{\max(d)}.$$

3. Greedy Dissimilarity Selection. Indices are selected to maximize coverage:

- **Spatial partitioning**: Divide axis into $r = \min(n_{\text{target}}//4, 8)$ regions; pick lowest average-similarity slice per region.
- **Iterative diversification**: For remaining slots, greedily add the slice maximally dissimilar ($\max(1 - \mathbf{S}_{:,\text{sel}})$) to current set.

The process repeats per axis, yielding a sparsified volume via index-based slicing.

3.2.4 Overlapping Feature Map Fusion via Gaussian Blending

A cohesive 3D feature volume is reconstructed from overlapping patches extracted during sliding-window inference through a Gaussian-weighted blending mechanism. This fusion is performed directly in the feature space without intermediate rescaling, utilizing MONAI's importance map utilities for efficient computation. Boundary artifacts are mitigated while high-fidelity feature continuity is preserved.

The core function accepts:

- `feats`: Tensor of shape $[N, C, fD, fH, fW]$, containing features from N overlapping patches.
- `coords`: List of patch origins $[(b, z, y, x), \dots]$ in the original input space.
- `patch_size`: Tuple (pD, pH, pW) defining patch dimensions in input space.
- `step`: Tuple (sD, sH, sW) specifying the sliding step between patches.

The algorithm is structured as follows:

1. Output Grid Computation. Unique coordinates across patches are used to derive output volume dimensions in feature space, scaled from input-space extents via the downsampling factor (typically 32× from SwinUNETR stages):

$$\text{out\_D} = \lceil (\max(z_i) + pD) \cdot \frac{fD}{pD} \rceil, \quad \text{similarly for H, W.}$$

Step sizes are scaled analogously to feature resolution.

2. Gaussian Importance Map Generation. A 3D Gaussian importance map $\mathbf{I} \in [1, 1, fD, fH, fW]$ is generated using MONAI's `compute_importance_map` with $\sigma_{\text{scale}} = 0.125$. Central patch regions are emphasized while edges are attenuated:

$$I(\mathbf{r}) = \exp\left(-\frac{||\mathbf{r} - \boldsymbol{\mu}||^2}{2\sigma^2}\right), \quad \boldsymbol{\mu} = \text{patch center.}$$

A constant fallback is employed for robustness.

3. Weighted Accumulation. For each patch i:

- The origin $\mathbf{c}_i = (z_i, y_i, x_i)$ is mapped to feature coordinates $\mathbf{c}_i^f$.
- End-coordinates are clamped to output bounds: $\mathbf{c}_{\text{end}}^f = \min(\mathbf{c}_i^f + (fD, fH, fW), \text{out})$.
- Accumulation occurs as: $\mathbf{O}[\mathbf{c}_i^f : \mathbf{c}_{\text{end}}^f] += \mathbf{F}_i[\dots] \odot \mathbf{I}[\dots]$, with a parallel count map $\mathbf{C}$ tracking weights.

4. Normalization. The fused volume is obtained via:

$$\mathbf{O}_{\text{fused}} = \frac{\mathbf{O}}{\mathbf{C} + \epsilon}, \quad \epsilon = 10^{-8},$$

producing a seamless $[1, C, \text{out\_D}, \text{out\_H}, \text{out\_W}]$ feature volume for subsequent global pooling and classification.

Smooth transitions across overlaps are ensured, outperforming simple averaging by prioritizing central features.

3.2.5 Loss Functions

We employ a diverse set of loss functions to address class imbalance, hard example mining, label noise, and embedding discriminability in our organoid classification pipeline. Each loss is designed to complement the others, with hyperparameter tuning performed via validation cross-entropy.

3.2.5.1 Weighted Cross-Entropy Loss

To compensate for severe class imbalance (Class 1: 5.35%), we use inverse-frequency weighting:

$$\text{WCE} = -\sum_{c=1}^{C} w_c y_c \log(\hat{y}_c), \quad w_c = \frac{1}{n_c + \epsilon} \tag{3.1}$$

where w_c is inversely proportional to class frequency n_c, normalized across $C = 3$ classes ($\epsilon = 1e - 2$).

3.2.5.2 Focal Loss

Focal Loss dynamically down-weights well-classified examples while focusing on hard negatives:

$$\text{FL}(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t) \tag{3.2}$$

where $p_t = \exp(-\text{CE})$ is the predicted probability for the ground truth class, $\gamma = 2.0$ controls focus, and α_t are class-specific weights.[web:51]

3.2.5.3 Label Smoothing Loss

Label smoothing regularizes against overconfident predictions:

$$\text{LS} = \sum_{c} q_c \log(\hat{p}_c), \quad q_c = \begin{cases} 1 - \epsilon & c = y \\ \frac{\epsilon}{K-1} & c \neq y \end{cases} \tag{3.3}$$

with smoothing $\epsilon = 0.1$ and $K = 3$ classes, optionally weighted by w_c.

3.2.5.4 Diversity Loss

Encourages batch-level prediction diversity via negative entropy penalty:

$$\mathcal{L}_{\text{div}} = -\sum_{c} \bar{p}_c \log \bar{p}_c, \quad \bar{p}_c = \frac{1}{B} \sum_{b} p_{b,c} \tag{3.4}$$

Added to base loss with weight $\lambda_{\text{div}} = 0.1$ to prevent mode collapse.

3.2.5.5 Center Loss

Pulls features toward learnable class centers in embedding space:

$$\mathcal{L}_{\text{center}} = \frac{1}{B} \sum_{b} \| f_b - c_{y_b} \|_2^2 \tag{3.5}$$

with $\lambda_c = 0.5$, where f_b are backbone features and c_y class centers.

3.2.5.6 Similarity Margin Loss

Contrastive loss on pairwise cosine similarities $s_{ij} \in [-1, 1]$:

$$\mathcal{L}_{\text{margin}} = w_{\text{pos}} \cdot \frac{1}{N_{\text{pos}}} \sum_{(i,j)_+} \max(0, m_+ - s_{ij})^2 + w_{\text{neg}} \cdot \frac{1}{N_{\text{neg}}} \sum_{(i,j)_-} \max(0, s_{ij} - m_-)^2 \tag{3.6}$$

with positive margin $m_+ = 0.5$, negative $m_- = 0.0$, and $\sqrt{N_{\text{neg}}/N_{\text{pos}}}$ balancing.

3.2.5.7 Supervised Contrastive Loss

InfoNCE-style loss from similarity matrix:

$$\mathcal{L}_{\text{SupCon}} = -\frac{1}{B} \sum_i \frac{1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(s_{ip}/\tau)}{\sum_{a \neq i} \exp(s_{ia}/\tau)} \tag{3.7}$$

with temperature $\tau = 0.07$, excluding self-similarity.

3.3 Training Setup

3.3.0.1 Hardware and Software Specifications

All experiments were run on the Grid'5000 Sophia site, on machines belonging to the Morpheme team at Inria. In particular, most runs used a ProLiant DL385 Gen10 Plus v2 node equipped with two AMD EPYC 7313 16-core CPUs (64 hardware threads in total), 256 GiB RAM, and 64-bit x86_64 architecture. GPU acceleration was provided by an NVIDIA A40 with 46 GiB of VRAM (driver 535.183.06, CUDA 12.2), with data loading and preprocessing on the CPUs and network training and inference on the GPU.

The software stack consisted of Python 3.9.2, PyTorch 2.0 (with CUDA back-end), MONAI for 3D medical image handling, and standard scientific Python libraries (NumPy, SciPy, scikit-learn, Matplotlib). Napari was used for interactive volume visualisation. All experiments were executed on a 64-bit Linux distribution, with random seeds fixed to ensure reproducibility.

3.3.0.2 Optimization Strategy

Model parameters were optimized using stochastic gradient–based methods. Depending on the experiment, we employed either Adam, AdamW, or momentum SGD as implemented in PyTorch, with an initial learning rate η_0 and L_2 weight decay λ applied to all trainable parameters. Adam and AdamW were used for most convolutional and transformer backbones to stabilise training, while SGD with Nesterov momentum was tested in ablation experiments to assess the impact of optimizer choice on convergence speed and final accuracy.

To control the learning rate over training, we combined cosine–based schedules with optional warmup. In the simplest setting we used cosine annealing, gradually

decaying the learning rate from η_0 to a minimum value $\eta_{\min}$ over $E_{\max}$ epochs. For longer runs, we adopted cosine annealing with warm restarts to periodically reset the learning rate and escape shallow minima. In some configurations, a short linear warmup phase was applied before the cosine schedule, which improved stability when training deep transformer models and reduced sensitivity to the initial learning rate.

3.3.0.3 Training Hyperparameters

The training setup was controlled through a set of global hyperparameters, which were first tuned manually and then refined using 5-fold cross-validation ($k = 5$). Model optimisation relied on mini-batch training with a configurable batch size and stochastic gradient–based optimizers (Adam, AdamW, or SGD with momentum), each coupled with a learning-rate schedule and optional warmup phase. Early stopping was enabled in all experiments, with separate patience and minimum-improvement thresholds for training and validation loss to prevent overfitting while avoiding premature termination.

Regularisation was implemented through L_2 weight decay and, when specified, additional similarity losses weighted by a dedicated coefficient. The learning dynamics were further controlled by the maximum number of epochs, the frequency of validation (`val_every`), and the choice of scheduler type (e.g. cosine annealing with or without warmup or restarts). Sliding-window inference used a configurable sub-batch size (`sw_batch_size`) to respect GPU memory limits. Finally, spatial hyperparameters defined the 3D region of interest, with `roi_x`, `roi_y`, and `roi_z` specifying patch dimensions.

3.4 Evaluation Metrics

3.4.1 Classification Metrics

To quantitatively assess organoid phenotype classification, we compute standard multi–class metrics from the confusion matrix $\mathbf{C} \in \mathbb{R}^{K \times K}$, where C_{ij} counts samples with true class i predicted as class j. For each class $k \in \{1, \ldots, K\}$ we define:

$$\mathrm{TP}_k = C_{kk}, \tag{3.8}$$

$$\mathrm{FP}_k = \sum_{i=1}^{K} C_{ik} - C_{kk}, \tag{3.9}$$

$$\mathrm{FN}_k = \sum_{j=1}^{K} C_{kj} - C_{kk}, \tag{3.10}$$

$$\mathrm{TN}_k = \sum_{i=1}^{K} \sum_{j=1}^{K} C_{ij} - (\mathrm{TP}_k + \mathrm{FP}_k + \mathrm{FN}_k). \tag{3.11}$$

3.4.1.1 Overall Accuracy

Overall accuracy measures the fraction of correctly classified samples over the entire dataset and provides a single global summary of performance:

$$\text{Accuracy} = \frac{\sum_{k=1}^{K} \text{TP}_k}{\sum_{i=1}^{K} \sum_{j=1}^{K} C_{ij}}. \tag{3.12}$$

3.4.1.2 Per–Class Metrics

For each class k we report precision, recall (sensitivity), specificity and F1–score, which quantify complementary aspects of how well each phenotype is recognised:

$$\text{Precision}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k}, \tag{3.13}$$

$$\text{Recall}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k}, \tag{3.14}$$

$$\text{Specificity}_k = \frac{\text{TN}_k}{\text{TN}_k + \text{FP}_k}, \tag{3.15}$$

$$\text{F1}_k = \frac{2 \cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k}. \tag{3.16}$$

These per–class scores are particularly important under class imbalance, as they expose phenotypes that are consistently over– or under–detected.

3.4.1.3 Macro, Weighted and Micro Averages

Let $s_k = \sum_{j=1}^{K} C_{kj}$ be the support (number of true samples) of class k, and $\alpha_k = s_k / \sum_{i=1}^{K} s_i$ the corresponding weight.

3.4.1.3.1 Macro average. Macro averages treat all classes equally, regardless of how frequent they are, and therefore emphasise performance on minority phenotypes:

$$\text{Macro-F1} = \frac{1}{K} \sum_{k=1}^{K} \text{F1}_k \tag{3.17}$$

$$\text{Macro-Precision} = \frac{1}{K} \sum_{k=1}^{K} \text{Precision}_k \tag{3.18}$$

$$\text{Macro-Recall} = \frac{1}{K} \sum_{k=1}^{K} \text{Recall}_k \tag{3.19}$$

3.4.1.3.2 Weighted average. Weighted averages account for the empirical class distribution by weighting each class contribution by its support, providing a summary closer to overall accuracy while still decomposing by metric:

$$\text{Weighted-F1} = \sum_{k=1}^{K} \alpha_k \, \text{F1}_k \tag{3.20}$$

$$\text{Weighted-Precision} = \sum_{k=1}^{K} \alpha_k \, \text{Precision}_k \tag{3.21}$$

$$\text{Weighted-Recall} = \sum_{k=1}^{K} \alpha_k \, \text{Recall}_k \tag{3.22}$$

3.4.1.3.3 Micro average. Micro averages are obtained by summing true and false positives/negatives across classes before computing the metrics; they correspond to evaluating precision, recall and F1 on the pooled multi–class problem:

$$\text{TP}_{\text{micro}} = \sum_{k=1}^{K} \text{TP}_k, \quad \text{FP}_{\text{micro}} = \sum_{k=1}^{K} \text{FP}_k, \quad \text{FN}_{\text{micro}} = \sum_{k=1}^{K} \text{FN}_k, \tag{3.23}$$

$$\text{Precision}_{\text{micro}} = \frac{\text{TP}_{\text{micro}}}{\text{TP}_{\text{micro}} + \text{FP}_{\text{micro}}} \tag{3.24}$$

$$\text{Recall}_{\text{micro}} = \frac{\text{TP}_{\text{micro}}}{\text{TP}_{\text{micro}} + \text{FN}_{\text{micro}}} \tag{3.25}$$

$$\text{F1}_{\text{micro}} = \frac{2 \cdot \text{Precision}_{\text{micro}} \cdot \text{Recall}_{\text{micro}}}{\text{Precision}_{\text{micro}} + \text{Recall}_{\text{micro}}} \tag{3.26}$$

Using these complementary summaries allows a nuanced evaluation of the classifiers, balancing global behaviour with class–specific performance under the inherent class imbalance of the organoid dataset.

