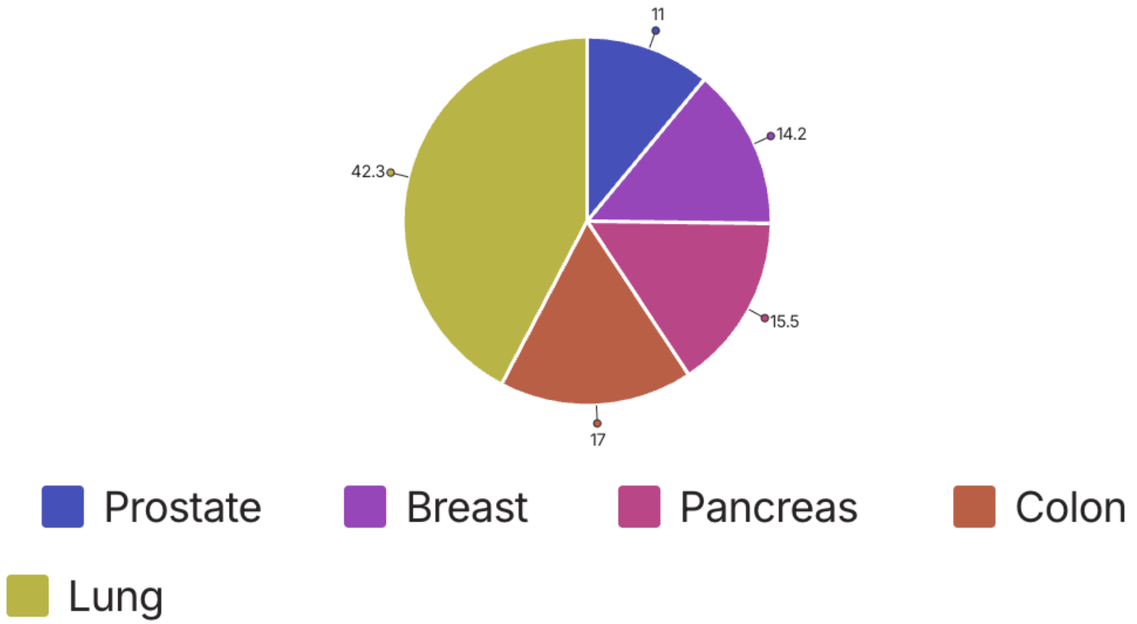
Lung Cancer Detection Using Super-Resolution Stacking

Authors: Abhilash, Dhruv, Rishaan

Problem Statement

Motivation

Leading type of cancer deaths in the US:



- Lung cancer causes 1.8 million deaths each year (WHO, 2023).
- Detection at an early stage increases survival chances by up to 20%.
- Manual segmentation = slow, inconsistent
- Need for faster, more accurate tools

Why Lung Nodule Segmentation is Difficult?

01

Boundary Detection Issues

The edges blend in, making it hard to detect clear contours

02

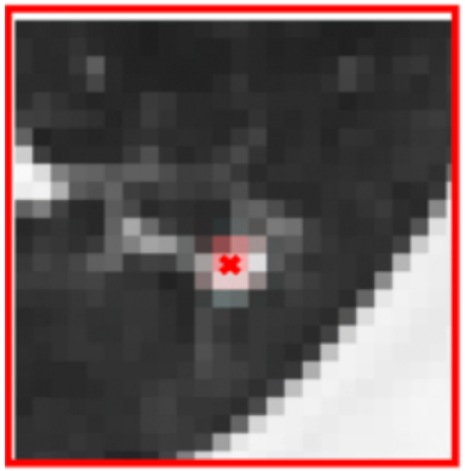
Dimensional Limitations

Most ML models rely on 2D slices and single-resolution data, which fail to capture volumetric and contextual information.

03

Nodules vary in shape, size, location

CT image interpretation is heavily dependent on radiologist skill and subjective judgment.



