Machine learning in medicine

University of science and technology of Ha Noi

Final project: Lung cancer segmentation

Students:
Bui Dinh Lam (ID: 22BI13234)
Dao Hoang Dung (ID: 22BI13101)
Bui Dang Quang (ID: 22BI13378)
Nguyen Minh Tuan (ID: 22BI13447)

1 Introduction

1.1 Abstraction

Lung cancer is the leading cause of cancer deaths in all over the world. It is a type of cancer that is often hard to be detected until it has progressed for later stages. This project aims to find a deep learning model that is capable of performing segmentation of dangerous lung nodules.

1.2 Problem statement

The Lung cancer in its initial stages often progresses silently, as symptoms do not show up externally until the disease has significantly worsen the patient's health. However, this symptoms is most easily and effectively treated when found at an early stage. Statistically, those with lung cancer caught early on have a much higher likelihood of surviving at least five years after diagnosis than those diagnosed when the lung cancer is more advanced.
In this project, we aim to find ways to detect lung cancer early, before it becomes too serious. We aim to implement deep learning in this project, particularly CNN(Convolutional Neural Network) and ViT(Vision Transformers).

1.3 Dataset

About the dataset, we use the Lung cancer segmentation dataset with Lung-RADS class [7]. This dataset is a fusion of original Kazakhstani local data from the Kazakh Research Institute of Oncology and Radiology, and the openly available LIDC-IDRI dataset, which has been re-labeled by experts. The entire dataset has been divided into training and testing sets, with the training set consisting of 708 CT images and the test set containing 264 CT images in pickle file.

2 Method

2.1 Preprocessing

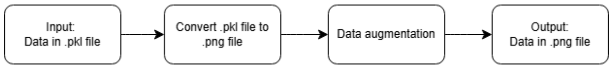

Figure 1: preprocessing pipeline

We begin by converting our dataset from .pkl format to .png using NumPy, ensuring compatibility with our model's standardized input.
After conversion, we apply various data augmentation techniques to enhance diversity and improve training performance. We apply these geometric augmentations include rotation (randomly rotating within ± 30 degrees), horizontal and vertical flips. Elastic transformation also being apply, which mimics non-rigid deformations commonly seen in medical imaging. Additionally, brightness and contrast adjustments introduce variations in intensity, improving the model's robustness to different imaging conditions.
The final preprocessed dataset consists of approximately 3,400 samples, including normal and masked images in .png format. Finally, all images are resized to 256×256 pixels for consistency.
Here are some example of the training set when we apply this techniques:

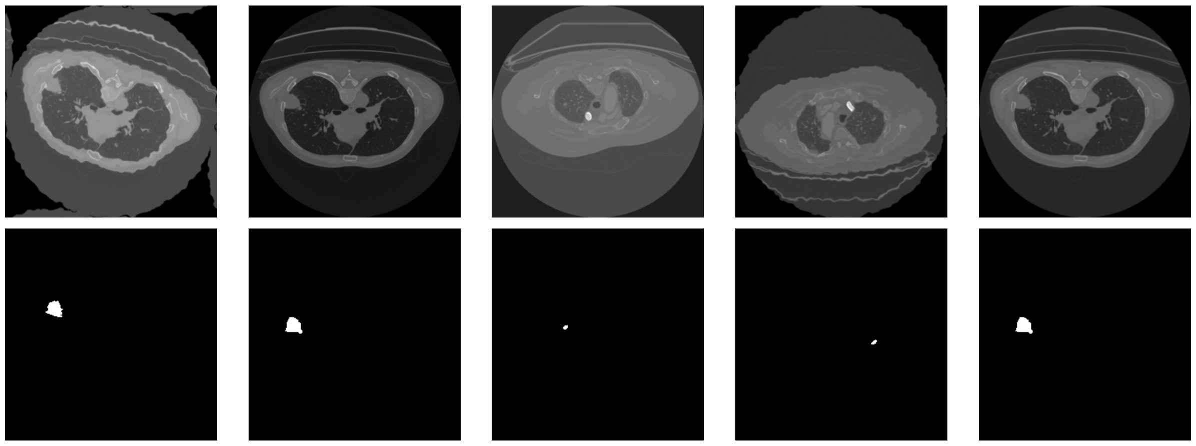

Table 1: Augmented Images

2.2 Swin U-net

Swin U-Net[1] is a combination of both U-Net and Swin Transformer, both of which were originally designed for semantic segmentation. The key idea behind both models is to capture features at different scales. Swin Transformer achieves this through a mechanism called Swin Attention (Shifted Window Attention), which helps the model efficiently model long-range dependencies while maintaining computational efficiency. This mechanism enables Swin U-Net to better capture contextual information and improve segmentation performance, especially on complex images.

2.2.1 Attention mechanism

The attention mechanism is a widely used technique within the world of natural language processing [8]. But recently, it has gained a lot of potential within the field of computer vision in a variety of tasks, such as object classification, object detection, and semantic segmentation. The mechanism can be generalized by the following equation:

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

Where: Q is the query matrix, K is the key matrix, V is the value matrix, d_k is the dimension of the key vectors. This mechanism allows the model to attend to different parts of the data to change to data representation based on the relevance of those parts. In natural language processing, the language data first gets converted into indexes through a process called tokenization, then it turned into embedding for further processing. In computer vision and our task, the images get divided into patches of MxM sizes then projected into a matrix of size C through a projection network

2.2.2 Shifted window mechanism

In our model, a modified version of the attention mechanism called shifted window attention is implemented. Using the original attention mechanism to process the entire image would require a large amount

of computing power due to having to process every single token at the same time. The author of the Swin Transformers[6] proposed a more efficient approach to this using a local attention mechanism. This method allows the model to be more scalable and computationally efficient.
The image first gets divided into MxM patches of size WxW, then the image is splitted into an even matter without any overlapping windows of size $N \times N$ patches, then the attention mechanism is applied to produce an image representation. But this would lead to a lack of connection between windows and reduce the ability of the model to understand the image's context. To solve this, the later module of the model shifts the window position over ($W/2$ x $W/2$).
But this would lead to windows with sizes smaller than $M \times M$, to alleviate this, cyclic shifting is applied. Where patches moving out of the window border are wrapped around, maintaining consistent window sizes and reducing computational resource needed.

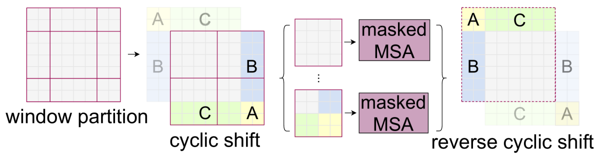

Figure 2: Cyclic shifting mechanism

Furthermore, to avoid the attention mechanism from attending to parts of images that are not connected to each other, a special masking procedure is implemented.

2.2.3 Swin transformer Block

The Swin Transformer block is summarized in Figure 3. Its architecture is similar

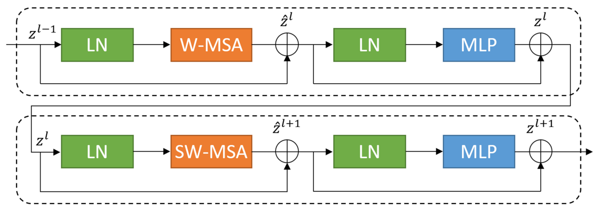

Figure 3: Cyclic shifting mechanism

to that of a standard transformer encoder block, except that the LayerNorm (LN) layer is placed before tensor operations. Furthermore, a noticable difference is that instead of two regular transformer layers with self-attention, the block uses window multi headed self-attention(W-MSA) along with shifted window multi-headed self-attention. For better context capturing, a skip connection is applied to add z_{l-1} with $\hat{z}_l$, similar operation is seen down the next stages. Finally, at the end of each child block, a 2 layer Multi-layer perceptron is applied with the activation function GELU.

2.2.4 Model's architecture

The model's architecture can be seen through figure 4 to be inspired from Unet, utilitizing various techniques like skip connections and bottleneck. After dividing the image into patches, the model uses a patch embedding layer (which includes a linear projection) to convert each patch into a feature vector of size C (in our case, 96). This results in a tokenized representation of the image, which is then processed further through the Swin Transformer blocks.