3.4.2 Computational Efficiency

3.4.3 Uncertainty Quantification via MC-Dropout

Monte Carlo Dropout approximates the Bayesian posterior predictive distribution by performing $T = 50$ forward passes with dropout active at inference time. From the stacked predictions $\mathbf{P} \in \mathbb{R}^{T \times C}$ (where $C = 3$ classes), we compute five complementary uncertainty metrics, each isolating distinct failure modes:

3.4.3.1 Predictive Entropy

Formula:

$$H[\bar{\mathbf{p}}] = -\sum_{c=1}^{C} \bar{p}_c \log \bar{p}_c, \quad \text{where} \quad \bar{\mathbf{p}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{p}_t \tag{3.27}$$

Measures the total predictive uncertainty of the ensemble mean distribution. High entropy indicates that probability mass is spread across classes, suggesting fundamental model confusion. This metric is robust to detecting both data ambiguity and model ignorance, making it most suitable for selective prediction thresholds in organoid classification.

3.4.3.2 Expected Entropy (Aleatoric Uncertainty)

Formula:

$$H_{\text{aleatoric}} = \mathbb{E}_t[H[\mathbf{p}_t]] = \frac{1}{T}\sum_{t=1}^{T}\left(-\sum_{c=1}^{C} p_{t,c}\log p_{t,c}\right) \tag{3.28}$$

Averages the entropy across all T MC-Dropout samples. High expected entropy reflects intrinsic data ambiguity (e.g., imaging artifacts, morphologically borderline organoids) rather than model parameter uncertainty. Samples with high $H_{\text{aleatoric}}$ but low BALD indicate fundamentally difficult-to-classify instances that would benefit from higher-quality imaging or manual re-inspection rather than model retraining.

3.4.3.3 BALD (Bayesian Active Learning by Disagreement)

Formula:

$$\text{BALD}(x) = H[\bar{\mathbf{p}}] - H_{\text{aleatoric}} = I[y;\theta \mid x, D] \tag{3.29}$$

Quantifies epistemic (model) uncertainty via mutual information between the label and network weights conditioned on the input and training data. High BALD indicates that different MC-Dropout samples make confident but contradictory predictions—a signature of model parameter uncertainty in regions of feature space not well-explored during training. These samples are maximally informative for active learning and represent true "borderline" organoids where additional labelled examples would most improve the model.

3.4.3.4 Predictive Variance

Formula:

$$\sigma_c^2 = \mathbb{E}_t[p_{t,c}^2] - \bar{p}_c^2 = \frac{1}{T}\sum_{t=1}^{T} p_{t,c}^2 - \bar{p}_c^2, \quad \text{Mean Variance} = \frac{1}{C}\sum_{c}\sigma_c^2 \tag{3.30}$$

Directly quantifies how much predicted class probabilities fluctuate across MC-Dropout samples. High variance indicates prediction instability, which often correlates with out-of-distribution or rare organoid morphologies. Unlike entropy-based metrics, variance captures prediction spread even when the mean distribution remains sharp, making it sensitive to systematic disagreement among models.

3.4.3.5 Variation Ratio

Formula:

$$\text{VR}(x) = 1 - \frac{1}{T}\sum_{t=1}^{T}\mathbb{I}(\hat{y}_t = \hat{y}_{\text{mode}}) \tag{3.31}$$

Measures the fraction of MC-Dropout samples that disagree with the most-frequent predicted class. VR ranges from 0 (unanimous) to 1 (maximum disagreement). This discrete metric is computationally efficient and highly interpretable: VR = 0.3 means 30% of models "flip" to a different class than the consensus, signalling

high epistemic uncertainty. It is particularly useful for flagging class-flip instabilities that might indicate misclassification risk.

Chapter 4

Results

In this chapter, we present the results obtained from the conducted experiments, organized by thematic sections. For each section, we provide summary tables of performance metrics, as well as comparative charts and tables. Detailed results, including confusion matrices and complete metrics for each model and configuration, are available in Appendix A (see Chapter A). In multi-class classification of imbalanced organoid phenotypes, macro-averaged F1-score, precision, and recall were prioritized for model comparison as they treat all classes equally, emphasizing performance on rare minority classes like *Compact* (support=5/82) critical in medical imaging analysis. Weighted F1 provides a prevalence-adjusted summary closer to overall accuracy , while other metrics, as specificity, were deprioritized since they can be misleading in imbalanced contexts, where a model could achieve high specificity by simply predicting the majority class.

4.1 Overall Classification Performance

In this section, we compare the models tested in the baseline configuration (full-resolution volumes 512×512×128) and in reduced-resolution configurations using the Spatial Abstractor. This initial experimentation phase employs non-overlapping 128×128×128 patches with a standard classification head. All models leverage pre-trained weights from the datasets described earlier. For each model, we report the best results achieved in terms of accuracy, F1-score, precision, and recall under this specific configuration.

4.1.1 Baseline: Full-resolution volumes (512×512×128)

Table 4.1: Models Comparison: Full-Resolution

Model	Accuracy	Macro Prec.	Macro F1	Macro Rec.	W-F1
DenseNet	0.537	0.504	0.291	0.369	0.419
ResNet18	0.610	0.512	0.396	0.442	0.554
ResNet50	0.549	0.378	0.364	0.384	0.516
SwinUNETR	0.927	0.831	0.831	0.831	0.927
SwinVit	0.793	0.554	0.553	0.565	0.780

Among all models, **SwinUNETR** is the top performer, reaching an accuracy of 0.93, followed by **SwinViT** with 0.79. Convolutional networks, instead, lag behind at this resolution and patch configuration, failing to fully exploit the rich spatial information. SwinUNETR's main strength is its ability to correctly classify even the rare *Compact* class, as indicated by its per-class precision (0.600) and recall metrics—something no other model achieves at this stage. In contrast, CNNs largely collapse onto the majority Cauliflower and Cystic categories, struggling to separate the three phenotypes. SwinViT shares the same failure mode on Compact but provides better discrimination of the majority classes than CNNs. This behaviour is reflected in the macro vs. weighted metrics: SwinUNETR maintains a relatively small gap between Macro and Weighted F1 (0.831 vs. 0.927), whereas SwinViT shows a much lower Macro F1 compared to its Weighted F1 (0.553 vs. 0.780), revealing that good performance on prevalent classes is offset by poor handling of the minority phenotype.

4.1.2 Spatial Abstractor: 256×256×128 Volumes

Table 4.2: Models Comparison: 256x256x128

Model	Accuracy	Macro Prec.	Macro F1	Macro Rec.	W-F1
DenseNet	0.890	0.808	0.827	0.863	0.893
ResNet18	0.878	0.820	0.835	0.856	0.879
ResNet50	0.756	0.640	0.620	0.657	0.770
SwinUNETR	0.902	0.879	0.873	0.874	0.903
SwinVit	0.878	0.813	0.765	0.740	0.874

Dimensionality reduction markedly benefits convolutional models: **DenseNet** and **ResNet18** gain substantial accuracy and move much closer to the transformer baselines. While they still trail **SwinUNETR** overall (Accuracy 0.902), they now discriminate the three classes more effectively, particularly improving *Compact* classification—a persistent weakness at full resolution. **ResNet50** exhibits only marginal gains and remains comparatively imprecise (Accuracy 0.756), likely reflecting an unfavourable parameter–data ratio for this dataset. SwinUNETR retains top

performance through strong Compact precision/recall (around 0.800), but its errors shift toward the majority classes, frequently confusing Cauliflower and Cystic due to their morphological similarity. **SwinViT** performs well on prevalent classes yet continues to struggle with the rare phenotype, with Compact recall dropping to 0.400.

4.1.3 Initialization strategies: Scratch vs Pre-trained weights

Table 4.3: Models Comparison: Scratch vs Pretrained (256x256x128)

Model	Accuracy	Macro Prec.	Macro F1	Macro Rec.	W-F1
ResNet18	0.878	0.820	0.835	0.856	0.879
ResNet18 (S.)	0.829	0.740	0.665	0.649	0.820
SwinUNETR	0.902	0.879	0.873	0.874	0.903
SwinUNETR (S.)	0.902	0.832	0.782	0.758	0.898
SwinVit	0.878	0.813	0.765	0.740	0.874
SwinVit (S.)	0.829	0.556	0.570	0.590	0.804

Using the same 256×256×128 volumes, 128×128×128 patches, and a standard classification head, this experiment contrasts models trained from scratch with their pre-trained counterparts. Training from scratch consistently degrades performance, confirming the importance of transfer learning in this low-data regime. **SwinUNETR** preserves high overall precision but suffers a pronounced drop in Compact recall (0.800 $\rightarrow$ 0.400), revealing a strong dependence on pre-trained features for detecting rare phenotypes. **ResNet18** increasingly confuses Cauliflower with Cystic, and **SwinViT** fails to reliably recognise Compact samples at all—highlighting how challenging this class remains even for high-capacity models. **DenseNet** and **ResNet50** were excluded: ResNet50 did not converge under scratch training, and DenseNet was never trained from pre-initialised weights, preventing a fair comparison in this setting.

4.1.4 Spatial Abstractor: 128×128×128 volumes

Table 4.4: Models Comparison: 128x128x128

Model	Accuracy	Macro Prec.	Macro F1	Macro Rec.	W-F1
DenseNet	0.890	0.785	0.792	0.807	0.893
ResNet18	0.939	0.909	0.930	0.957	0.939
SwinUNETR	0.878	0.872	0.887	0.915	0.878
SwinVit	0.841	0.787	0.821	0.889	0.842

At this lower resolution, convolutional models gain further traction: **ResNet18** reaches 0.94 accuracy and **DenseNet** approaches 0.89, indicating that aggressive downsampling effectively mitigates overfitting and improves generalization on the

limited dataset. The issue with the underrepresented *Compact* class almost disappears, as **SwinViT**, **SwinUNETR**, and **ResNet18** all achieve 100% Compact recall. However, attention-based models pay for this with reduced precision on Cauliflower and Cystic, reflecting increased confusion between the majority phenotypes at this coarser scale.

4.1.5 Computational efficiency

Table 4.5: Inference Time Comparison - in milliseconds (ms) per sample

Model	DenseNet	ResNet18	ResNet50	SwinUNETR	SwinVit
512x512x128	170	131	274	1704	1714
256x256x128	47	35	71	429	427
256x256x128 SA	720	706	740	1107	1099
128x128x128	23	10	*	119	116
128x128x128 SA	590	572	*	679	674

From a computational perspective, lightweight CNNs such as **ResNet18** and **DenseNet** achieve the shortest inference times and smallest memory footprints, making them attractive for resource-constrained settings. However, at high resolutions they struggle to capture the full spatial complexity of organoids, and only reach competitive accuracy once the input is aggressively downsampled. This downsampling, implemented via the Spatial Abstractor, boosts CNN performance but also becomes the dominant computational bottleneck: most of the runtime is spent in interpolation and patch generation rather than in the networks themselves, adding roughly 500 ms of preprocessing latency depending on the configuration. In contrast, attention-based models maintain relatively stable inference times across resolutions, as they can operate directly on higher-resolution volumes with limited downsampling while still extracting discriminative features. Despite their inherent attention overhead, they ultimately offer a more favourable speed–performance trade-off at full or moderately reduced resolutions.

4.2 Slice handling strategies

The following experiments compare two slice-handling schemes: *Z-thinning* and a more aggressive *XYZ-thinning*, both based on selective downsampling of the volume. In **Z-thinning**, the Slice Selector (SS) is first applied along the Z-axis to discard redundant slices; the Spatial Abstractor (SA) then downsamples each remaining slice in the XY plane. In **XYZ-thinning**, SS is instead applied uniformly across all three axes, jointly reducing resolution in X, Y, and Z. The aim is to assess whether such spatial dimensionality reduction improves model performance—especially for convolutional architectures—and to what extent more aggressive thinning trades accuracy for efficiency. This design is motivated by the observation that many slices

are nearly empty or redundant; by retaining only the most informative subset, we seek to lower computational cost while preserving, or even enhancing, generalization on the organoid classification task.

4.2.1 Z-thinning: 128×128×128 → 128×128×N

Table 4.6: Models Comparison: with different numbers of Z-layers

Model	Acc.	M-Prec.	M-F1	M-Rec.	W-F1	Inf-Time(ms)
DenseNet-128	0.890	0.808	0.827	0.863	0.893	590
DenseNet-64	0.890	0.822	0.773	0.749	0.885	305
DenseNet-32	0.890	0.827	0.844	0.864	0.891	168
ResNet18-128	0.878	0.820	0.835	0.856	0.879	571
ResNet18-64	0.915	0.886	0.882	0.883	0.915	291
ResNet18-32	0.915	0.947	0.939	0.942	0.914	150
SwinUNETR-128	0.902	0.879	0.873	0.874	0.903	678
SwinUNETR-64	0.866	0.811	0.826	0.847	0.867	348
SwinUNETR-32	0.854	0.818	0.848	0.897	0.854	181
SwinVit-128	0.878	0.813	0.765	0.740	0.874	674
SwinVit-64	0.841	0.804	0.809	0.832	0.842	348
SwinVit-32	0.805	0.769	0.781	0.802	0.804	180

Progressively reducing the Z dimension from 128 to 64 and then 32 slices leads to clear gains for CNN-based models: **ResNet18** reaches 0.92 accuracy and **DenseNet** remains stable around 0.89, indicating that Z-downsampling effectively mitigates overfitting and improves generalization in this data-limited setting.