Fuzzy Nodule Boundary

Potential Applications

1. **Clinical Decision Support**
2. **Lung Cancer Screening**
3. **Deployable CAD Tools**

Project Goals

Accuracy

Robust segmentation with multi-resolution CT slices

Efficiency

Improve contour precision using super-resolution techniques

Robustness

Evaluate model generalizability and reproducibility

Literature Survey

Evolution of Lung Nodule Detection

1

Manual (Pre-1980s)

Radiologist-driven image analysis

2

CAD & Rule-Based (80s–00s)

Heuristic methods, edge detection

3

Deep Learning Today

CNNs, U-Nets, hybrid architectures

Manual Segmentation Issues

Inter-Observer Variability	High
Consistency	Low
Time Required	Extensive

CAD vs. CNN

Traditional CAD

- Rule-based filters
- High false positives (15-40%)
- Weak generalization
- Fuzzy edges

Deep Learning CNNs

- Data-driven feature learning
- Lower false positives (5-10%)
- Strong generalization
- Precise boundary detection

CAD vs. CNN in Lung Nodule Segmentation

Metric	SBF (Traditional CAD)	U-Net (CNN)
Dice Similarity Coefficient (DSC)	66.30%	83%
Sensitivity	77.5%	84%
Specificity	82.7%	94%
False Positive Rate (FPR)	21%	10%
Processing Time	Longer	Shorter

Evolution of CNNs in Medical Image Segmentation

1

2012: AlexNet

Introduced deep learning to image analysis using 2D CNNs.

Operates on individual CT slices, with low volumetric context.

2

2015: 2D U-Net

Revolutionized medical image segmentation with efficient 2D networks.

Computationally light but lacks depth information.

3

2017: 3D U-Net

Expanded filters to 3D volumes, capturing spatial context across slices.

Improved boundary consistency and anatomical accuracy.

Base Model: 2D U-Net

Encoder

Bottleneck

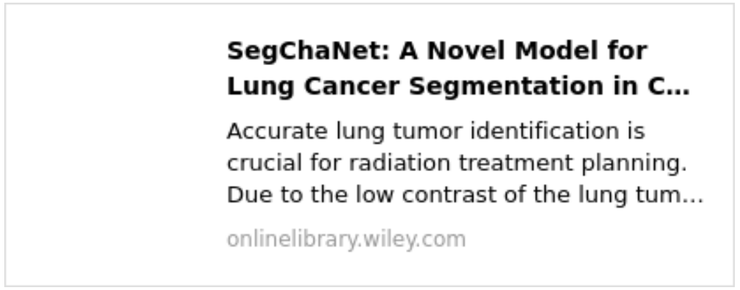
Decoder

2D CNN vs. 3D CNN for Stacked Multi-Resolution CT Input

Aspect	2D CNN (U-Net)	3D CNN (3D U-Net)	Why 2D
Input Format	2D slices or 3-channel images (e.g. 512×512×3)	Volumetric cubes (e.g. 64×512×512)	Our stacked inputs are different resolution slices of the same anatomical plane, not 3D
Data Compatibility	Works with heterogeneous slice thickness inputs	Requires consistent volumetric spacing across all slices	RIDER dataset includes variable slice thickness; 2D can handle this with stacking.
Computational Requirements	Low GPU memory, faster training	High GPU memory, longer training	2D is feasible on standard GPUs; 3D requires premium hardware or strong compute.
Use Case Fit	Ideal for per-slice or stacked-feature analysis	Best for continuous spatial analysis across slices	We're enhancing single slices with stacked resolution views, not doing time-series 3D

Author	Technique	Accuracy
2D methods		
Kalinovsky et al. (2017)	DCNN, GoogLeNet, AlexNet	88.7%
Liu et al. (2018)	Mask R-CNN, ResNet, FPN	
Serj et al. (2018)	DCNN	
Zhang et al. (2018)	U-Net	
Bhatia et al. (2019)	ResNet, XBG Boost, Random Forest	84%
Christe et al. (2019)	CNN, Clustering, Fast-marching method	
Madan et al. (2019)	CNN	93%
Park and Monahan (2019)	Genetic encoding, CNN	97.15%
Kalaivani et al. (2020)	CNN, ADABOOST	90.85%
Meraj et al. (2021)	CNN	
Angeline et al. (2022)	VGG-16, FCN	78.87%
Cui et al. (2022)	U-Net, VGG-Net	
Salama et al. (2022)	Generative model	98.91%
Shimazaki et al. (2022)	CNN	
Gunasekaran (2023)	YOLOv5	
Riaz et al. (2023)	MobileNetv2, U-Net	
3D methods		
Baek et al. (2019)	U-Net	
Ozdemir et al. (2019)	V-Net	
Sasikala et al. (2019)	CNN	96%
Bansal et al. (2020)	ResNet	
Kamal et al. (2020)	DenseNet, U-Net	
Borrelli et al. (2022)	U-Net	
Kasinathan et al. (2022)	Active Contour Model, CNN	97.1%
Dunn et al. (2023)	iMRRN	95.65%
Said et al. (2023)	UNETR	97.83%

