Final Project: Lung Cancer Segmentation

Bui Dinh Lam	22BI13234
Dao Hoang Dung	22BI13101
Bui Dang Quang	22BI13378
Nguyen Minh Tuan	22BI13447

Abstraction

Lung cancer is the leading cause of cancer deaths worldwide. It is a type of cancer that often goes undetected until it has progressed to later stages.

This project aims to find a deep learning model capable of performing segmentation of dangerous lung nodules.

Problem Statement

Lung cancer in its initial stages often progresses silently, as symptoms do not manifest externally until the disease has significantly advanced. However, this condition is most effectively treated when detected early.

Statistically, individuals diagnosed with early-stage lung cancer have a significantly higher likelihood of surviving at least five years compared to those diagnosed at more advanced stages.

In this project, we aim to explore methods for early lung cancer detection before it becomes critical. We will implement deep learning models, particularly CNN (Convolutional Neural Networks) and ViT (Vision Transformers), to enhance segmentation and detection capabilities.

Lung Cancer Segmentation Dataset

We utilize the Lung Cancer Segmentation dataset with Lung-RADS classification. This dataset is a fusion of:

- Kazakhstani local data from the Kazakh Research Institute of Oncology and Radiology.
- The openly available LIDC-IDRI dataset, which has been re-labeled.

Dataset Division:

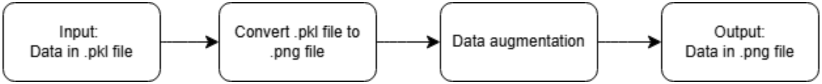
- **Training set:** 708 CT images
- **Test set:** 98 CT images (stored in pickle file format)

Preprocessing

Preprocessing Pipeline:

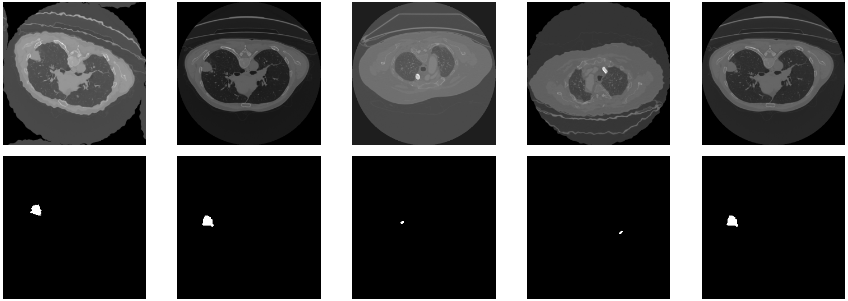
- Convert dataset from .pkl format to .png using NumPy.
- Apply data augmentation techniques:
  - Random rotation ($\pm$30 degrees)
  - Horizontal and vertical flips
  - Elastic transformations for non-rigid deformations
  - Brightness and contrast adjustments
- Resize all images to 256$\times$256 pixels for consistency.
- Final dataset size: **3,540 images** (original + augmented)

Preprocessing Pipeline



Preprocessing Pipeline

Augmented Image Examples


Augmented data

Swin U-Net

Overview:

- Hybrid model combining U-Net and Swin Transformer.
- Captures multi-scale features using Swin Attention (Shifted Window Attention).
- Efficiently models long-range dependencies for improved segmentation.

Attention Mechanism

Equation:

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

Process:

- Converts images into patches.
- Applies self-attention to capture relationships.
- Improves understanding of image regions.

Shifted Window Mechanism

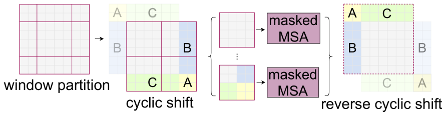


Figure: Cyclic Shifting Mechanism

Since using a global window mechanism can be computationally expensive, a local window mechanism is utilized for scalability. A local window attention operation operates within a window of $M \times M$ patches. But this wouldn't capture the cross-window context and padding the windows that surpass the boundary of the tensor would require padding meaning more operations to be computed.