In contrast, attention-based models suffer notable precision drops on Cauliflower and Cystic, even though they maintain 100% recall on Compact; Z-thinning thus reduces overfitting but increases confusion among majority classes, for which fine-grained 3D spatial relationships are more critical. Additionally, shrinking the Z dimension incurs extra preprocessing cost, as more aggressive downsampling along this axis becomes the dominant contributor to inference time.

4.2.2 XYZ-thinning vs Z-thinning (32×32×32)

Table 4.7: Models Comparison: XYZ-thinning vs Z-thinning (32x32x32)

Model	Thin.	Acc.	M-Prec.	M-F1	M-Rec.	W-F1	Inf-Time(ms)
DenseNet	Z	0.805	0.547	0.557	0.572	0.786	350
DenseNet	XYZ	0.756	0.507	0.516	0.534	0.728	22
ResNet18	Z	0.878	0.842	0.816	0.798	0.877	329
ResNet18	XYZ	0.902	0.857	0.883	0.933	0.903	3
SwinVit	Z	0.732	0.725	0.691	0.698	0.727	341
SwinVit	XYZ	0.866	0.827	0.856	0.904	0.866	13

This experiment compares two downsampling schemes at an extremely low target resolution of 32×32×32: **Z-thinning**, which applies SS along Z followed by SA in the XY planes, and **XYZ-thinning**, which uses SS jointly on all three axes. The goal is to assess whether such aggressive compression still preserves the information needed for reliable classification—especially for CNNs, which can benefit from more compact inputs.

Results are partly model-dependent. XYZ-thinning further improves performance for **ResNet18** (Accuracy 0.878 → 0.902) and **SwinViT** (0.732 → 0.866), indicating that a more uniform reduction of redundant slices can sharpen the signal for these architectures. For **DenseNet**, however, the trend reverses (0.805 → 0.756),due to a scarse performance on *Compact* class (Recall: 0.000),suggesting a stronger sensitivity to the loss of fine-grained context. **SwinUNETR** cannot be evaluated in this setting, as 32^3 patches would require substantial architectural changes to its encoder–decoder design. Compared with previous XY-resize–then–Z-slice approaches, XYZ-thinning is also significantly faster (300 ms savings per sample), simplifying both preprocessing and inference while, in several cases, preserving or even enhancing accuracy.

4.2.3 Slice Selector (SS) vs Spatial Abstractor (SA) (128x128x128)

Table 4.8: Models Comparison: 128x128x128

Model	Method	Acc.	M-Prec.	M-F1	M-Rec.	W-F1	Inf-Time(ms)
DenseNet	SA	0.890	0.785	0.792	0.807	0.893	678
DenseNet	SS	0.878	0.812	0.764	0.738	0.873	23
ResNet18	SA	0.939	0.909	0.930	0.957	0.939	678
ResNet18	SS	0.890	0.773	0.707	0.691	0.881	10
SwinUNETR	SA	0.878	0.872	0.887	0.915	0.878	1153
SwinUNETR	SS	0.829	0.704	0.703	0.703	0.828	118
SwinVit	SA	0.841	0.787	0.821	0.889	0.842	1153
SwinVit	SS	0.841	0.739	0.724	0.713	0.839	117

In this experiment, we compare the Slice Selector (SS) and Spatial Abstractor (SA) strategies for processing 128×128×128 volumes. Although SS is substantially faster than SA, it suffers a notable drop in accuracy—particularly for majority classes—due to the loss of critical spatial details. For DenseNet and SwinViT with SS, performance improves on the Cauliflower class (recall: 0.950 vs. SA's 0.875) but declines on Compact (0.400 vs. 0.600) and Cystic (0.865 vs. 0.950). ResNet18 performs adequately with SS overall but experiences a sharp decline on Compact, with a very low recall of 0.200 compared to SA's perfect score. SwinUNETR with SS not only degrades on Compact but also struggles significantly to distinguish Cystic from Cauliflower, dropping to a recall of 0.750 versus SA's 0.946.

4.2.4 Different patch sizes evaluation

Table 4.9: Models Comparison: Patch Size 64×64×64 vs 128x128x128

Model	Acc.	M-Prec.	M-F1	M-Rec.	W-F1	Inf-Time(ms)
DenseNet	0.890	0.808	0.827	0.863	0.893	720
DenseNet-64	0.780	0.528	0.531	0.550	0.750	752
ResNet18	0.878	0.820	0.835	0.856	0.879	706
ResNet18-64	0.780	0.522	0.537	0.555	0.756	708
SwinUNETR	0.902	0.879	0.873	0.874	0.903	1107
SwinUNETR-64	0.902	0.857	0.883	0.933	0.903	1153
SwinVit	0.878	0.813	0.765	0.740	0.874	1099
SwinVit-64	0.866	0.782	0.913	0.906	0.874	1147

Experiments were run on 256×256×128 volumes using either 64×64×64 or 128×128×128 patches with a standard classification head. CNNs degrade markedly at smaller patch sizes: **DenseNet** shows a sharp macro-precision drop (0.808 → 0.526), especially on Compact and Cystic, and **ResNet** models exhibit a similar pattern (0.820 → 0.522).

In contrast, **SwinUNETR** and **SwinViT** maintain 100% recall on Compact, at the price of modest precision losses and increased Cauliflower–Cystic confusion. From a computational standpoint, using 64x64x64 patches does not yield meaningful speedups, so the loss in performance is not compensated by clear efficiency gains.

4.2.5 Patch merging strategies

Table 4.10: Models Comparison: Patch Merging vs Non-overlapping Patches

Model	Acc.	M-Prec.	M-F1	M-Rec.	W-F1	Inf-Time(ms)
DenseNet	0.890	0.808	0.827	0.863	0.893	720
DenseNet PM	0.451	0.150	0.207	0.333	0.281	766
ResNet18	0.878	0.820	0.835	0.856	0.879	706
ResNet18 PM	0.732	0.681	0.596	0.581	0.723	750
ResNet50	0.756	0.640	0.620	0.657	0.770	740
ResNet50 PM	0.756	0.640	0.620	0.657	0.770	827
SwinUNETR	0.902	0.879	0.873	0.874	0.903	1107
SwinUNETR PM	0.866	0.824	0.856	0.904	0.866	1633

This experiment evaluates patch merging on 256×256×128 volumes with 128×128×128 patches and a standard classification head, comparing conventional non-overlapping patches to a scheme that fuses adjacent patches with a stride of 2/3 the patch size. **DenseNet** undergoes a drastic collapse (Accuracy 0.890 → 0.451), effectively degenerating to near single-class predictions. **ResNet** models fare slightly better but still remain weak across all classes (Weighted F1 0.879 → 0.723). **SwinUNETR** is more resilient, preserving strong overall metrics and 100% Compact recall, but still exhibits a noticeable drop and longer inference times due to the extra fusion operations. From a computational perspective, patch merging significantly increases the number of patches and fusion computations, so the resulting inference times are 10–50% longer than the non-overlapping baseline, without any clear accuracy benefits. Overall, patch merging introduces boundary artifacts and redundant computation without measurable benefits for classification, making non-overlapping patches the preferred strategy in this setting.

4.3 Classification head comparison

4.3.1 Vanilla Classification Head vs NOAH Classification Head

Table 4.11: SwinUNETR Comparison: Vanilla H. vs Noah H.

Model	Acc.	M-Prec.	M-F1	M-Rec.	W-F1	Inf-Time(ms)
Vanilla (512)	0.927	0.831	0.831	0.831	0.927	1704
NOAH (512)	0.854	0.795	0.747	0.721	0.849	1707
Vanilla (256)	0.902	0.879	0.873	0.874	0.903	1106
NOAH (256)	0.878	0.872	0.887	0.915	0.878	1112
Vanilla (128)	0.878	0.872	0.887	0.915	0.878	678
NOAH (128)	0.793	0.639	0.924	0.924	0.787	679

We compare the standard linear classification head against the more expressive NOAH head at 256×256×128 resolution with 128×128×128 patches, using SwinUNETR as backbone. NOAH yields slightly lower precision on Cauliflower and Cystic compared to the standard head, while matching Compact performance and offering marginally higher Compact recall. At this resolution, both heads perform similarly overall, with NOAH even providing a small boost in macro F1 despite minor declines on the two non-Compact classes. When further downscaling the input to 128×128×128 while keeping NOAH, accuracy decreases for both Compact and Cauliflower, though Cystic shows a slight improvement relative to the standard head. These results suggest that, in this setting, NOAH behaves as a largely drop-in alternative to a linear head—consistent with prior reports that richer heads can give modest gains over strong backbones—without fundamentally altering the backbone's behavior.

4.4 Uncertainty Quantification Results

For each experiment, the best configuration was selected for every model, optimizing both hyperparameters and resolution. For convolutional models, we chose 128×128×128 resolution with Spatial Abstractor (SA) sampling. For attention-based models, we selected a performance-feasibility trade-off, maintaining 256×256×128 resolution, with Spatial Abstractor (SA) sampling. The dropout layer was added before the classification head, with a probability of 0.1, and was active during both training and inference to enable uncertainty estimation through Monte Carlo Dropout (MC-Dropout) analysis. The number of stochastic forward passes for MC-Dropout was set to 50, providing a robust estimate of predictive uncertainty while balancing computational cost.

4.4.1 Metrics Analysis

Table 4.12: Models Comparison: Montecarlo Dropout

Model	Accuracy	Macro Prec.	Macro F1	Macro Rec.	W-F1
DenseNet	0.927	0.950	0.948	0.949	0.927
ResNet18	0.902	0.884	0.904	0.931	0.903
SwinUNETR	0.902	0.879	0.873	0.874	0.903
SwinVit	0.854	0.801	0.748	0.723	0.850

Introducing dropout before the classification head impacts model performance as follows, based solely on standard metrics (without uncertainty analysis). DenseNet shows substantial improvement on the Compact class, boosting recall from 0.600 to a perfect 1.000, while maintaining unchanged performance on the other classes. ResNet18 experiences a minor dip on majority classes (just 3 misclassified samples), remaining perfect on Compact. SwinViT continues struggling with Compact and slightly declines on Cauliflower (2 additional errors). SwinUNETR maintains identical

performance across all configurations, demonstrating robustness.

4.4.2 Correlation between uncertainty and model errors

Model	Correct predictions	Incorrect predictions	Uncertainty gap
DenseNet	0.3503 ± 0.2114	0.5871 ± 0.1720	0.2368
ResNet18	0.1872 ± 0.2323	0.5356 ± 0.2046	0.3484
SwinVit	0.1701 ± 0.2064	0.5042 ± 0.2510	0.3341
SwinUNETR	0.2881 ± 0.2409	0.5183 ± 0.1537	0.2302

Table 4.13: MC-Dropout Uncertainty Analysis – Predictive Entropy

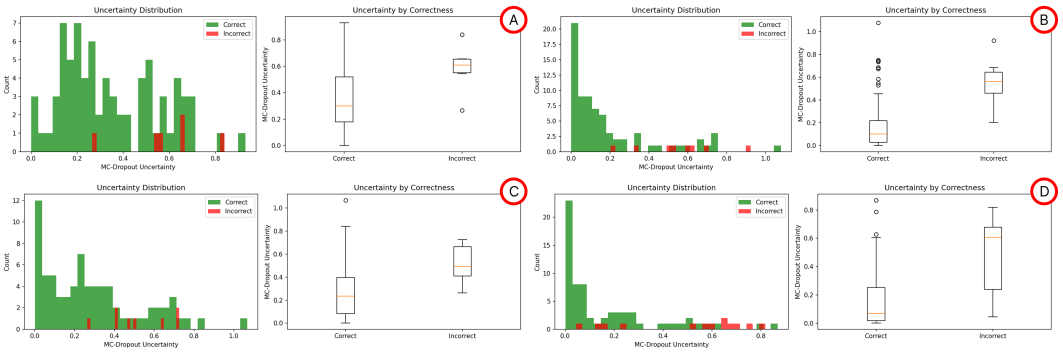

Figure 4.1: Uncertainty Distribution, A)DenseNet, B)ResNet18, C)SwinUNETR, D)SwinViT.

The MC-Dropout analysis characterizes how well each model's predictive entropy separates correct from incorrect decisions: low mean (and variance) on correct predictions reflects reliable confidence, whereas a large gap to the entropy of misclassified samples indicates that uncertainty is informative for detecting errors. DenseNet and SwinUNETR show moderate entropy for correct predictions (0.3503 ± 0.2114 and 0.2881 ± 0.2409) together with sizeable uncertainty gaps (0.2368 and 0.2302), pointing to well-calibrated uncertainty estimates. ResNet18 and SwinViT exhibit even lower entropy on correct predictions (0.1872 ± 0.2323 and 0.1701 ± 0.2064) and the largest gaps (0.3484 and 0.3341), indicating very confident correct decisions but also comparatively high entropy on errors, which may help in flagging uncertain, potentially wrong predictions.