At each stage, the model makes the image smaller while increasing the number of

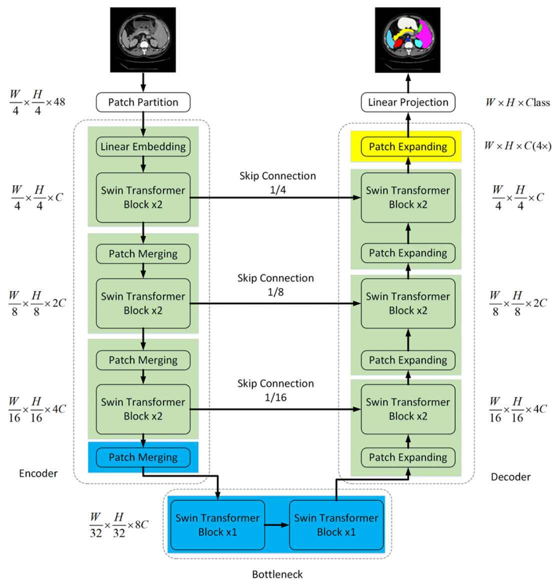
$\frac{W}{4}\times\frac{H}{4}\times 48$
Patch Partition
Linear Embedding
$\frac{W}{4}\times\frac{H}{4}\times C$
Swin Transformer Block x2
Skip Connection 1/4
Patch Merging
$\frac{W}{8}\times\frac{H}{8}\times 2C$
Swin Transformer Block x2
Skip Connection 1/8
Patch Merging
$\frac{W}{16}\times\frac{H}{16}\times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

Figure 4: Swin Unet architecture

feature channels, similar to how convolutional networks downsample images. This helps the model first capture fine details before gradually understanding the bigger picture. The model processes the image at different scales, typically 1/4, 1/8, and 1/16 of the original size, before reaching the bottleneck. Then, the decoder rebuilds the image step by step, combining information from different levels using skip connections to keep important details.

The model processes the image in multiple stages. As it goes deeper, it merges patches together to reduce the image size while increasing feature richness. This process, called patch merging, works like downsampling—it combines neighboring patches and increases the number of channels to retain important information. This helps the model first focus on small details before understanding larger patterns.

Later, in the decoder, the model needs to reconstruct the image. It does this through patch expanding, which is like upsampling it and gradually restores the image size while keeping the learned features. Along the way, skip connections help bring back fine details from earlier stages, ensuring a sharp and accurate reconstruction.

2.2.5 Implementation details

For Swin-Unet, we trained on the P100 gpu with 16gb of ram, with the famous Adam optimizer and a learning rate of 0.005. the image's input size is 224×224 with a patch size of 4 and embedding dimension of 96. Furthermore we also differs from the orgininal paper in our cost function, the original paper used an weighted ensemble loss function with Dice loss and Binary Cross-Entropy loss. But during our implementation it couldn't converge unlike the use of Binary Cross-Entropy loss. And to avoid excessive training, a simple early stopping mechanism is utilized, where if the model does not improve on the validation set after 10 times after surpassing a minimum amount of training epochs of 100 epochs.

2.3 UNETR: Unet TRansformers

UNet Transformers (UNETR)[2] is a deep learning architecture that combines the strengths of the UNet model and Transformer-based self-attention mechanisms for medical image segmentation. Unlike the traditional UNet model that relies on convolutional neural networks (CNNs) for feature extraction, UNETR follows a contraction-expansion structure, where a series of Transformer layers becomes the encoder, which is linked to the decoder through skip connections.

2.3.1 Embedding and Attention Mechanisms in UNETR

Similar to their use in natural language processing (NLP), Tranformers in UNETR process a 1D sequence of input embeddings. For further processing, a 3D input volume with dimensions (H, W, D) and C channels is divided into non-overlapping, flattened patches of size (P, P, P), forming a 1D sequence of length $N = HWD/P^3$. Each patch is then turned into a K-dimensional embedding vector using a linear layer, ensuring a consistent representation across all Transformer layers. To keep positional information, a learnable 1D positional embedding E_{pos} is added to the projected patch embeddings, ensuring that spatial relationships are preserved. Mathematically, this process is defined as follows:

$$z_0 = [x_1E; x_2E; \ldots; x_NE] + E_{\text{pos}}$$

where each x_iE represents the embedded patches.