- To avelliate the problem of being incapable of cross-window context, the next modules displace the window by $[M/2, M, 2]$ with the M being the patch size(in our case, 4).
- The padding problem can be solved with using cyclic shifting mechanism described in the figure above

Swin Transformer Block

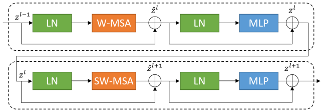
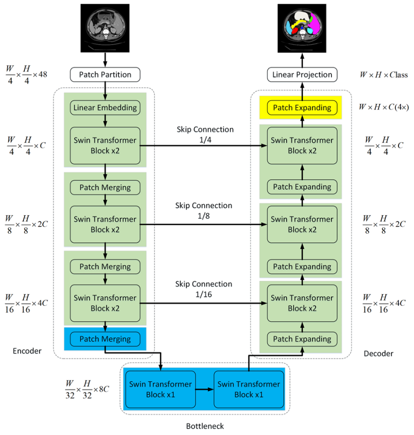


Figure: Swin transformer block

Overview: The Swin Transformer block is similar to the regular transformer's encoder block with several key differences:

- **Layer Normalization (LN):** Applied before tensor operations.
- **Self-Attention Mechanism:**
  - Uses Window Multi-Headed Self-Attention (W-MSA).
  - Includes Shifted Window Multi-Headed Self-Attention for better context capturing.
- **Skip Connections:** Adds previous tensor z_{l-1} with $\hat{z}_l$, continuing in deeper layers.
- **Final Processing:** Applies a 2-layer Multi-Layer Perceptron

Swin-UNet architecture



$\frac{W}{4}\times\frac{H}{4}\times 48$
Patch Partition
Linear Embedding
Swin Transformer Block x2
Patch Merging
Swin Transformer Block x2
Patch Merging
Swin Transformer Block x2
Patch Merging
$\frac{W}{4}\times\frac{H}{4}\times C$
$\frac{W}{8}\times\frac{H}{8}\times 2C$
$\frac{W}{16}\times\frac{H}{16}\times 4C$
Encoder
Skip Connection 1/4
Skip Connection 1/8
Skip Connection 1/16
Linear Projection
$W\times H\times Class$
Patch Expanding
$W\times H\times C(4\times)$
Swin Transformer Block x2
Patch Expanding
Swin Transformer Block x2
Patch Expanding
Swin Transformer Block x2
Patch Expanding
$\frac{W}{4}\times\frac{H}{4}\times C$
$\frac{W}{8}\times\frac{H}{8}\times 2C$
$\frac{W}{16}\times\frac{H}{16}\times 4C$
Decoder
$\frac{W}{32}\times\frac{H}{32}\times 8C$
Swin Transformer Block x1
Swin Transformer Block x1
Bottleneck

Figure: Swin-UNet architecture

- **Patch Embedding:**
  - The image is divided into patches.
  - A linear projection converts each patch into a feature vector of size 96.
  - This results in a tokenized representation of the image.
- **Encoder**
  - The Swin Transformer processes the tokenized image.
  - The model downsamples the image at scales of $1/4$, $1/8$, and $1/16$.
  - Feature channels increase to capture both fine details and large-scale structures.
- **Decoder:**
  - The decoder gradually upscales the image using patch expanding (upsampling).
  - Skip connections restore fine details from earlier stages.
  - This ensures a sharp and accurate segmentation.

UNETR: U-Net Transformer

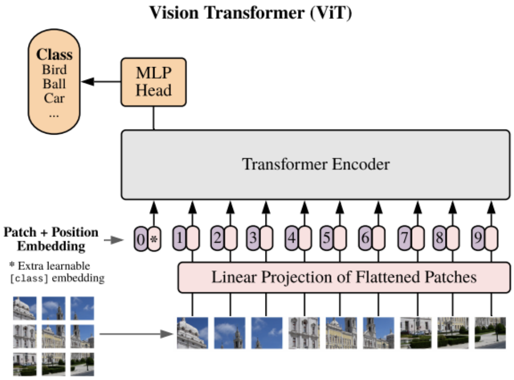
Overview:

- Transformer-based encoder with a U-Net-like decoder.
- Captures global contextual features.
- Uses multi-head self-attention (MSA) for feature extraction.

UNETR: Understanding Vision Transformer encoder

What is Vision Transformer ?

Vision Transformer (ViT) is an architecture that is used for image recognition. It is based on the Transformer architecture, which was originally developed for natural language processing. ViTs have been shown to achieve state-of-the-art results on a variety of image recognition tasks, including ImageNet classification.


Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0
*
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

Multi-Head Attention

What is Multi-Head Self-Attention?

Multi-head attention extends self-attention by splitting the input into multiple heads, enabling the model to capture diverse relationships and patterns. Instead of using a single set of Q, K, and V matrices, the input embeddings are projected into multiple sets (heads), each with its own Q, K, and V.

- **Linear Transformation:** The input X is projected into smaller-dimensional subspaces using different weight matrices:

  $$Q_i = XW_i^Q, \quad K_i = XW_i^K, \quad V_i = XW_i^V$$

  where i denotes the head index.
- **Independent Attention Computation:** Each head computes its own self-attention using the scaled dot-product formula.
- **Concatenation:** The outputs from all heads are concatenated.