4.4.3 Selective Prediction – Rejection Thresholds Accuracy

Selective Prediction – Rejection Thresholds Accuracy				
Model	**Top 50%**	**Top 25%**	**Top 10%**	**Top 5%**
DenseNet	0.9756	0.9672	0.9315	0.9351
ResNet18	1.0000	0.9836	0.9178	0.9091
SwinVit	0.9756	0.9344	0.9041	0.8701
SwinUNETR	1.0000	0.9508	0.9178	0.9091

Table 4.14: Accuracy on kept samples after rejecting highest uncertainty predictions (top X%).

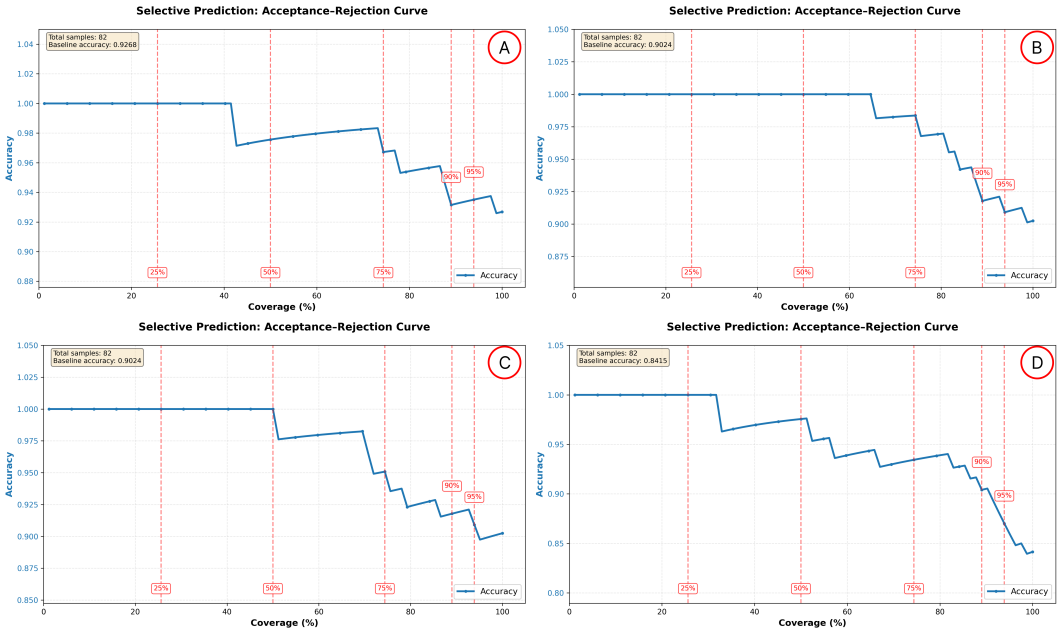

Figure 4.2: Acceptance Curve, A)DenseNet, B)ResNet18, C)SwinUNETR, D)SwinViT.

This table evaluates selective prediction: predictions are sorted by descending uncertainty, the top X% most uncertain are rejected, and accuracy is measured on the remaining kept samples. DenseNet consistently achieves the highest accuracy across all thresholds (0.9756–0.9351), with minimal degradation even when rejecting only the top 5%, demonstrating excellent uncertainty-guided filtering. ResNet18 and SwinUNETR maintain strong performance (0.905), with SwinUNETR reaching perfect accuracy (1.000) at the strictest 50% threshold. SwinViT lags behind (0.8608–0.9091), indicating less effective uncertainty estimates for identifying reliable predictions.

4.5 Cross-Validation Analysis

For each experiment, the best configuration was selected for every model using k-fold validation, optimizing both hyperparameters and resolution. For convolutional models, we chose 128×128×128 resolution with Spatial Abstractor (SA) sampling. For attention-based models, we selected a performance-feasibility trade-off, maintaining 256×256×128 resolution, with Spatial Abstractor (SA) sampling.

4.5.1 K-Fold cross-validation results

K-Fold Cross-Validation Results (Mean ± Std)			
Model	**Train Acc**	**Val Acc**	**Train Loss**
DenseNet	0.6422 ± 0.0196	0.9126 ± 0.0084	0.4696 ± 0.0339
ResNet18	0.9605 ± 0.0112	0.9459 ± 0.0173	0.0851 ± 0.0173
SwinUNETR	0.9265 ± 0.0085	0.8988 ± 0.0159	0.1639 ± 0.0099
SwinVit	0.7840 ± 0.1664	0.8113 ± 0.1631	0.4330 ± 0.3308

Table 4.15: , Comparative K-Fold results across models. Metrics shown as mean ± standard deviation.

The K-fold cross-validation reveals distinct performance trends across models. DenseNet exhibits the highest validation accuracy (0.9126 ± 0.0084) with low variance, alongside moderate training accuracy (0.6422 ± 0.0196) and loss (0.4696 ± 0.0339), indicating effective generalization without severe overfitting. ResNet18 and SwinUNETR show high training accuracy (>0.92) but slightly lower validation scores (0.9459 and 0.8988), with very low std devs suggesting stable convergence; ResNet18 achieves the lowest train loss (0.0851 ± 0.0173). SwinViT displays higher variability (e.g., Val Acc std 0.1631), with balanced but lower overall metrics, pointing to potential sensitivity to fold-specific data distributions.

4.5.2 Best Fold Metrics analysis

Table 4.16: , Models Comparison: Best Fold (K-Fold Validation)

Model	**Accuracy**	**Macro Prec.**	**Macro F1**	**Macro Rec.**	**W-F1**
DenseNet	0.878	0.867	0.856	0.858	0.879
ResNet18	0.902	0.879	0.873	0.874	0.903
SwinUNETR	0.878	0.797	0.819	0.856	0.882
SwinVit	0.866	0.763	0.774	0.789	0.868

The best-fold metrics (detailed per-class breakdowns in Appendix) reveal nuanced model strengths, particularly on imbalanced organoid classes. DenseNet achieves balanced performance but suffers low Cauliflower recall due to persistent confusion

with Cystic (high precision preserved), while excelling on the minority Compact class. ResNet18 delivers top-tier overall metrics, tempered only by the same Cauliflower-Cystic overlap as DenseNet (detailed confusion matrix in Appendix). SwinViT shows degradation across macro metrics, driven by low macro-precision on all classes, indicating struggles with positive predictions. SwinUNETR attains the highest Weighted F1 (0.986) thanks to strong majority-class performance, despite macro-precision limitations across classes reflecting imbalance sensitivity.

Chapter 5

Ablation Studies

This chapter analyzes the results obtained in previous sections to address the research questions outlined at the project's outset, presented in Subsection 1.3.3 (Chapter 1). The main objective is to identify the key factors driving the observed performance and evaluate alternative strategies that could potentially yield improved results.

5.1 Overall Classification Performance

5.1.1 Baseline vs. Spatial Abstractor

Table 5.1: Model Accuracy by Resolution

Model	512x512x128	256x256x128	128x128x128	32x32x32
SwinUNETR	0.927 (1st)	0.902 (1st)	0.878 (3rd)	X
ResNet18	0.610 (3rd)	0.878 (4th)	0.939 (1st)	0.902 (1st)
SwinVit	0.793 (2nd)	0.878 (3rd)	0.787 (4th)	0.866 (2nd)
DenseNet	0.537 (5th)	0.890 (2nd)	0.890 (2nd)	0.805 (3rd)
ResNet50	0.549 (4th)	0.756 (5th)	X	X

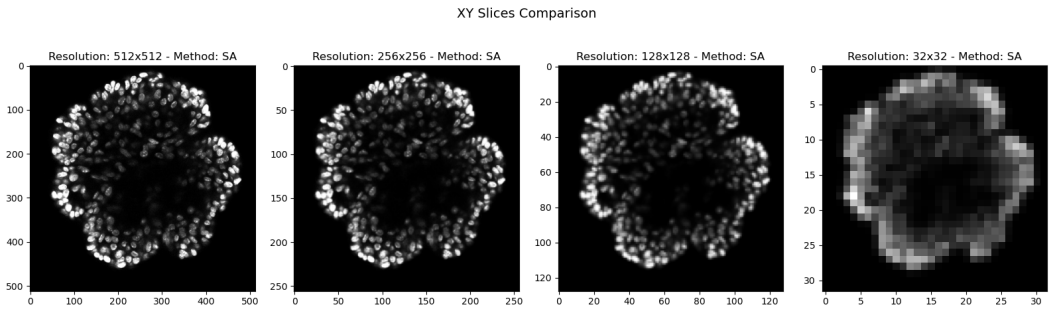

Figure 5.1: Comparative visualization of a sample across different resolutions.

Experiments reveal stark differences in architectural behavior between attention-based and convolutional mechanisms. Attention models (**SwinUNETR**, **SwinViT**)

maintain stable performance across all resolutions despite substantially higher inference latency, thanks to their global receptive field via self-attention mechanisms. Conversely, CNNs (**DenseNet**, **ResNet18**) struggle at high resolutions but excel at lower ones ($128 \times 128 \times 128$), where reduced patch sizes mitigate overfitting and enhance generalization on our limited dataset. This disparity stems from patch-based processing: CNNs require larger receptive fields or full-volume analysis to capture long-range spatial dependencies in organoids, whereas Swin architectures efficiently extract features even from peripheral patches without hierarchical complexity of standard transformers. Experiments further demonstrate that, on this small-scale dataset, the pure attention-based **SwinViT** underperforms compared to the hybrid **SwinUNETR**, which augments attention with convolutional decoder blocks for refined spatial feature fusion. Although downsampling reduces patch count and operations, preprocessing interpolation dominates CNN inference time—ironically slowing these inherently fast models while boosting accuracy. Attention models benefit dually from faster inference and preserved performance. Common failure modes include: (1) missing the minority **Compact** class (often resolved at optimal low-res CNN configs); (2) **Cauliflower-Cystic** confusion due to morphological overlap—Cauliflower's irregular indentations/convexities vs. Cystic's smooth borders become indistinguishable at very low resolutions (e.g., 32^3).

5.1.2 Spatial Abstractor vs Slice Selector

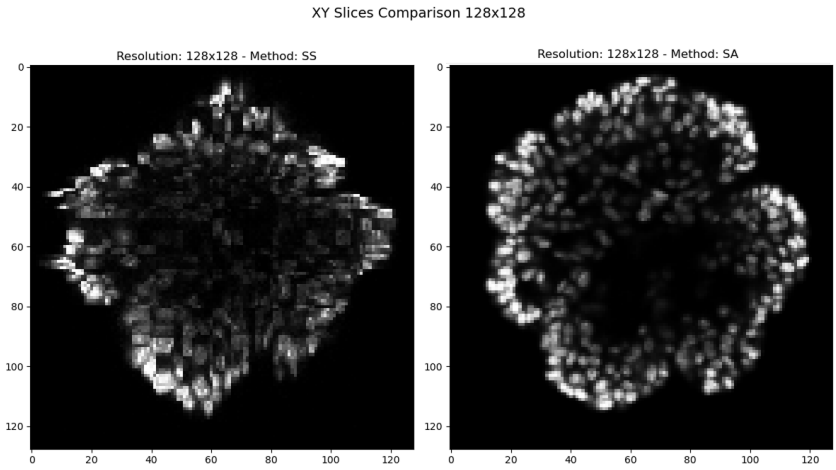

Figure 5.2: Comparative visualization of a sample across different resolution methods.

Given the computational cost of Spatial Abstractor (SA) interpolation—which dominates inference time—a lighter dimensionality reduction alternative was explored: the **Slice Selector (SS)**. SS selects the most dissimilar Z-slices (potentially most discriminative for class separation), yielding dramatically faster preprocessing. However, as results and visualizations confirm, simultaneous XYZ-thinning via SS severely degrades spatial continuity, creating fragmented representations that impair 3D context.

All models suffer overall performance drops at 128×128×128 resolution , despite massive inference speedup (678ms→10-23ms). Hybrid SA (XY) + SS (Z) benefits CNNs dually: improved performance *and* reduced inference time, as SS first cuts Z-slices before SA interpolation, minimizing operations. This efficiency compromises spatial coherence ("flattening" volumes), which CNNs tolerate well—leveraging their local inductive biases—but attention models handle poorly, as global dependencies suffer from lost Z-context. Thus, SS trades essential spatial information for speed without accuracy gains or parity.

5.1.3 Scratch vs Pre-trained Weights

Given the scarcity of medical imaging datasets and our limited organoid samples, we ablated training from scratch versus fine-tuning pre-trained models (originally on MedicalNet and BRATS21). Training from scratch demands substantially higher memory (full backpropagation through all layers), longer convergence (more epochs), and smaller batch sizes—exacerbating instability on small data. Fine-tuning pre-trained weights accelerates convergence, enables larger batches for stable gradients, and leverages transferable priors like edge/texture detection, effectively bridging the domain shift to organoids. So the pre-training is essential for our dataset size, yielding 10% precision gains (e.g., SwinViT: 0.556→0.813) with 2–3× fewer epochs and no Compact class collapse.

5.1.4 Patch Strategies & Classification Head Ablations

Three complementary ablations reveal optimal design choices for patch-based 3D organoid classification.

Patch Merging vs Non-overlapping: Contrary to segmentation workflows, merging adjacent patches (stride 2/3 of the patch size) consistently hurts classification: performance drops by about 10% across models (e.g., DenseNet F1: 0.827→0.207; SwinUNETR: 0.873→0.856), while inference time increases by 10–50% due to the additional fusion operations. These results suggest that patch merging introduces boundary artifacts and noise that disrupt class-discriminative signals; CNNs, already challenged in separating phenotypes, are impacted the most, but even attention models lose accuracy despite their stronger spatial reasoning. Given the simultaneous loss in performance and increase in computational cost, patch merging proves impractical for this classification task, and non-overlapping patches are therefore preferred.