The sequence of embeddings produced by the Transformer backbone is processed using a stack of Transformer blocks, each consisting of multi-head self-attention (MSA) and multilayer perceptron (MLP) sublayers. These layers operate according to the following transformations:

$$z_i = \text{MSA}(\text{Norm}(z_{i-1})) + z_{i-1}, \quad i = 1, \ldots, L$$

$$z_i = \text{MLP}(\text{Norm}(z_i)) + z_i, \quad i = 1, \ldots, L$$

where Norm() represents layer normalization, MLP consists of two linear layers with GELU activation functions, i is the intermediate block index, and L is the total number of Transformer layers.
Each MSA sublayer contains n parallel self-attention (SA) heads, which are responsible for capturing global context in the input. Specifically, the SA block is a learnable function that maps query (q) representations to corresponding key (k) and value (v) representations in a sequence $z \in \mathbb{R}^{N \times K}$. The attention weights A are computed by measuring the similarity between elements within the sequence and their key-value pairs using:

$$A = \text{Softmax}\left(\frac{qk^T}{K_h}\right)$$

where $K_h = K/n$ serves as a scaling factor that maintains a constant number of parameters across different key values. Using the computed attention weights, the final output of the self-attention mechanism is given by:

$$\text{SA}(z) = Av$$

Here, v represents the values in the input sequence. The output of multi-head self-attention (MSA) is then computed as:

$$\text{MSA}(z) = [\text{SA}_1(z); \text{SA}_2(z); \ldots; \text{SA}_n(z)]W_{\text{msa}}$$

where $W_{\text{msa}} \in \mathbb{R}^{nK_h \times K}$ represents the trainable multi-headed parameter weights.

2.3.2 Feature Extraction and Integration with the Decoder

Inspired by U-Net-like architectures, where multi-resolution encoder features are combined with the decoder, this approach extracts sequence representations z_i (where $i = 3, 6, 9, 12$), each with a size of HWD/P^3. These representations are reshaped into tensors of size:

$$\frac{H}{P} \times \frac{W}{P} \times \frac{D}{P} \times K$$

where K represents the embedding size of the Transformer output.
As illustrated in Fig. 5, at each resolution, the reshaped feature tensors are projected back into the input space using consecutive $3 \times 3 \times 3$ convolutional layers, followed by normalization layers. This transformation ensures effective spatial reconstruction, enabling the decoder to generate high-resolution segmentation outputs.

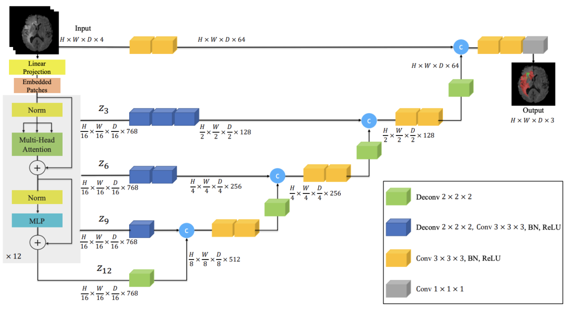
Input
$H \times W \times D \times 4$
$H \times W \times D \times 64$
Linear Projection
Embedded Patches
Norm
Multi-Head Attention
Norm
MLP
× 12
z_3
z_6
z_9
z_{12}
$\frac{H}{16} \times \frac{W}{16} \times \frac{D}{16} \times 768$
$\frac{H}{2} \times \frac{W}{2} \times \frac{D}{2} \times 128$
$\frac{H}{4}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

Figure 5: Unet Transformer architecture

2.3.3 Implementation details

In this task, we trained the UNETR (U-Net Transformer) model for lung cancer segmentation using the NVIDIA Tesla P100 GPU. The dataset was passed through a data loader to produce training, validation, and testing sets. We also set the batch size to 8, with an initial learning rate of 0.1, using the Stochastic Gradient Descent (SGD) optimizer, allowing stable convergence over 100 epochs.