2D and 3D methods both give similar accuracy scores, and choosing the type of model depends on the dataset, methodolgy, and type of output.



https://indjst.org/articles/a-novel-3d-multi-layer-convolutional-neural-networks-for-lung-cancer-segmentation-in-ct-images

Study Aim
Develop two deep learning models for early lung cancer detection:
TransSegNet for CT scan segmentation
MinClassNet for histopathological image classification
Datasets Used
ACDC-LungHP for segmentation
Public histopathology dataset for classification
Models
TransSegNet: Transformer-based segmentation (44 patches, 4 blocks)
MinClassNet: Lightweight 7-layer CNN for classification
Features
Automatically learned (no handcrafted features)
Metrics
TransSegNet:
Accuracy: 99.62%, Mean IoU: 49.8
MinClassNet:
Accuracy: 98.39%
Key Findings
Transformers improved segmentation accuracy
CNN improved classification over older methods
Both models show strong potential for clinical lung cancer detection

Aim of the Study
Develop a 3D Multi-Layer CNN combined with K-Means pre-segmentation for accurate and fast lung tumor segmentation in CT scans.
Dataset
TCIA (3D CT scans)
Manual + K-Means-enhanced labeling
Model Overview
3D-MLCNN with encoder-decoder architectures
Features Extracted
3D texture, intensity, edge, and depth-aware features
Feature fusion at root node for better spatial context
Performance Metrics
Accuracy: 98%
Dice: 0.921
IoU: 0.842
Sensitivity: 0.894

Identifying Lung Cancer Using Image Processing Techniques” by Disha Sharma & Gagandeep Jindal

Aim of the Study
Develop a rule-based CAD system for early lung cancer detection from CT scans by segmenting lungs, detecting nodules, and classifying them as benign or malignant based on handcrafted features.
Dataset
NIH/NCI LIDC (1,000 CT images)
Public, standardized, DICOM format
Techniques Used
CAD
Features Extracted
Nodule size & shape
Contrast enhancement
Calcification patterns
3D location & geometry
Performance Metrics
Sensitivity: 90%

https://www.sciencedirect.com/science/article/abs/pii/S0010482517300926?via%3Dihub

Aim of the Study
Compare deep learning models (CNN, DBN, SDAE) vs traditional CADx systems with handcrafted features for lung cancer diagnosis using CT images.
Dataset Used
LIDC-IDRI
Models Used
CNN (best performance)
DBN
SDAE
Traditional CADx with handcrafted features
Features
Deep models: Learned spatial, textural, and semantic features
CADx: GLCM, wavelets, intensity, shape, calcification
AUC
CNN: 0.899 ± 0.018
CADx: 0.848 ± 0.026
Key Findings
CNN outperforms CADx in diagnostic accuracy
Deep models capture more meaningful, generalizable features
Highlights potential of DL to replace handcrafted systems in CAD

Literature Gaps in Lung Cancer Detection

1. Bridging the Resolution Gap

Most deep learning models (e.g., basic U-Net, TransSegNet, even 3D CNNs) operate on single-resolution input, either 2D slices or 3D volumes of fixed thickness.

This ignores valuable anatomical detail available in multi-resolution CT scans (e.g., RIDER dataset has 1.25mm, 2.5mm, 5mm).

2. Avoiding 3D CNN Limitations

3D CNNs (e.g., 3D-MLCNN) are accurate but require huge computational power, aligned volumetric data, and lots of memory.

They also perform poorly on small datasets and struggle with variable slice thickness.

3. Addressing Label Inconsistencies

In LIDC or ACDC datasets, inter-observer variability exists — leading to noisy masks.

4. Volume-Inspired Accuracy with 2D-Efficiency

2D CNNs alone (e.g., the GFG and MinClassNet models) miss cross-slice consistency (nodules appearing in several adjacent slices).

Our Novel Approach

Multi-Resolution Stacking

We stack multi-resolution slices (thin + thick) into a 3-channel input.

This allows the model to see both fine nodule edges (from thin slices) and broader anatomical context (from thicker slices).

Enables resolution-aware learning without needing 3D convolutions.