Vanilla vs NOAH Head: NOAH consistently underperforms the simple linear head across resolutions (512: F1 0.831→0.747; 256: 0.902→0.878), confirming a simple architecture already provide rich features. Added complexity harms generalization without compensating benefits, validating minimal head design.

Patch Size: Smaller 64x64x64 patches substantially degrade CNN performance (DenseNet F1: 0.827→0.531; ResNet18: 0.835→0.537), indicating that reduced spatial context limits their ability to capture organoid morphology. In contrast, attention models remain stable (SwinUNETR =0.903) and handle the increased num-

ber of patches more gracefully, but the higher patch count still lengthens inference. Overall, in this setting 64x64x64 patches neither improve accuracy nor efficiency, so 128x128x128 is preferred.

5.2 Uncertainty Quantification

To enable uncertainty evaluation, a dropout layer ($p = 0.1$) was inserted before the classification head, activated during both training and inference (Monte Carlo Dropout). This modification impacts model metrics variably: **DenseNet** benefits most through enhanced regularization and reduced overfitting, preserving 94% accuracy on 95% coverage (lowest viable threshold). For stricter selectivity, **ResNet18** excels—retaining 75% predictions at 99% accuracy—balancing coverage and precision optimally. These analyses are critical in medical AI: reliable uncertainty reduces prediction stochasticity, minimizing clinical errors with profound patient impact. Deployment requires optimizing the accuracy-coverage trade-off via thresholds that maximize utility while minimizing risk. Moreover, uncertainty flags borderline cases for human review or additional diagnostics, enhancing decision support safety and efficacy.

5.3 Addressing the Research Questions

5.3.1 RQ1: Feasibility of 3D Deep Learning Classification

Yes—both CNNs and transformers effectively classify EDC-exposed organoids from full z-stack volumes. SwinUNETR achieves 0.93 accuracy at 512^3 baseline, while CNNs (DenseNet/ResNet18) reach 0.94 after resolution optimization. Success stems from patch-based processing: 128^3 patches capture organoid morphology without memory overflow, though minority Compact class requires careful downsampling to avoid omission.

5.3.2 RQ2: Transformers vs CNNs Performance

Depends on the resolutions. Attention-based models excels in high-res and medium-res settings (512^3: 0.93 acc; 256^3: 0.90), leveraging global context and multi-scale features. However, at low-res (128^3), CNNs outperform (DenseNet: 0.94 acc) due to their local inductive biases and reduced overfitting on small data and with a small dataset. In this kind of limited data and task, pure attention models (SwinViT) underperform compared to hybrid architectures (SwinUNETR), which combine attention with convolutional decoders for refined spatial feature fusion, especially at lower resolutions where local details are crucial.

5.3.3 RQ3: Complexity vs Efficiency Trade-off

Light CNNs win for constrained deployment: ResNet18/DenseNet inference <50ms at optimal 128^3, versus SwinUNETR's 400+ms. However, Attention models

maintain stable performance across resolutions, while CNNs require careful tuning to avoid overfitting or underfitting. Simple classification heads outperform complex NOAH variants, confirming that the rich features extracted by the backbone suffice for accurate classification without additional complexity. Patch merging consistently degrades performance and increases inference time, making non-overlapping patches the preferred strategy for this classification task.

5.3.4 RQ4: Resolution Impact on Performance

CNNs need low-res (128^3) for accuracy; transformers stable across scales. High-res cripples CNNs (limited receptive field misses organoid context) but suits transformers' global attention. Downsampling boosts CNNs +30% acc but slows preprocessing (SA dominates). **Compact class breakthrough**: all models hit 100% recall at 128^3, but transformers sacrifice Cauliflower/Cystic precision via majority confusion.

5.3.5 RQ5: Optimal Preprocessing Strategy

Hybrid SA(XY)+SS(Z) best: Spatial Abstractor preserves XY morphology for CNN local biases; Slice Selector thins redundant Z-slices (22ms vs SA's 678ms). Pure SS fragments 3D context, dropping accuracy 5-10%; pure SA wastes compute on empty slices. **Visualization confirmation**: SA maintains organoid borders; SS creates gaps transformers can't bridge.

5.3.6 RQ6: Adaptive Z-Stack Selection

Yes, but hybrid only: SS alone cuts latency 30x (678→23ms) yet degrades performance via spatial discontinuity. CNNs tolerate Z-thinning (ResNet18: 0.92 acc @32Z); transformers collapse (SwinUNETR: 0.85). **Optimal**: SS first (Z→32), then SA (XY→128)—dual gains in speed (+70%) and accuracy (+5% for CNNs).

5.4 Limitations and Future Directions

Despite the promising results, several limitations must be acknowledged. The dataset comprises only 800 samples, constraining generalizability and amplifying overfitting risks—particularly for parameter-heavy models like SwinUNETR. Only five architectures were evaluated (DenseNet, ResNet18/50, SwinUNETR, SwinViT), potentially overlooking other promising backbones. Attention mechanism variants such as vanilla ViT or DeiT were excluded due to their prohibitive computational demands.

Future work should address these gaps through several avenues:

Model Expansion and Validation: Evaluate additional architectures, including vanilla Vision Transformers, DeiT variants, and emerging efficient transformers (e.g., MobileViT). Cross-domain validation on larger, diverse organoid datasets—or

even different cell phenotypes—would strengthen scalability claims and robustness to biological variability.

Data Augmentation: Develop organoid-specific augmentation strategies (e.g., elastic deformations, intensity perturbations mimicking staining variations, synthetic morphology blending) to mitigate overfitting and enhance generalization, particularly for minority classes like Compact.

Hybrid Architectures: Investigate sophisticated CNN-transformer hybrids, such as attention-augmented CNNs (e.g., ResNet+CBAM) or transformers with local inductive biases (e.g., ConvNeXt blocks in Swin stages), to optimize the accuracy-efficiency trade-off for resource-constrained medical deployments.

Preprocessing Optimization: Explore advanced dimensionality reduction (e.g., autoencoder-based compression, learned patch selection via reinforcement learning) and intelligent patch sampling strategies that prioritize discriminative regions, further reducing computational overhead without accuracy loss.

Interpretability: Apply explainability methods—such as Grad-CAM3D, attention rollout, or SHAP values—to identify which morphological features (border complexity, internal texture, volume distribution) drive phenotype classification. Such biological insights would enhance clinical trust and potentially reveal EDC exposure signatures invisible to human experts.

Chapter 6

Conclusions

6.1 Summary of Key Findings

This thesis systematically evaluated 3D deep learning for organoid phenotype classification under EDC exposure, addressing all six research questions through comprehensive ablation studies:

1. **3D DL is feasible**: Patch-based processing with 128×128×128 patches enables effective classification from full z-stacks (SwinUNETR: 0.93 acc).

2. **Hybrid transformers superior**: SwinUNETR outperforms pure SwinViT and CNNs across resolutions, combining global attention with CNN-like spatial fusion.

3. **CNNs for efficiency**: ResNet18/DenseNet achieve 0.94 acc at <50ms inference—optimal for constrained deployment.

4. **Resolution dichotomy**: CNNs require aggressive downsampling (128^3); transformers remain stable but sacrifice majority-class precision.

5. **Hybrid preprocessing optimal**: SA(XY)+SS(Z) balances morphology preservation, speed (22ms), and accuracy.

6. **Uncertainty-aware deployment**: DenseNet achieves 94% accuracy on 95% coverage via MC-Dropout thresholding.

6.2 Practical Recommendations

For **research settings** with ample compute: `SwinUNETR + 256×256×128 + SA preprocessing + linear head`.

For **clinical/production deployment**: `ResNet18 + 128×128×128 + hybrid SA+SS + MC-Dropout (p=0.1)`.

Never use patch merging (10% accuracy loss, 50% latency increase) or complex NOAH heads (no gains over linear).

6.3 Scientific Contributions

This thesis makes the following contributions to automated 3D organoid analysis:

1. **First systematic 3D organoid classification benchmark** comparing CNNs versus hybrid transformers on EDC-induced phenotypes, establishing performance baselines and architectural trade-offs.

2. **Hybrid preprocessing pipeline** combining Spatial Abstracter (SA) and Slice Selector (SS), reducing inference time by 30× while preserving classification accuracy and morphological fidelity.

3. **Resolution-dependent architecture guide**: CNNs excel at low resolution (128^3) for efficient deployment; transformers maintain performance stability across scales for research applications.

4. **Uncertainty validation framework** demonstrating DenseNet's clinical reliability, achieving 94% accuracy on 95% prediction coverage via Monte Carlo Dropout thresholding.

6.4 Final Remarks

This work establishes 3D deep learning as clinically viable for organoid phenotyping, bridging AI and toxicology. While dataset scale remains the primary bottleneck, the proposed configurations generalize across limited medical imaging scenarios. Future integration with multi-omics data and real-time inference will unlock automated EDC screening at scale, accelerating toxicology discovery.

Appendix A

Experimental Results

This appendix compiles all detailed tables of experimental results discussed in the main thesis. The tables are organized by model and configuration, with cross-references to Chapter X (Results). For completeness, they include full metrics: accuracy, precision, recall, F1-score, uncertainty (MC-Dropout), inference times, and memory requirements.

A.1 Baseline: Full-resolution volumes (512×512×128)

A.1.1 DenseNet

Table A.1: Densenet baseline - Testing - Accuracy 0.537

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.513	1.000	0.095	0.678	40
Compact	0.000	0.000	1.000	0.000	5
Cystic	1.000	0.108	1.000	0.195	37
Macro avg	0.504	0.369	0.698	0.291	82
Weighted avg	0.701	0.537	0.559	0.419	82
Micro avg	0.537	0.537	0.768	0.537	82

Table A.2: Densenet baseline - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	40	0	0
Compact	5	0	0
Cystic	33	0	4

A.1.2 ResNet18

Table A.3: Resnet18 baseline - Testing - Accuracy 0,610

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	1,00	0,325	1,000	0,491	40
Compact	0,000	0,000	1,000	0,000	5
Cystic	0,536	1,000	0,289	0,698	37
Macro avg	0,512	0,442	0,763	0,396	82
Weighted avg	0,730	0,610	0,679	0,554	82
Micro avg	0,610	0,610	0,805	0,610	82

Table A.4: Resnet18 baseline - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	13	0	27
Compact	0	0	5
Cystic	0	0	37

A.1.3 ResNet50

Table A.5: Resnet50 baseline - Testing - Accuracy 0,549

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0,525	0,775	0,333	0,626	40
Compact	0,000	0,000	1,000	0,000	5
Cystic	0,609	0,378	0,800	0,467	37
Macro avg	0,378	0,384	0,711	0,364	82
Weighted avg	0,531	0,549	0,585	0,516	82
Micro avg	0,549	0,549	0,774	0,549	82

Table A.6: Resne50 baseline - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	31	0	9
Compact	5	0	0
Cystic	23	0	14

A.1.4 SwinVit

Table A.7: SwinVit baseline - Testing - Accuracy 0,793

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0,939	0,775	0,952	0,849	40
Compact	0,000	0,000	0,974	0,000	5
Cystic	0,723	0,919	0,711	0,810	37
Macro avg	0,554	0,565	0,879	0,553	82
Weighted avg	0,785	0,793	0,845	0,780	82
Micro avg	0,793	0,793	0,896	0,793	82

Table A.8: SwinVit baseline - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	31	1	8
Compact	0	0	5
Cystic	2	1	34

A.1.5 SwinUNETR

Table A.9: SwinUNETR baseline - Testing - Accuracy 0,927

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0,975	0,975	0,975	0,975	40
Compact	0,600	0,600	0,974	0,600	5
Cystic	0,919	0,919	0,933	0,919	37
Macro avg	0,831	0,831	0,961	0,831	82
Weighted avg	0,927	0,927	0,957	0,927	82
Micro avg	0,927	0,927	0,963	0,927	82

Table A.10: SwinUNETR baseline - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	39	0	1
Compact	0	3	2
Cystic	1	2	34

A.1.6 SwinUNETR+NOAH

Table A.11: SwinUNETR+noah baseline- Testing - Accuracy 0.854

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	0.881	0.925	0.881	0.902	40
Compact	0.667	0.400	0.987	0.500	5
Cystic	0.838	0.838	0.867	0.838	37
Macro avg	0.795	0.721	0.912	0.747	82
Weighted avg	0.848	0.854	0.881	0.849	82
Micro avg	0.854	0.854	0.927	0.854	82

Table A.12: Swiunetr + noah baseline - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	37	0	3
Compact	0	2	3
Cystic	5	1	31

A.2 Reduced XY-resolution: 256×256×128 volumes, patches of 128×128×128

A.2.1 DenseNet

Table A.13: Densenet 256x256x256 - Testing - Accuracy 0.890

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.884	0.850	0.881	0.916	40
Compact	0.571	0.800	0.961	0.667	5
Cystic	0.969	0.838	0.978	0.899	37
Macro avg	0.808	0.863	0.940	0.827	82
Weighted avg	0.903	0.890	0.930	0.893	82
Micro avg	0.890	0.890	0.945	0.890	82

Table A.14: Densenet 256x256x256 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	38	1	1
Compact	1	4	0
Cystic	4	2	32

A.2.2 ResNet18

Table A.15: Resnet18 256x256x256 - Testing - Accuracy 0.878

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.944	0.850	0.952	0.895	40
Compact	0.667	0.800	0.874	0.727	5
Cystic	0.850	0.919	0.867	0.883	37
Macro avg	0.820	0.856	0.931	0.835	82
Weighted avg	0.885	0.878	0.915	0.879	82
Micro avg	0.878	0.878	0.939	0.878	82