Multi-Head Self-Attention

What is Multi-Head Self-Attention?

- **Final Linear Transformation:** A final weight matrix W_O is applied to transform the concatenated output into the desired dimension.

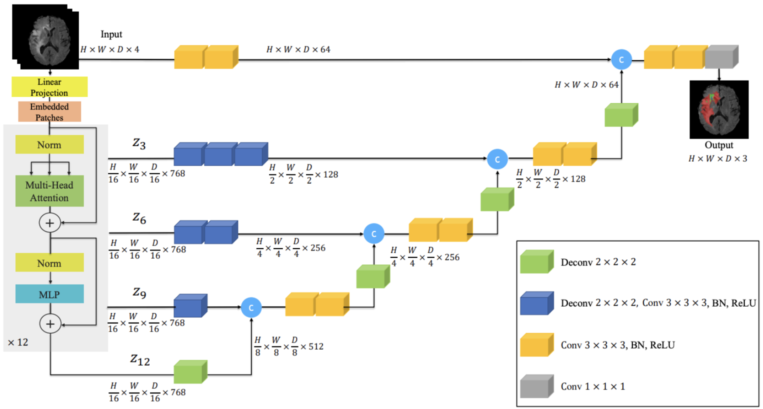
Mathematically, multi-head attention is expressed as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W_O$$

where:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

UNETR Architecture



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

Fig. 1. Overview of the UNETR architecture. We extract sequence representations of different layers in the transformer and merge them with the decoder via skip connections. Output sizes demonstrated for patch dimension $N = 16$ and embedding size $C = 768$.

UNETR Model Architecture

Figure: UNETR Architecture

- **Patch Embedding:**
  - The input image is divided into non-overlapping patches.
  - A linear projection embeds each patch into a 768-dimensional feature vector.
  - This tokenized representation is then passed into the Transformer encoder.
- **Transformer Encoder:**
  - Consists of 12 layers of self-attention blocks.
  - Processes the embedded patches and learns hierarchical feature representations.
  - Outputs multi-scale latent representations at different depths.
- **Decoder:**
  - The decoder progressively upsamples feature maps through deconvolutions.
  - Skip connections merge hierarchical features from the Transformer with the decoder.
  - Convolutional layers refine spatial details for accurate segmentation output.

UNETR Implementation

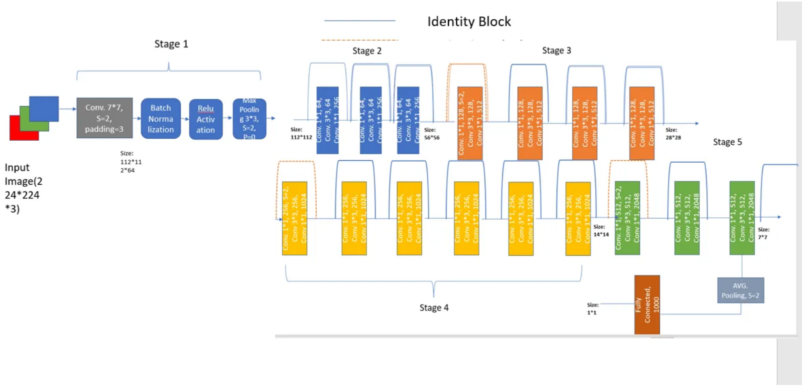
- Trained UNETR for lung cancer segmentation using an NVIDIA Tesla P100 GPU.
- Data split into training, validation, and test sets.
- Batch size: 8, Learning rate: 0.1, Optimizer: SGD.
- Trained for 100 epochs with callbacks:
  - **ModelCheckpoint:** Saved best model weights.
  - **ReduceLROnPlateau:** Adjusted learning rate when validation loss plateaued (min LR: $1e^{-7}$).
  - **EarlyStopping:** Stop training after 20 epochs without improvement.

ResNet50-U-Net

Overview:

- ResNet50-U-Net is an adaptation of U-Net that integrates a ResNet50 encoder.
- Utilizes pre-trained weights (e.g., ImageNet) for feature extraction.
- Enhances segmentation performance by leveraging residual learning.

ResNet50 Architecture



ResNet50 architecture

- ResNet50 is a 50-layer deep convolutional network.
- Addresses vanishing gradient problem using residual learning.
- Organized into five stages with convolutional and identity blocks.

Convolutional Layers in ResNet50

- Initial 7x7 convolutional layer captures low-level features.
- Followed by a 3x3 max pooling layer to reduce spatial dimensions.
- Convolutional and identity blocks enable deep feature extraction.