2D CNN with Stacked Channels

We use a 2D CNN with stacked channels, which:

Works with unaligned or semi-aligned data.

Requires less GPU memory.

Trains faster and is easier to debug and visualize.

Consistent Annotations

We only use paired test-retest scans with consistent radiologist annotations (RIDER dataset).

This minimizes label noise and gives us stable segmentation ground truth.

Pseudo-3D Awareness

Our stacked slices are from the same anatomical location at different resolutions.

This simulates a pseudo-3D awareness using 2D models — improving boundary prediction and minimizing misclassification of edge slices.

Dataset – RIDER Lung CT

FileName	PatientID	Modality	StudyDate	StudyTime	Manufacturer	SliceThickness	PixelSpacing	ImageOrientationPatient	ImagePositionPatient
1-1.dcm	RIDER-3160137230	SEG	20070310	120351	pydicom-seg				
1-128.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, -138.75]
1-198.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, -226.25]
1-193.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, -220]
1-007.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, 12.5]
1-086.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, -86.25]
1-181.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, -205]
1-142.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, -156.25]
1-129.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, -140]
1-201.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, -230]
1-154.dcm	RIDER-3160137230	CT	20070310	120351	GE MEDICAL SYSTEMS	1.25	[0.824219, 0.824219]	[1, 0, 0, 0, 1, 0]	[-214.100006, -171, -171.25]

Contains scans from 31 patients with non-small cell lung cancer (NSCLC).
Has image position coordinates for each scan.

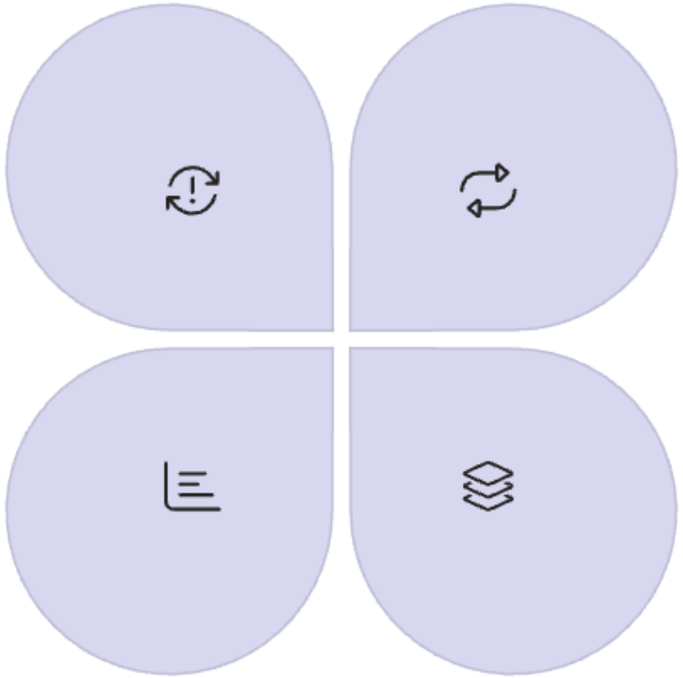
Why RIDER was chosen

Variability Analysis

Allows analysis of variability between scans.

Expert-Annotated Ground Truth

Tumor boundaries were manually delineated by oncologists, which supports supervised learning.



Test-Retest Scan Protocol

Each patient has paired test-retest CT scans taken under the same conditions within a short interval which enables analysis of feature stability and reproducibility.

Multi-Resolution

Enables use of stacking from different resolutions for super-resolution imaging.

Preprocessing steps

1. Load and Clean the Metadata

```
# STEP 1: Load Metadata
df = pd.read_csv(csv_path)
df = df.dropna(subset=[
    "x (in pixel)", "x (in pixel).1",
    "y(in pixel)", "y(in pixel).1",
    "z (imagenumber1)", "z (imagenumber2)"
```

2. Build Tumor Map with Inverted Y-Axis

Purpose: Convert the bounding box annotations to **(x, y)** center coordinates for each slice

```
# STEP 2: Build tumor map with inverted y-axis
tumor_map = defaultdict(dict)
for _, row in df.iterrows():
    rider_id = row["RIDER-ID"]
    x = int(round((row["x (in pixel)"] + row["x (in pixel).1"]) / 2))
```