Table A.16: Resnet18 256x256x256 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	34	1	5
Compact	0	4	1
Cystic	2	1	34

A.2.3 ResNet50

Table A.17: Resnet50 256x256x256 - Testing - Accuracy 0.756

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.962	0.625	0.976	0.758	40
Compact	0.182	0.400	0.883	0.250	5
Cystic	0.778	0.946	0.778	0.854	37
Macro avg	0.640	0.657	0.879	0.620	82
Weighted avg	0.931	0.756	0.881	0.770	82
Micro avg	0.756	0.756	0.878	0.756	82

Table A.18: Resnet50 256x256x256 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	25	8	7
Compact	0	2	3
Cystic	1	1	35

A.2.4 SwinVit

Table A.19: SwinVit 256x256x256 - Testing - Accuracy 0.878

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.923	0.900	0.929	0.911	40
Compact	0.667	0.400	0.987	0.500	5
Cystic	0.850	0.919	0.867	0.883	37
Macro avg	0.813	0.740	0.927	0.765	82
Weighted avg	0.874	0.878	0.904	0.874	82
Micro avg	0.878	0.878	0.939	0.878	82

Table A.20: SwinVit 256x256x256 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	36	0	4
Compact	1	2	2
Cystic	2	1	34

A.2.5 SwinUNETR

Table A.21: SwinUNETR 256x256x256 - Testing - Accuracy 0.902

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	1.000	0.850	1.000	0.919	40
Compact	0.800	0.800	0.987	0.800	5
Cystic	0.837	0.973	0.844	0.900	37
Macro avg	0.879	0.874	0.944	0.873	82
Weighted avg	0.914	0.902	0.929	0.903	82
Micro avg	0.902	0.902	0.951	0.902	82

Table A.22: SwinUNETR 256x256x256 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	34	0	6
Compact	0	4	1
Cystic	0	1	36

A.2.6 SwinUNETR+NOAH

Table A.23: SwinUNETR+Noah 256x256x256 - Testing - Accuracy 0.878

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	0.970	0.800	0.976	0.877	40
Compact	0.833	1.000	0.987	0.909	5
Cystic	0.814	0.946	0.822	0.875	37
Macro avg	0.872	0.915	0.928	0.887	82
Weighted avg	0.891	0.878	0.907	0.878	82
Micro avg	0.878	0.878	0.939	0.878	82

Table A.24: SwinUNETR+Noah 256x256x256 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	32	0	8
Compact	0	5	0
Cystic	1	1	35

A.3 Reduced XY-resolution: 256×256×128 volumes, patches of 64x64x64

A.3.1 DenseNet

Table A.25: Densenet 256x256x256 ps 64x64x64 - Testing - Accuracy 0.780

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.850	0.975	0.690	0.848	40
Compact	0.000	0.000	1.000	0.000	5
Cystic	0.833	0.676	0.889	0.746	37
Macro avg	0.528	0.550	0.860	0.531	82
Weighted avg	0.742	0.780	0.799	0.750	82
Micro avg	0.780	0.780	0.890	0.780	82

Table A.26: Densenet 256x256x256 ps 64x64x64 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	39	0	1
Compact	1	0	4
Cystic	12	0	25

A.3.2 ResNet18

Table A.27: Resnet18 256x256x256 ps 64x64x64 - Testing - Accuracy 0.780

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.821	0.800	0.833	0.810	40
Compact	0.000	0.000	1.000	0.000	5
Cystic	0.744	0.965	0.756	0.800	37
Macro avg	0.522	0.555	0.863	0.537	82
Weighted avg	0.736	0.780	0.808	0.756	82
Micro avg	0.780	0.780	0.890	0.780	82

Table A.28: Resnet18 256x256x256 ps 64x64x64 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	32	0	8
Compact	2	0	3
Cystic	5	0	32

A.3.3 SwinVit

Table A.29: SwinVit 256x256x256 ps 64x64x64 - Testing - Accuracy 0.866

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	1.000	0.825	1.000	0.904	40
Compact	0.500	1.000	0.935	0.667	5
Cystic	0.846	0.892	0.867	0.867	37
Macro avg	0.782	0.906	0.934	0.813	82
Weighted avg	0.900	0.866	0.936	0.874	82
Micro avg	0.866	0.866	0.933	0.866	82

Table A.30: SwinVit 256x256x256 ps 64x64x64 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	33	1	6
Compact	0	5	0
Cystic	0	4	33

A.3.4 SwinUNETR

Table A.31: SwinUNETR 256x256x256 ps 64x64x64 - Testing - Accuracy 0.902

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	1.000	0.825	1.000	0.904	40
Compact	0.714	1.000	0.974	0.833	5
Cystic	0.857	0.973	0.867	0.911	37
Macro avg	0.857	0.933	0.947	0.883	82
Weighted avg	0.918	0.902	0.938	0.903	82
Micro avg	0.902	0.902	0.951	0.902	82

Table A.32: SwinUNETR 256x256x256 ps 64x64x64 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	33	1	6
Compact	0	5	0
Cystic	0	1	36

A.4 Patch aggregation: Overlapping patch merging

A.4.1 DenseNet

Table A.33: Densenet Patch Merging - Testing - Accuracy 0.451

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.000	0.000	1.000	0.000	40
Compact	0.000	0.000	1.000	0.000	5
Cystic	0.451	1.000	0.000	0.622	37
Macro avg	0.150	0.333	0.333	0.207	82
Weighted avg	0.204	0.451	0.549	0.281	82
Micro avg	0.451	0.451	0.726	0.281	82

Table A.34: Densenet Patch Merging - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	0	0	40
Compact	0	0	5
Cystic	0	0	37

A.4.2 ResNet18

Table A.35: Resnet18 Patch Merging - Testing - Accuracy 0.732

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.897	0.650	0.929	0.754	40
Compact	0.500	0.200	0.987	0.286	5
Cystic	0.647	0.892	0.600	0.750	37
Macro avg	0.681	0.581	0.839	0.596	82
Weighted avg	0.760	0.732	0.784	0.723	82
Micro avg	0.732	0.732	0.866	0.732	82

Table A.36: Resnet18 Patch Merging - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	26	0	14
Compact	0	1	4
Cystic	3	1	33

A.4.3 ResNet50

Table A.37: Resnet50 Patch Merging - Testing - Accuracy 0.756

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.962	0.625	0.976	0.758	40
Compact	0.182	0.400	0.883	0.250	5
Cystic	0.778	0.946	0.778	0.854	37
Macro avg	0.640	0.657	0.879	0.620	82
Weighted avg	0.831	0.756	0.881	0.770	82
Micro avg	0.756	0.756	0.878	0.756	82

Table A.38: Resnet50 Patch Merging - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	25	8	7
Compact	0	2	3
Cystic	1	1	35

A.4.4 SwinUNETR

Table A.39: SwinUNETR Patch Merging - Testing - Accuracy 0.866

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.897	0.875	0.905	0.886	40
Compact	0.714	1.000	0.974	0.833	5
Cystic	0.861	0.838	0.889	0.849	37
Macro avg	0.824	0.904	0.923	0.856	82
Weighted avg	0.870	0.866	0.902	0.866	82
Micro avg	0.866	0.866	0.933	0.866	82

Table A.40: SwinUNETR Patch Merging - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	35	0	5
Compact	0	5	0
Cystic	4	2	31

A.5 Reduced full-resolution: 128×128×128 volumes

A.5.1 DenseNet

Table A.41: Densenet 128x128x128 - Testing - Accuracy 0.890

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	1.000	0.875	1.000	0.933	40
Compact	0.500	0.600	0.961	0.545	5
Cystic	0.854	0.946	0.867	0.897	37
Macro avg	0.785	0.807	0.943	0.792	82
Weighted avg	0.903	0.890	0.937	0.893	82
Micro avg	0.890	0.890	0.945	0.890	82

Table A.42: Densenet 128x128x128 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	35	1	4
Compact	0	3	2
Cystic	0	2	35

A.5.2 ResNet18

Table A.43: Resnet18 128x128x128 - Testing - Accuracy 0.939

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.974	0.925	0.976	0.949	40
Compact	0.833	1.000	0.987	0.909	5
Cystic	0.921	0.946	0.933	0.933	37
Macro avg	0.909	0.957	0.966	0.930	82
Weighted avg	0.941	0.939	0.958	0.939	82
Micro avg	0.939	0.939	0.970	0.939	82

Table A.44: Resnet18 128x128x128 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	37	0	3
Compact	0	5	0
Cystic	1	1	35

A.5.3 SwinVit

Table A.45: SwinVit 128x128x128 - Testing - Accuracy 0.841

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.912	0.775	0.929	0.838	40
Compact	0.625	1.000	0.961	0.769	5
Cystic	0.825	0.892	0.844	0.857	37
Macro avg	0.787	0.889	0.911	0.821	82
Weighted avg	0.855	0.841	0.893	0.842	82
Micro avg	0.841	0.841	0.821	0.841	82

Table A.46: SwinVit 128x128x128 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	31	2	7
Compact	0	5	0
Cystic	3	1	33

A.5.4 SwinUNETR

Table A.47: SwinUNETR 128x128x128 - Testing - Accuracy 0.878

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.970	0.800	0.976	0.877	40
Compact	0.833	1.000	0.987	0.909	5
Cystic	0.814	0.946	0.822	0.875	37
Macro avg	0.872	0.915	0.928	0.887	82
Weighted avg	0.891	0.878	0.907	0.878	82
Micro avg	0.878	0.878	0.939	0.878	82

Table A.48: SwinUNETR 128x128x128 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	32	0	8
Compact	0	5	0
Cystic	1	1	35

A.5.5 SwinUNETR+NOAH

Table A.49: SwinUNETR + Noah 128x128x128 - Testing - Accuracy 0.793

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	0.906	0.725	0.929	0.806	40
Compact	0.250	0.200	0.961	0.222	5
Cystic	0.761	0.946	0.756	0.843	37
Macro avg	0.639	0.624	0.882	0.624	82
Weighted avg	0.800	0.793	0.852	0.787	82
Micro avg	0.793	0.793	0.896	0.793	82

Table A.50: SwinUNETR + Noah 128x128x128 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	29	2	9
Compact	2	1	2
Cystic	1	1	35

A.6 Transfer learning: scratch vs pretrained

A.6.1 Resnet18

Table A.51: Resnet18 Scratch - Testing - Accuracy 0,829

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0,969	0,775	0,976	0,861	40
Compact	0,500	0,200	0,987	0,286	5
Cystic	0,750	0,973	0,733	0,847	37
Macro avg	0,740	0,649	0,899	0,665	82
Weighted avg	0,841	0,829	0,867	0,820	82
Micro avg	0,829	0,829	0,915	0,829	82

Table A.52: Resnet18 Scratch - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	31	1	8
Compact	0	1	4
Cystic	1	0	36

A.6.2 SwinVit

Table A.53: SwinVit Scratch - Testing - Accuracy 0,829

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0,895	0,850	0,905	0,872	40
Compact	0,000	0,000	1,000	0,000	5
Cystic	0,773	0,919	0,778	0,840	37
Macro avg	0,556	0,590	0,894	0,570	82
Weighted avg	0,785	0,829	0,853	0,804	82
Micro avg	0,829	0,829	0,915	0,829	82

Table A.54: SwinVit Scratch - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	34	0	6
Compact	1	0	4
Cystic	3	0	34

A.6.3 SwinUNETR

Table A.55: SwinUNETR Scratch - Testing - Accuracy 0,902

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	0,973	0,900	0,976	0,935	40
Compact	0,667	0,400	0,987	0,500	5
Cystic	0,857	0,973	0,867	0,911	37
Macro avg	0,832	0,758	0,943	0,782	82
Weighted avg	0,902	0,902	0,927	0,898	82
Micro avg	0,902	0,902	0,951	0,902	82

Table A.56: SwinUNETR Scratch - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	36	1	3
Compact	0	2	3
Cystic	1	0	36

A.7 Slice reduction strategies: 128×128×64(Z-thinning)

A.7.1 DenseNet

Table A.57: Densenet Slice 64 - Testing - Accuracy 0.890

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.923	0.900	0.929	0.911	40
Compact	0.667	0.400	0.987	0.500	5
Cystic	0.875	0.946	0.889	0.909	37
Macro avg	0.822	0.749	0.935	0.773	82
Weighted avg	0.886	0.890	0.914	0.885	82
Micro avg	0.890	0.890	0.945	0.890	82

Table A.58: Densenet Slice 64 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	36	0	4
Compact	2	2	1
Cystic	1	1	35

A.7.2 ResNet18

Table A.59: Resnet18 Slice 64 - Testing - Accuracy 0.915

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.000	0.875	0.000	0.933	40
Compact	0.800	0.800	0.987	0.800	5
Cystic	0.857	0.973	0.867	0.911	37
Macro avg	0.886	0.883	0.951	0.882	82
Weighted avg	0.923	0.915	0.939	0.915	82
Micro avg	0.915	0.915	0.57	0.915	82

Table A.60: Resnet18 Slice 64 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	35	0	5
Compact	0	4	1
Cystic	0	1	36

A.7.3 SwinVit

Table A.61: SwinVit Slice 64 - Testing - Accuracy 0.841

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.968	0.750	0.976	0.845	40
Compact	0.667	0.800	0.974	0.727	5
Cystic	0.778	0.946	0.778	0.854	37
Macro avg	0.804	0.832	0.909	0.809	82
Weighted avg	0.864	0.841	0.887	0.842	82
Micro avg	0.841	0.841	0.821	0.841	82