To enhance the training process, we incorporated several callbacks. The ModelCheckpoint function ensured that the best-performing model was saved during training, preventing the loss of optimal weights. The ReduceLROnPlateau callback dynamically adjusted the learning rate when validation loss plateaued, and we set the learning rate will not be reduced below 1e-7 (0.0000001), ensuring that it does not become too small, which could cause the model to stagnate and stop learning. Futhermore, EarlyStopping stops training if there's no improvement for 20 consecutive epochs, preventing wasted time and resources.

2.4 ResNet50-Unet

The ResNet50-U-Net architecture is an adaptation of the traditional U-Net segmentation network that leverages a ResNet50 encoder pre-trained on large-scale image datasets (e.g., ImageNet). This design combines the representational power of residual networks with the effective up-sampling and skip-connection strategy of U-Net, making it particularly suitable for complex segmentation tasks in medical imaging and beyond.

2.4.1 ResNet50

(ResNet50)[3] is a deep residual network. composed of 50 layers, known for its exceptional ability to learn hierarchical feature representations in deep architectures. It was introduced by (He et al. in 2016) as a solution to the vanishing gradient problem, which is a major challenge when training very deep convolutional neural networks (CNNs). Unlike traditional CNNs, where the performance degrades as the network depth increases, ResNet50 introduces "residual learning" to enable effective training of very deep networks

In the ResNet50-U-Net architecture, ResNet50 serves as the encoder. The encoder is responsible for extracting meaningful features. It progressively downsamples the image, creating a hierarchy of feature maps that capture different levels of information. The initial layers extract low-level features such as edges and textures, while deeper layers capture high-level features. In our task, the ResNet50 encoder is initialized with weights pre-trained on large datasets like ImageNet. This transfer learning technique leverages the knowledge learned from these datasets, enabling the model to learn more effectively with smaller task-specific datasets, which is particularly useful in medical imaging where data can be limited.

2.4.2 Residual Learning and Skip Connections

The key mechanism in ResNet50 is the use of **residual blocks**, which allow the input of a layer to bypass multiple intermediate layers through **skip connections**. This structure ensures that the network learns residual mappings instead of attempting to directly map inputs to outputs. A residual block can be represented as:

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

where $F(x)$ represents the transformation applied by a set of convolutional layers, and x is the original input. This addition of the input directly to the output enables the network to retain critical information and improves gradient flow during backpropagation, making it easier to optimize deep networks[3].

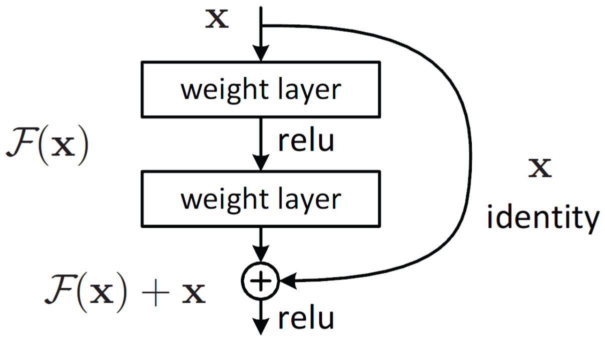
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 6: Basic Residual Block

2.4.3 Model Architecture

The U-Net's encoder-decoder structure with skip connections enables precise and accurate segmentation. In this combined architecture, the ResNet50 encoder replaces the standard encoder in the original U-Net architecture. The output feature maps from different stages of the ResNet50 encoder are then passed to the decoder part of the U-Net. The decoder then upsamples these features and combines them with the corresponding feature maps from the encoder via skip connections. This allows the model to effectively utilize the strong feature extraction capabilities of ResNet50 while maintaining the precise localization accuracy of U-Net.