3. Helper Functions

`extract_slice_index(filename)`

- Extracts slice number from DICOM filename like `slice-0123.dcm`

`create_ellipse_mask(shape, center, axes)`

- Creates a **filled elliptical mask** at given center `(x, y)` with specified radii (`ellipse_axes = (12, 16)`)

`extract_rider_id(path)`

- Extracts the `RIDER-xxxxxxxxxx` patient ID from filepath using regex

```
# STEP 3: Helper functions
def extract_slice_index(filename):
    try:
        return int(filename.split('-')[-1].replace('.dcm', '').lstrip("0") or "0")
    except:
        return None
```

4. Walk Through DICOM Files and Create Masks

Check if the current slice has a tumor annotation from `tumor_map`

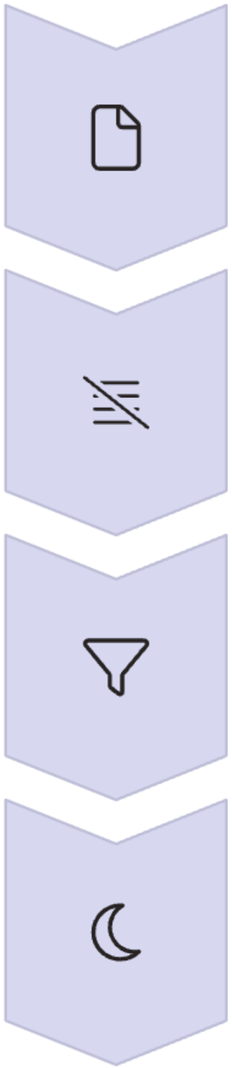
- **If yes:** draw a filled ellipse mask at the tumor center
- **If no:** create an **empty (black) mask**

```
# STEP 4: Generate one mask per .dcm file with correct y-inversion
os.makedirs(output_mask_dir, exist_ok=True)

for base_dir in base_dirs:
    for root, _, files in os.walk(base_dir):
        for file in files:
            if not file.endswith(".dcm"):
                continue
```

5. Convert dicom to png

Preprocessing Pipeline



Convert DICOM files

Convert DICOM files to PNG

Align Slices

Align slices based on anatomical landmarks.