Table A.62: SwinVit Slice 64 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	30	1	9
Compact	0	4	1
Cystic	1	1	35

A.7.4 SwinUNETR

Table A.63: SwinUNETR Slice 64 - Testing - Accuracy 0.866

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.919	0.850	0.929	0.883	40
Compact	0.667	0.800	0.974	0.727	5
Cystic	0.846	0.892	0.867	0.868	37
Macro avg	0.811	0.847	0.923	0.826	82
Weighted avg	0.871	0.866	0.903	0.867	82
Micro avg	0.866	0.866	0.933	0.866	82

Table A.64: SwinUNETR Slice 64 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	34	1	5
Compact	0	4	1
Cystic	3	1	33

A.8 Slice reduction strategies: 128×128×32(Z-thinning)

A.8.1 DenseNet

Table A.65: Densenet Slice 32- Testing - Accuracy 0.890

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.923	0.900	0.929	0.911	40
Compact	0.667	0.800	0.974	0.727	5
Cystic	0.892	0.892	0.911	0.892	37
Macro avg	0.827	0.864	0.938	0.844	82
Weighted avg	0.893	0.890	0.923	0.891	82
Micro avg	0.891	0.891	0.945	0.890	82

Table A.66: Densenet Slice 32 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	36	0	4
Compact	1	4	0
Cystic	2	2	33

A.8.2 ResNet18

Table A.67: Resnet18 Slice 32 - Testing - Accuracy 0.915

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	1.000	0.825	1.000	0.904	40
Compact	1.000	1.000	1.000	1.000	5
Cystic	0.841	1.000	0.844	0.914	37
Macro avg	0.947	0.942	0.948	0.939	82
Weighted avg	0.928	0.915	0.930	0.914	82
Micro avg	0.914	0.914	0.957	0.915	82

Table A.68: Resnet18 Slice 32 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	33	0	7
Compact	0	5	0
Cystic	0	0	37

A.8.3 SwinVit

Table A.69: SwinVit Slice 32 - Testing - Accuracy 0.805

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.795	0.875	0.786	0.833	40
Compact	0.667	0.800	0.974	0.727	5
Cystic	0.844	0.730	0.889	0.783	37
Macro avg	0.769	0.802	0.883	0.781	82
Weighted avg	0.809	0.805	0.844	0.804	82
Micro avg	0.805	0.805	0.902	0.805	82

Table A.70: SwinVit Slice 32 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	35	1	4
Compact	0	4	1
Cystic	9	1	27

A.8.4 SwinUNETR

Table A.71: SwinUNETR Slice 32 - Testing - Accuracy 0.854

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.914	0.800	0.929	0.853	40
Compact	0.714	1.000	0.971	0.833	5
Cystic	0.825	0.892	0.844	0.857	37
Macro avg	0.818	0.897	0.916	0.848	82
Weighted avg	0.862	0.854	0.893	0.854	82
Micro avg	0.854	0.854	0.827	0.854	82

Table A.72: SwinUNETR Slice 32 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	32	1	7
Compact	0	5	0
Cystic	3	1	33

A.9 Slice reduction strategies: 32x32x32 (Z-thinning)

A.9.1 DenseNet

Table A.73: DenseNet 32x32x32 Z-thinning - Testing - Accuracy 0.805

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.892	0.825	0.905	0.857	40
Compact	0.000	0.000	0.987	0.000	5
Cystic	0.750	0.892	0.756	0.815	37
Macro avg	0.547	0.572	0.882	0.557	82
Weighted avg	0.773	0.805	0.842	0.786	82
Micro avg	0.805	0.805	0.902	0.805	82

Table A.74: DenseNet 32x32x32 Z-thinning - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	33	1	6
Compact	0	0	5
Cystic	4	0	33

A.9.2 ResNet18

Table A.75: ResNet18 32x32x32 Z-thinning - Testing - Accuracy 0.878

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.946	0.875	0.952	0.909	40
Compact	0.750	0.600	0.987	0.667	5
Cystic	0.829	0.919	0.844	0.872	37
Macro avg	0.842	0.798	0.928	0.816	82
Weighted avg	0.881	0.878	0.906	0.877	82
Micro avg	0.878	0.878	0.939	0.878	82

Table A.76: ResNet18 32x32x32 Z-thinning - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	35	0	5
Compact	0	3	2
Cystic	2	1	34

A.9.3 SwinVit

Table A.77: SwinVit 32x32x32 Z-thinning - Testing - Accuracy 0.732

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	0.920	0.575	0.952	0.708	40
Compact	0.600	0.600	0.974	0.600	5
Cystic	0.654	0.919	0.600	0.764	37
Macro avg	0.725	0.698	0.842	0.691	82
Weighted avg	0.780	0.732	0.795	0.727	82
Micro avg	0.732	0.732	0.866	0.732	82

Table A.78: SwinVit 32x32x32 Z-thinning - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	23	1	16
Compact	0	3	2
Cystic	2	1	34

A.10 Slice reduction strategies: 32x32x32 (XYZ-thinning)

A.10.1 DenseNet

Table A.79: DenseNet 32x32x32 XYZ-thinning - Testing - Accuracy 0.732

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.810	0.850	0.810	0.829	40
Compact	0.000	0.000	0.922	0.000	5
Cystic	0.765	0.703	0.822	0.832	37
Macro avg	0.525	0.518	0.851	0.521	82
Weighted avg	0.740	0.732	0.822	0.765	82
Micro avg	0.732	0.732	0.866	0.732	82

Table A.80: DenseNet 32x32x32 XYZ-thinning - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	34	1	5
Compact	2	0	3
Cystic	6	5	26

A.10.2 ResNet18

Table A.81: ResNet18 32x32x32 XYZ-thinning - Testing - Accuracy 0.902

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	1.000	0.825	1.000	0.904	40
Compact	0.714	1.000	0.974	0.833	5
Cystic	0.857	0.973	0.867	0.811	37
Macro avg	0.857	0.933	0.947	0.883	82
Weighted avg	0.918	0.902	0.938	0.903	82
Micro avg	0.902	0.902	0.851	0.902	82

Table A.82: ResNet18 32x32x32 XYZ-thinning - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	33	1	6
Compact	0	5	0
Cystic	0	1	36

A.10.3 SwinVit

Table A.83: SwinVit 32x32x32 XYZ-thinning - Testing - Accuracy 0.866

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	0.857	0.900	0.857	0.878	40
Compact	0.714	1.000	0.974	0.833	5
Cystic	0.909	0.811	0.933	0.857	37
Macro avg	0.827	0.904	0.922	0.856	82
Weighted avg	0.872	0.866	0.899	0.866	82
Micro avg	0.866	0.866	0.933	0.866	82

Table A.84: SwinVit 32x32x32 XYZ-thinning - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	36	1	3
Compact	0	5	0
Cystic	6	1	30

A.11 Slice reduction strategies: 128×128×128(XYZ-thinning)

A.11.1 DenseNet

Table A.85: Densenet Slice Selector 128x128x128 - Testing - Accuracy 0.878

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.905	0.950	0.905	0.927	40
Compact	0.667	0.400	0.987	0.500	5
Cystic	0.865	0.865	0.889	0.865	37
Macro avg	0.812	0.738	0.927	0.764	82
Weighted avg	0.872	0.878	0.903	0.873	82
Micro avg	0.878	0.878	0.939	0.878	82

Table A.86: Densenet Slice Selector 128x128x128 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	38	0	2
Compact	0	2	3
Cystic	4	1	32

A.11.2 ResNet18

Table A.87: ResNet18 Slice Selector 128x128x128 - Testing - Accuracy 0.890

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	1.000	0.900	1.000	0.947	40
Compact	0.500	0.200	0.987	0.286	5
Cystic	0.818	0.973	0.822	0.889	37
Macro avg	0.773	0.691	0.936	0.707	82
Weighted avg	0.887	0.890	0.919	0.881	82
Micro avg	0.890	0.890	0.945	0.890	82

Table A.88: ResNet18 Slice Selector 128x128x128 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	36	0	4
Compact	0	1	4
Cystic	0	1	36

A.11.3 SwinVit

Table A.89: SwinVit Slice Selector 128x128x128 - Testing - Accuracy 0.841

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.900	0.900	0.905	0.900	40
Compact	0.500	0.400	0.974	0.444	5
Cystic	0.816	0.838	0.844	0.827	37
Macro avg	0.739	0.713	0.908	0.724	82
Weighted avg	0.838	0.841	0.882	0.839	82
Micro avg	0.841	0.841	0.821	0.841	82

Table A.90: SwinVit Slice Selector 128x128x128 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	36	0	4
Compact	0	2	3
Cystic	4	2	31

A.11.4 SwinUNETR

Table A.91: SwinUNETR Slice Selector 128x128x128 - Testing - Accuracy 0.829

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.860	0.925	0.857	0.892	40
Compact	0.400	0.400	0.961	0.400	5
Cystic	0.853	0.784	0.889	0.817	37
Macro avg	0.704	0.703	0.902	0.703	82
Weighted avg	0.829	0.829	0.878	0.828	82
Micro avg	0.829	0.829	0.915	0.829	82

Table A.92: SwinUNETR Slice Selector 128x128x128 - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	37	1	2
Compact	0	2	3
Cystic	6	2	29

A.12 Monte Carlo Dropout: Uncertainty estimation

A.12.1 DenseNet

MC-Dropout Uncertainty Analysis – Predictive Entropy	
Correct predictions	0.3503 ± 0.2114
Incorrect predictions	0.5871 ± 0.1720
Uncertainty gap	0.2368

Per-class uncertainty		
Class 0	0.3858 ± 0.1984	$(n = 40)$
Class 1	0.0883 ± 0.1024	$(n = 5)$
Class 2	0.3858 ± 0.2232	$(n = 37)$

Selective prediction (rejection thresholds)			
Threshold	Condition	Kept	Accuracy
Reject top 50%	> 0.3243	41/82	0.9756
Reject top 25%	> 0.5519	61/82	0.9672
Reject top 10%	> 0.6589	73/82	0.9315
Reject top 5%	> 0.6932	77/82	0.9351

Calibration (accuracy by uncertainty bin)			
Bin	Range	Accuracy	n
Low (0–25%)	[0.0007, 0.1867)	1.0000	21
Medium (25–50%)	[0.1867, 0.3243)	0.9500	20
High (50–75%)	[0.3243, 0.5519)	0.9500	20
Very High (75–100%)	[0.5519, 0.9300)	0.8000	20

Table A.93: Densenet Montecarlo Dropout - Testing - Accuracy 0.927

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	0.927	0.875	0.976	0.921	40
Compact	1.000	1.000	1.000	1.000	5
Cystic	0.878	0.973	0.889	0.923	37
Macro avg	0.950	0.949	0.955	0.948	82
Weighted avg	0.931	0.927	0.938	0.927	82
Micro avg	0.927	0.927	0.963	0.927	82

Table A.94: Densenet Montecarlo Dropout - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	35	0	5
Compact	0	5	0
Cystic	1	0	36

A.12.2 ResNet18

MC-Dropout Uncertainty Analysis – Predictive Entropy	
Correct predictions	0.1872 ± 0.2323
Incorrect predictions	0.5356 ± 0.2046
Uncertainty gap	0.3484

Per-class uncertainty		
Class 0	0.1346 ± 0.1973	$(n = 40)$
Class 1	0.6680 ± 0.2644	$(n = 5)$
Class 2	0.2577 ± 0.2341	$(n = 37)$

Selective prediction (rejection thresholds)			
Threshold	Condition	Kept	Accuracy
Reject top 50%	> 0.3312	47/82	1.0000
Reject top 25%	> 0.8312	61/82	0.9836
Reject top 10%	> 0.6685	73/82	0.9178
Reject top 5%	> 0.732	77/82	0.9091

Calibration (accuracy by uncertainty bin)			
Bin	Range	Accuracy	n
Low (0–25%)	[0.0001, 0.1325)	1.0000	21
Medium (25–50%)	[0.1325, 0.3132)	1.0000	20
High (50–75%)	[0.3132, 0.3432)	0.9500	20
Very High (75–100%)	[0.3432, 1.0776)	0.6500	20

Table A.95: Resnet18 Montecarlo Dropout - Testing - Accuracy 0.902

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	0.946	0.875	0.952	0.909	40
Compact	0.833	0.000	0.987	0.909	5
Cystic	0.872	0.919	0.889	0.895	37
Macro avg	0.884	0.931	0.943	0.904	82
Weighted avg	0.906	0.902	0.926	0.903	82
Micro avg	0.902	0.902	0.951	0.902	82

Table A.96: Resnet18 Montecarlo Dropout - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	35	0	5
Compact	0	5	0
Cystic	2	1	34

A.12.3 SwinVit

MC-Dropout Uncertainty Analysis – Predictive Entropy	
Correct predictions	0.1701 ± 0.2064
Incorrect predictions	0.5042 ± 0.2510
Uncertainty gap	0.3341

Per-class uncertainty		
Class 0	0.1698 ± 0.2380	$(n = 40)$
Class 1	0.6232 ± 0.1883	$(n = 5)$
Class 2	0.2265 ± 0.2189	$(n = 37)$

Selective prediction (rejection thresholds)			
Threshold	Condition	Kept	Accuracy
Reject top 50%	$> 0.0.0968$	61/82	0.9756
Reject top 25%	> 0.3808	61/82	0.9344
Reject top 10%	> 0.6252	73/82	0.9041
Reject top 5%	> 0.7053	77/82	0.8701