Residual Learning and Skip Connections

- Residual blocks allow input to bypass intermediate layers.
- Improves gradient flow and optimization in deep networks.
- Residual block formulation:

$$y = F(x) + x \tag{1}$$

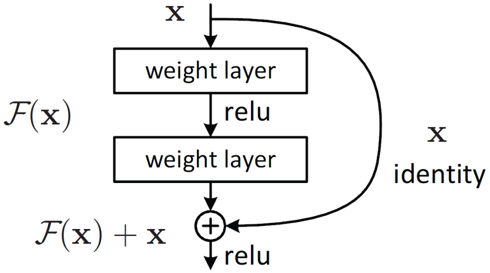


Figure: Basic Residual Block

U-Net Overview

- U-Net is a symmetric CNN for biomedical image segmentation.
- Encoder captures context, decoder enables precise localization.
- Skip connections combine spatial information with learned features.

ResNet50-U-Net Model Architecture

- ResNet50 is used as the encoder in the U-Net framework.
- Pre-trained ResNet50 enhances deep feature extraction.
- The decoder performs upsampling and segmentation refinement.

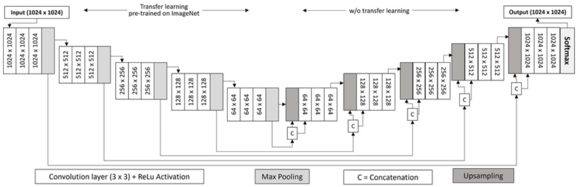


Figure: ResNet50-U-Net Architecture

Implementation Details

- Implemented using `segmentation_models_pytorch`.
- ResNet50 backbone initialized with ImageNet pre-trained weights.
- Loss function: Binary Cross Entropy with Logits Loss.
- Optimizer: Adam with initial learning rate 1×10^{-4}.
- Trained for 100 epochs on an NVIDIA RTX 3050 Laptop GPU.

Evaluation Metrics

We use three evaluation metrics to measure segmentation performance:

- **RMSE:** Measures error between predicted and actual values.
- **Dice Coefficient:** Measures similarity between predicted and ground truth masks.
- **IoU (Intersection over Union):** Evaluates object detection accuracy.

RMSE - Root Mean Square Error

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Purpose:

- Measures average prediction error.
- Lower RMSE indicates better performance.

Dice Coefficient

$$\text{Dice} = \frac{2 \times |A \cap B|}{|A| + |B|}$$

Purpose:

- Measures similarity between predicted and actual regions.
- Value ranges from 0 (no overlap) to 1 (perfect overlap).

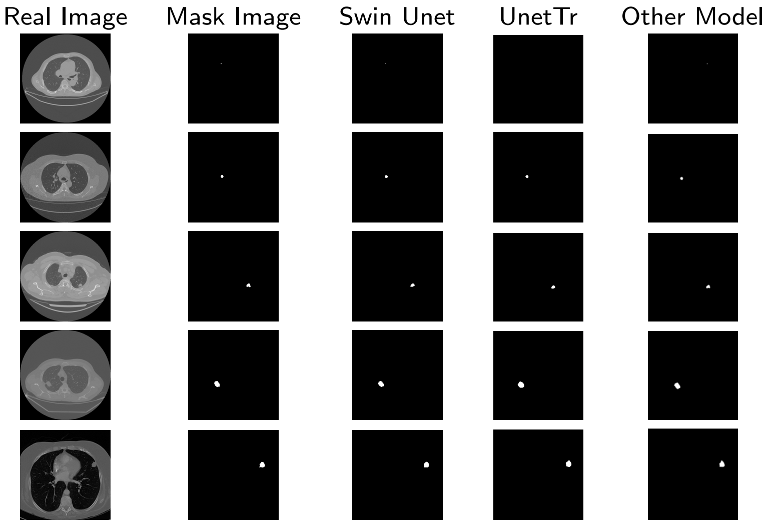
IoU - Intersection over Union

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|}$$

Purpose:

- Measures overlap between predicted and ground truth masks.
- Higher IoU indicates more accurate segmentation.

Result



Results

Future Works

- Our project still faces challenges, such as difficulty in capturing small tumors and accurately predicting tumor shapes.
- To enhance small tumor detection, integrating BioMedCLIP's vision encoder with MedSAM could be beneficial.
- BioMedCLIP's training on millions of medical images and captions grants it zero-shot capabilities.
- MedSAM, with its transformer-based structure, enhances attention to specific areas, making it effective for identifying small tumors.
- Combining these two models could significantly improve overall performance.
- Incorporating additional imaging modalities such as PET or MRI may provide more context on tumor location and behavior.

Thank You!

Thank You!

Any Questions?