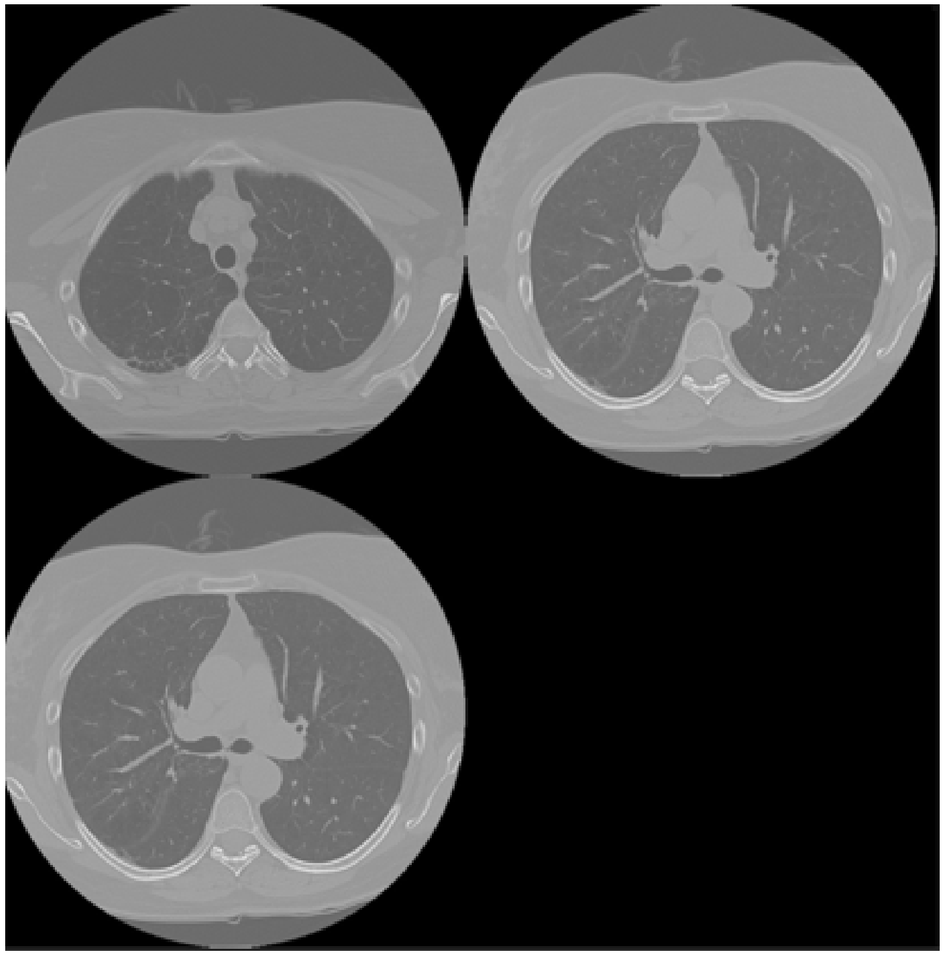
Apply noise reduction

Apply noise reduction using filters.

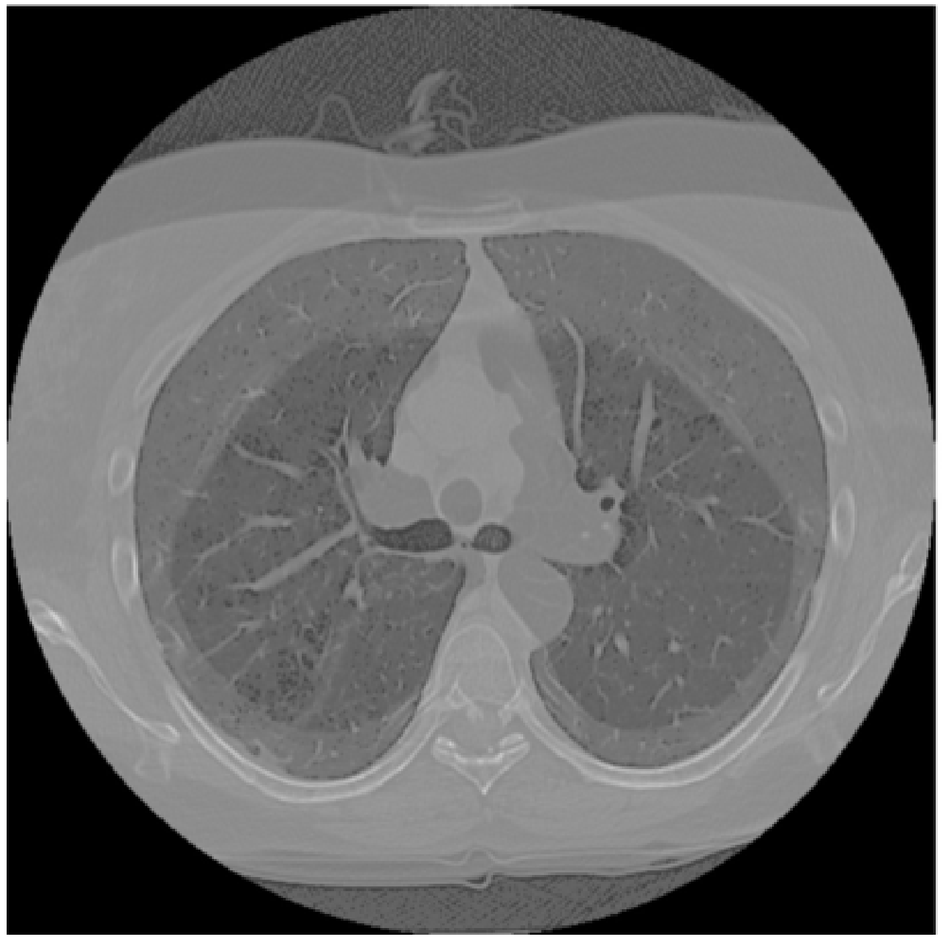
Enhance contrast

Enhance contrast through histogram equalization.