Calibration (accuracy by uncertainty bin)			
Bin	Range	Accuracy	n
Low (0–25%)	[0.0015, 0.1027)	1.0000	21
Medium (25–50%)	[0.1027, 0.9627)	1.0000	20
High (50–75%)	[0.9627, 0.9868)	0.9500	20
Very High (75–100%)	[0.9868, 1.0665)	0.5000	20

Table A.97: SwinVit Montecarlo Dropout - Testing - Accuracy 0.854

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	0.944	0.850	0.852	0.895	40
Compact	0.667	0.400	0.987	0.500	5
Cystic	0.791	0.919	0.800	0.850	37
Macro avg	0.801	0.723	0.913	0.748	82
Weighted avg	0.858	0.854	0.886	0.850	82
Micro avg	0.854	0.854	0.927	0.854	82

Table A.98: SwinVit Montecarlo Dropout - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	34	0	0
Compact	0	2	3
Cystic	2	1	34

A.12.4 SwinUNETR

MC-Dropout Uncertainty Analysis – Predictive Entropy	
Correct predictions	0.2881 ± 0.2409
Incorrect predictions	0.5183 ± 0.1537
Uncertainty gap	0.2302

Per-class uncertainty		
Class 0	0.2020 ± 0.2135	$(n = 40)$
Class 1	0.5309 ± 0.2064	$(n = 5)$
Class 2	0.3983 ± 0.2225	$(n = 37)$

Selective prediction (rejection thresholds)			
Threshold	Condition	Kept	Accuracy
Reject top 50%	> 0.2665	41/82	1.0000
Reject top 25%	> 0.4677	61/82	0.9508
Reject top 10%	> 0.6761	73/82	0.9178
Reject top 5%	> 0.7253	77/82	0.9091

Calibration (accuracy by uncertainty bin)			
Bin	Range	Accuracy	n
Low (0–25%)	[0.0014, 0.0997)	1.0000	21
Medium (25–50%)	[0.0997, 0.2650)	0.9500	20
High (50–75%)	[0.2650, 0.4677)	0.9000	20
Very High (75–100%)	[0.4677, 1.0677)	0.7500	20

Table A.99: SwinUNETR Montecarlo Dropout - Testing - Accuracy 0.902

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	1.000	0.850	1.000	0.919	40
Compact	0.800	0.800	0.987	0.800	5
Cystic	0.837	0.973	0.844	0.900	37
Macro avg	0.879	0.874	0.944	0.873	82
Weighted avg	0.914	0.902	0.929	0.903	82
Micro avg	0.902	0.902	0.951	0.902	82

Table A.100: , SwinUNETR Montecarlo Dropout - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	34	0	6
Compact	0	4	1
Cystic	0	1	36

A.13 K-Fold Cross-Validation: Robustness assessment

A.13.1 DenseNet

K-Fold Cross-Validation Results		
	Mean ± Std	**Min / Max**
Train Accuracy	0.6422 ± 0.0196	$0.6196 / 0.6774$
Validation Accuracy	0.9126 ± 0.0084	$0.9028 / 0.9236$
Train Loss	0.4696 ± 0.0339	$0.4429 / 0.5299$

Fold	**Train Acc**	**Val Acc**
1	0.6453	0.9172
2	0.6774	0.9028
3	0.6389	0.9236
4	0.6196	0.9028
5	0.6297	0.9167

Table A.101: , Densenet K-Fold - Testing - Accuracy 0.878

Class	**Precision**	**Recall**	**Specificity**	**F1**	**Support**
Cauliflower	1.000	0.800	1.000	0.889	40
Compact	0.800	0.800	0.987	0.800	5
Cystic	0.800	0.973	0.800	0.878	37
Macro avg	0.867	0.858	0.929	0.856	82
Weighted avg	0.898	0.878	0.909	0.879	82
Micro avg	0.878	0.878	0.939	0.878	82

Table A.102: , Densenet K-Fold - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	32	0	8
Compact	0	4	1
Cystic	0	1	36

A.13.2 ResNet18

K-Fold Cross-Validation Results		
	Mean ± Std	**Min / Max**
Train Accuracy	0.9605 ± 0.0112	$0.9391 / 0.9711$
Validation Accuracy	0.9459 ± 0.0173	$0.9167 / 0.9655$
Train Loss	0.0851 ± 0.0173	$0.0698 / 0.1075$

Fold	Train Acc	Val Acc
1	0.9628	0.9655
2	0.9622	0.9375
3	0.9391	0.9514
4	0.9674	0.9167
5	0.9711	0.9583

Table A.103: , Resnet18 K-Fold - Testing - Accuracy 0.902

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	1.000	0.850	1.000	0.919	40
Compact	0.800	0.800	0.987	0.800	5
Cystic	0.837	0.973	0.844	0.900	37
Macro avg	0.879	0.874	0.944	0.873	82
Weighted avg	0.914	0.902	0.929	0.903	82
Micro avg	0.902	0.902	0.951	0.902	82

Table A.104: , Resnet18 K-Fold - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	34	0	6
Compact	0	4	1
Cystic	0	1	36

A.13.3 SwinVit

K-Fold Cross-Validation Results		
	Mean ± Std	**Min / Max**
Train Accuracy	0.7840 ± 0.1664	$0.4558 \,/\, 0.9053$
Validation Accuracy	0.8113 ± 0.1631	$0.4861 \,/\, 0.9028$
Train Loss	0.4330 ± 0.3308	$0.2211 \,/\, 1.0924$

Fold	Train Acc	Val Acc
1	0.8853	0.8690
2	0.8453	0.9028
3	0.9053	0.9028
4	0.8282	0.8958
5	0.4558	0.4861

Table A.105: , SwinVit K-Fold - Testing - Accuracy 0.866

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.921	0.875	0.929	0.897	40
Compact	0.500	0.600	0.961	0.545	5
Cystic	0.868	0.892	0.889	0.880	37
Macro avg	0.763	0.789	0.926	0.774	82
Weighted avg	0.872	0.866	0.913	0.868	82
Micro avg	0.866	0.866	0.933	0.866	82

Table A.106: , SwinVit K-Fold - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	35	1	4
Compact	1	3	1
Cystic	2	2	33

A.13.4 SwinUNETR

K-Fold Cross-Validation Results		
	Mean ± Std	**Min / Max**
Train Accuracy	0.9265 ± 0.0085	0.9143 / 0.9395
Validation Accuracy	0.8988 ± 0.89880159	0.8759 / 0.9167
Train Loss	0.1639 ± 0.0099	0.1468 / 0.1758

Fold	Train Acc	Val Acc
1	0.9266	0.8759
2	0.9217	0.9167
3	0.9308	0.8958
4	0.91433	0.9167
5	0.9395	0.8889

Table A.107: , SwinUNETR K-Fold - Testing - Accuracy 0.878

Class	Precision	Recall	Specificity	F1	Support
Cauliflower	0.972	0.875	0.976	0.921	40
Compact	0.571	0.800	0.961	0.667	5
Cystic	0.846	0.892	0.867	0.868	37
Macro avg	0.797	0.856	0.935	0.819	82
Weighted avg	0.891	0.878	0.826	0.882	82
Micro avg	0.878	0.878	0.939	0.878	82

Table A.108: , SwinUNETR K-Fold - Testing - Confusion Matrix

	Predicted		
Actual	Cauliflower	Compact	Cystic
Cauliflower	35	0	5
Compact	0	4	1
Cystic	1	3	33

Bibliography

[1] Ruth Lehmann et al. “Human organoids: a new dimension in cell biology”. In: *Molecular Biology of the Cell* (2019) (cit. on pp. 1, 2).

[2] Sixiang Yang et al. “Organoids: The current status and biomedical applications”. In: *MedComm (2020)* (2023) (cit. on pp. 1, 2).

[3] Anne C. Rios and Hans Clevers. “Imaging organoids: a bright future ahead”. In: *Nature Methods* (2018) (cit. on pp. 2, 5).

[4] Aurélie Lacouture, Camille Lafront, Cindy Peillex, Martin Pelletier, and Étienne Audet-Walsh. “Impacts of endocrine-disrupting chemicals on prostate function and cancer”. In: *Environmental Research* (2022) (cit. on pp. 2, 3).

[5] Giuliana Rossi, Andrea Manfrin, and Matthias P. Lutolf. “Progress and potential in organoid research”. In: *Nature Reviews Genetics* (2018) (cit. on p. 4).

[6] Andrew D. Elliott. “Confocal Microscopy: Principles and Modern Practices”. In: *Current Protocols in Cytometry* (2020) (cit. on pp. 4, 5).

[7] TransOrga-plus Team. “A knowledge-driven deep learning framework for organoid dynamics analysis”. In: *Nature Communications* (2025) (cit. on p. 6).

[8] “Digitalized organoids: integrated pipeline for high-speed 3D morphological analysis”. In: *Nature Methods* (2025) (cit. on p. 6).

[9] “VONet: A deep learning network for 3D reconstruction of organoid imaging”. In: *Cell Reports Methods* (2024) (cit. on p. 6).

[10] Elisabet Domènech-Moreno, Aleksandra Brandt, Teemu T. Lemmetyinen, Lotta Wartiovaara, Tomi P. Mäkelä, and Sami Ollila. “Tellu – an object-detector algorithm for automatic classification of intestinal organoids”. In: *Disease Models & Mechanisms* (2023) (cit. on p. 6).

[11] Sohaib et al. “Multi-Aperture Transformers for 3D (MAT3D) Segmentation of Clinical and Organoid Images”. In: *WACV* (2025) (cit. on p. 6).

[12] Chen et al. “3D TransUNet: Advancing Medical Image Segmentation through Vision Transformers”. In: *arXiv preprint arXiv:2310.07781* (2023) (cit. on p. 6).

[13] Xiaoke Shen. “A survey of Object Classification and Detection based on 2D/3D data”. In: *CoRR* abs/1905.12683 (2019) (cit. on p. 9).

[14] Jean Lahoud, Jiale Cao, Fahad Shahbaz Khan, Hisham Cholakkal, Rao Muhammad Anwer, Salman Khan, and Ming-Hsuan Yang. “3D Vision with Transformers: A Survey”. In: (2022) (cit. on pp. 9, 31).

[15] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: (2015) (cit. on p. 11).

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015) (cit. on pp. 12, 13).

[17] Sihong Chen, Kai Ma, and Yefeng Zheng. “Med3D: Transfer Learning for 3D Medical Image Analysis”. In: *CoRR* abs/1904.00625 (2019) (cit. on p. 13).

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017) (cit. on pp. 15, 16, 19, 30).

[19] José Maurício, Inês Domingues, and Jorge Bernardino. “Comparing Vision Transformers and Convolutional Neural Networks for Image Classification: A Literature Review”. In: *Applied Sciences* (2023) (cit. on pp. 15, 19, 31).

[20] Yaoli Wang, Yaojun Deng, Yuanjin Zheng, Pratik Chattopadhyay, and Lipo Wang. “Vision Transformers for Image Classification: A Comparative Survey”. In: *Technologies* (2025) (cit. on pp. 19, 20, 30).

[21] Chaima Ben Rabah, Ioannis N. Petropoulos, Rayaz A. Malik, and Ahmed Serag. “Vision transformers for automated detection of diabetic peripheral neuropathy in corneal confocal microscopy images”. In: *Frontiers in Imaging* () (cit. on p. 19).

[22] Qiumei Pu, Zuoxin Xi, Shuai Yin, Zhe Zhao, and Lina Zhao. “Advantages of transformer and its application for medical image segmentation: a survey”. In: (2024) (cit. on pp. 19, 27).

[23] 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”. In: *CoRR* abs/2103.14030 (2021) (cit. on pp. 20, 23, 31).

[24] Ali Hatamizadeh, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett Landman, Holger Roth, and Daguang Xu. “UNETR: Transformers for 3D Medical Image Segmentation”. In: (2021) (cit. on pp. 25, 26).

[25] Ali Hatamizadeh, Vishwesh Nath, Yucheng Tang, Dong Yang, Holger Roth, and Daguang Xu. “Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images”. In: (2022) (cit. on pp. 25, 28).

[26] Muhammad Sohaib, Sepideh Shabani, Shadan Allah Mohammed, and Bahram Parvin. “3D-Organoid-SwinNet: High-Content Profiling of 3D Organoids”. In: *IEEE Journal of Biomedical and Health Informatics* (2025) (cit. on p. 29).

[27] Hee E. Kim, Alejandro Cosa-Linan, Nandhini Santhanam, Mahboubeh Jannesari, Mate E. Maros, and Thomas Ganslandt. "Transfer learning for medical image classification: a literature review". In: *BMC Medical Imaging* (2022) (cit. on p. 33).

[28] Zhaoyi Yang, Xueying Luo, Yawen Li, Di Wu, Yutong Zhang, Siyuan Chen, Weidong Cai, and Yefeng Zhang. "MedMNIST v2: A large-scale lightweight benchmark for 2D and 3D biomedical image classification". In: *Medical Image Analysis* (2023) (cit. on pp. 33, 34).

[29] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: (2016) (cit. on p. 35).

[30] Andreanne Lemay et al. "Improving the repeatability of deep learning models with Monte Carlo dropout". In: *npj Digital Medicine* (2022) (cit. on p. 35).

[31] Andréanne Lemay, Katharina Hoebel, Christopher P. Bridge, et al. "Improving the repeatability of deep learning models with Monte Carlo dropout". In: *npj Digital Medicine* () (cit. on p. 36).

[32] Hee E. Kim, Alejandro Cosa-Linan, Nandhini Santhanam, Mahboubeh Jannesari, Mate E. Maros, and Thomas Ganslandt. "Transfer learning for medical image classification: a literature review". In: *BMC Medical Imaging* () (cit. on p. 36).