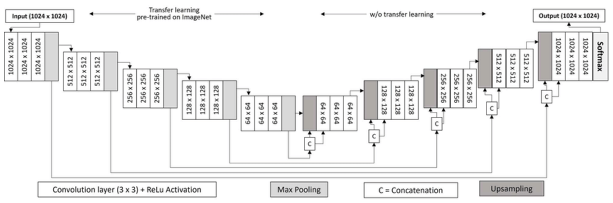

Figure 7: ResNet50-U-Net Architecture

2.4.4 Implementation Details

For the segmentation task, we implemented the U-Net architecture with a ResNet50 encoder using the `segmentation_models_pytorch` library [9]. The ResNet50 backbone is initialized with ImageNet pre-trained weights to leverage robust feature extraction, and the network is configured to accept 3-channel input images while producing a single-channel binary segmentation mask. The loss function used is Binary Cross Entropy with Logits Loss (`BCEWithLogitsLoss`), which is well-suited for binary segmentation problems, and the network parameters are optimized using the Adam optimizer with an initial learning rate of 1×10^{-4}. The model was trained for 100 epochs on an NVIDIA RTX 3050 Laptop GPU (4GB VRAM). The training loop processes batches of images and masks, computes the loss, performs backpropagation, and updates the model parameters; after each epoch, the model is evaluated on the validation set and relevant metrics are recorded.

3 Evaluation metrics

We use 3 types of metrics RMSE, Dice-Sørensen coefficient and IoU to evaluate our model.

3.1 RMSE

Root mean square errors (or RMSE) is the standard deviation of the residuals, or the average difference between the projected and actual values produced by a statistical model. The formula of this metric is:

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

3.2 Dice coefficient

The Dice coefficient(or Dice-Sørensen coefficient) is a measure of the similarity between two sets, A and B. The coefficient ranges from 0 to 1, where 1 indicates that the two sets are identical, and 0 indicates that the two sets have no overlap. The formula for this metric is:

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

3.3 IoU

Intersection over Union (IoU) is used to evaluate the performance of object detection by comparing the ground truth bounding box to the predicted bounding box. A higher IoU value indicate the better alignment between predicted and actual regions showing that the model is more accurate. The formula for this metric is:

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

4 Results

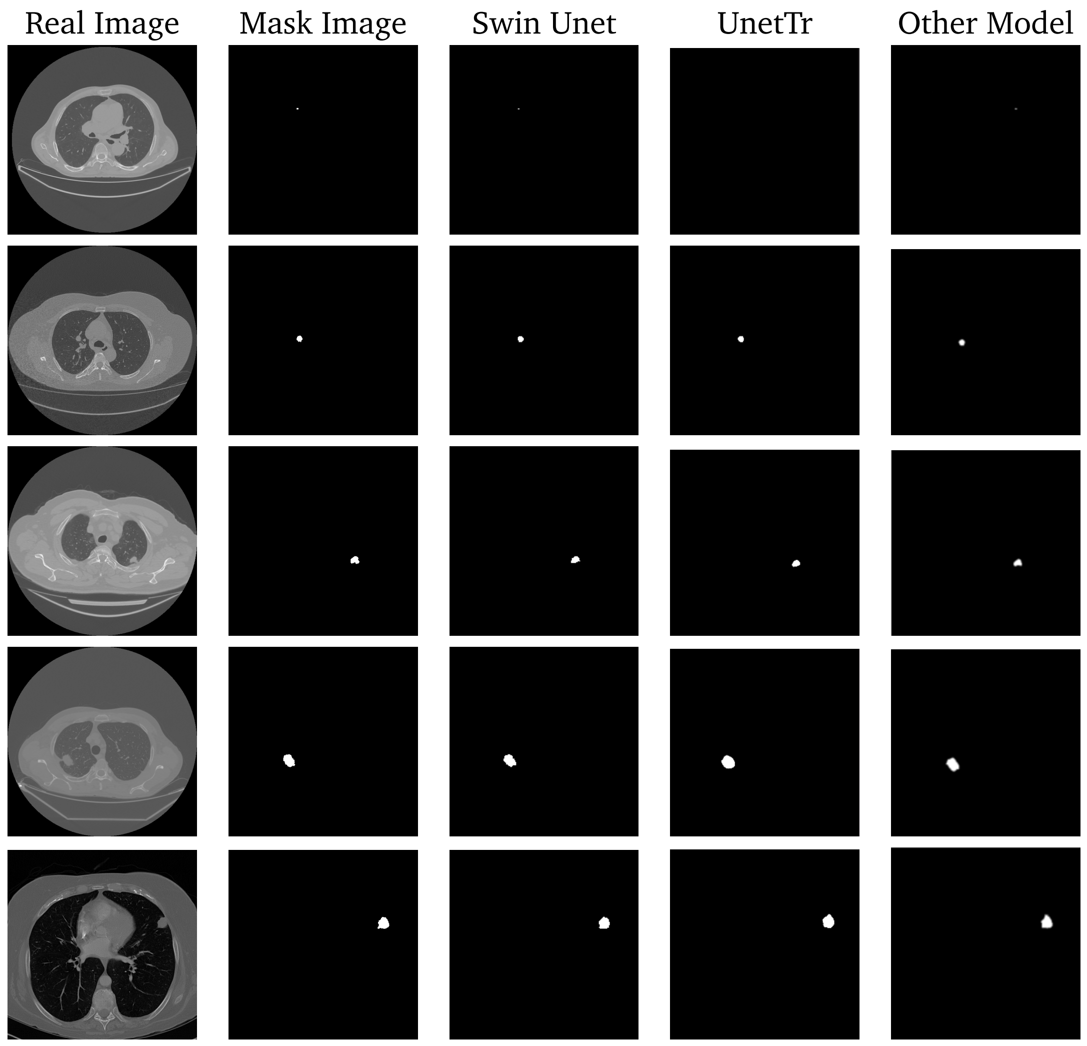

Table 2: Results

Model	Dice Coefficient	IoU	RMSE	Recall
SwinUNET	0.7969	0.7143	0.01193	0.7927
UNETR	0.8063	0.6772	0.0223	0.7727
ResNet50-Unet	0.7558	0.6754	0.0147	0.7760

Table 3: Metrics and Model Performance

Model	Dice Coefficient	IoU	RMSE
Unet[5]	0.862	-	-
RRc-UNet [4]	0.8777	0.7274	-

Table 4: Comparison of other's work based on Dice Coefficient, IoU, and RMSE.

5 Future works

In our project, a myriad of problems still exist, such as not being capable of capturing small tumors or uncapable of accurately predict the shape of the tumor. To better detect small tumors in lung cancer imaging, combining BioMedCLIP's vision encoder with MedSAM could help. BioMedCLIP's training on millions of medical images and captions allows it to have zero-shot capabilities. MedSAM uses a transformer structure to focus on certain areas, making it good at spotting tiny tumors. Combining these two models could significantly improve the overall performance. Adding other scan types (like PET or MRI) could also help by giving more clues about the tumor's location and behavior.

References

[1] Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. Swin-unet: Unet-like pure transformer for medical image segmentation. In Leonid Karlinsky, Tomer Michaeli, and Ko Nishino, editors, *Computer Vision – ECCV 2022 Workshops*, pages 205–218, Cham, 2023. Springer Nature Switzerland. pages 3

[2] Ali Hatamizadeh, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett Landman, Holger Roth, and Daguang Xu. Unetr: Transformers for 3d medical image segmentation, 2021. pages 5

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. pages 7, 8

[4] Van-Linh Le and Olivier Saut. Rrc-unet 3d for lung tumor segmentation from ct scans of non-small cell lung cancer patients. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 2308–2317, 2023. pages 10

[5] Ai-Hsien Adam Li, Cristian Daniel Aruperes, Yen-Jun Lai, Ting-Ying Chien, Yen-Ling Chiu, and Chien-Lung Chan. Lung nodule analysis in ct images: Deep learning for segmentation and measurement. In *Proceedings of the 2024 8th International Conference on Medical and Health Informatics*, ICMHI '24, page 13–17, New York, NY, USA, 2024. Association for Computing Machinery. pages 10

[6] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. pages 4

[7] Diana Nam. Lung cancer segmentation dataset with Lung-RADS class, March 2024. pages 2

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. pages 3

[9] Pavel Yakubovskiy. Segmentation models pytorch, 2019. https://github.com/qubvel/segmentation$_{m}odels.pytorch.pages$ 9