Stacked vs individual images



Input Grid
Pre Stacking



Output Image
Stacked Image

Feature Extraction

From Input Images (CT Slices)

Modality: DICOM CT slices, 512×512 (grayscale)

Preprocessing:

- HU normalization via RescaleSlope & Intercept
- CLAHE for contrast enhancement
- Percentile clipping (1st–99th)
- Resized to 256×256 and scaled to [0,1]

Stacking via MIP:

- Combines multiple slice thicknesses (1.25mm, 2.5mm, 5mm)
- Extracts a Maximum Intensity Projection (MIP) for better tumor visibility
- Forms multi-resolution 3-channel input

MIP image

From Ground Truth Masks

Elliptical Pseudo-Masks:

- Derived from metadata tumor coordinates (x, y, z)
- Fixed axes (12,16), simulating gross tumor volume

From Augmentation

Transforms:

- Flips, rotations (90°/180°/270°), brightness jitter

Purpose:

- Simulates real-world variance, improves generalization

From Model Architecture

- **Encoder**: Extracts edge, texture, and intensity-based features
- **Decoder**: Learns shape, boundary, and spatial context
- **Skip Connections**: Fuse local & global features for better segmentation

Masking from tumor coordinates

Methodology

Methodology

- **Dataset Used**: Rider Lung CT scans in DICOM format
- **Preprocessing**:
  - Convert DICOM files: Convert DICOM to PNG format.
  - Align slices: Based on anatomical lung landmarks.
  - Noise reduction: Using denoising filters.
  - Enhance contrast: Histogram equalization.
- **Super-Resolution Stacking**:
  - Utilized varying slice thicknesses (e.g., 1mm, 2.5mm, 5mm).
  - Resampled and aligned them to a common resolution.
  - Stacked to form richer 3D volumetric data per patient.
- **Segmentation Pipeline**:
  - Objective: Identify and localize lung nodules.
  - Binary masks: 1 = tumor, 0 = background.
  - Used segmentation as both preprocessing (to focus regions) and final output (tumor detection).
- **Model Architecture**:
  - Custom U-Net and Pretrained U-Net (ResNet34) from segmentation_models library.
  - Encoder-decoder structure with skip connections for spatial precision.
- **Loss Function**:
  - Combined Dice Loss and Focal Loss to handle class imbalance and improve boundary accuracy.
- **Training Strategy**:
  - Data augmentation: rotations, flips, intensity shifts to boost generalization.
  - Optimizer: Adam, LR scheduler, and early stopping to avoid overfitting.
- **Evaluation Metrics**:
  - Dice Coefficient and IoU used to measure overlap between predicted masks and ground truth.

Model architecture

2 Segmentation models used:

1)Pretrained UNet with ResNet34 Encoder

Library Used: segmentation_models (sm)

- Input shape: (256, 256, 3)
- Output: Single-channel mask (classes=1) with sigmoid activation.

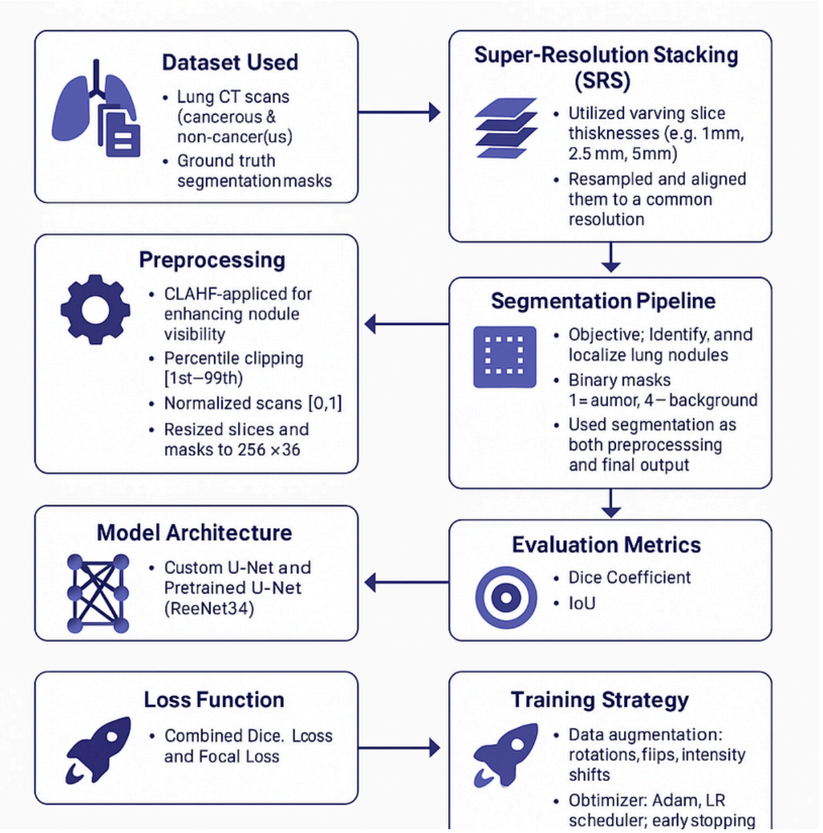
2)Custom UNet from scratch

Input shape: (256, 256, 1) (grayscale)

- Downsampling Path (Encoder):
  - 4 convolutional blocks with increasing filters (64 → 128 → 256 → 512)
  - Each block = 2 Conv2D + Dropout
  - MaxPooling after each block
- Bottleneck: 1024 filters with 2 Conv2D + Dropout
- Upsampling Path (Decoder):
  - UpSampling + Concatenate (skip connection)
  - Conv2D blocks with decreasing filters (512 → 256 → ...)
- Output Layer: Conv2D(1, kernel_size=1, activation='sigmoid')

Data Pipeline

- Step 1: Collect DICOM images and corresponding segmentation masks (.png)
- Step 2: Pair each image with its correct mask using filename matching
- Step 3: Preprocess DICOM images
  - Apply rescale slope/intercept
  - Normalize pixel intensity
  - Apply CLAHE (for contrast enhancement)
  - Clip intensity range and resize to 256×256
- Step 4: Preprocess masks
  - Resize to 256×256
  - Binarize (convert to 0/1)
- Step 5: Apply data augmentations
  - Horizontal flip
  - Random rotation (90°, 180°, 270°)
  - Random brightness adjustment
- Step 6: Normalize images to [0, 1] and convert to 3 channels if needed
- Step 7: Feed batches into a segmentation model (UNet / Pretrained UNet)
- Step 8: Train model using combined Dice + Focal loss
- Step 9: Evaluate using Dice Coefficient and IoU



Training Setup

- **Mode**l: UNet
- **Input Image Size**: 256 × 256 (grayscale or 3-channel after conversion)
- **Loss Function:**
- ▸ Combination of Dice Loss + Focal Loss
- ▸ Helps handle class imbalance and segmentation overlap
- **Optimizer:**
- ▸ Adam optimizer
- ▸ Learning rate: 0.0001
- **Metrics:**
- ▸ Dice Coefficient (overlap measure)
- ▸ IoU (Intersection over Union)
- **Batch Size**: 4
- **Epochs**: 50
- **Augmentation:**
- ▸ Horizontal flips
- ▸ Random rotations (by 90°, 180°, 270°)
- ▸ Random brightness variations
- **Training Pipeline:**
- ▸ Custom MedicalImageDataset with paired image-mask loading
- ▸ DataLoader for efficient batching and shuffling
- ▸ Model trained using .train() mode with validation loop

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 256, 256, 1)	0	–
conv2d_15 (Conv2D)	(None, 256, 256, 64)	640	input_layer_1[0][0]
conv2d_16 (Conv2D)	(None, 256, 256, 64)	36,928	conv2d_15[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 128, 128, 64)	0	conv2d_16[0][0]
conv2d_17 (Conv2D)	(None, 128, 128, 128)	73,856	max_pooling2d_3[0][0]
conv2d_18 (Conv2D)	(None, 128, 128, 128)	147,584	conv2d_17[0][0]
max_pooling2d_4 (MaxPooling2D)	(None, 64, 64, 128)	0	conv2d_18[0][0]
conv2d_19 (Conv2D)	(None, 64, 64, 256)	295,168	max_pooling2d_4[0][0]
conv2d_20 (Conv2D)	(None, 64, 64, 256)	590,080	conv2d_19[0][0]
max_pooling2d_5 (MaxPooling2D)	(None, 32, 32, 256)	0	conv2d_20[0][0]
conv2d_21 (Conv2D)	(None, 32, 32, 512)	1,180,160	max_pooling2d_5[0][0]
conv2d_22 (Conv2D)	(None, 32, 32, 512)	2,359,808	conv2d_21[0][0]
up_sampling2d_3 (UpSampling2D)	(None, 64, 64, 512)	0	conv2d_22[0][0]
concatenate_3 (Concatenate)	(None, 64, 64, 768)	0	up_sampling2d_3[0][0], conv2d_20[0][0]
conv2d_23 (Conv2D)	(None, 64, 64, 256)	1,769,728	concatenate_3[0][0]
conv2d_24 (Conv2D)	(None, 64, 64, 256)	590,080	conv2d_23[0][0]
up_sampling2d_4 (UpSampling2D)	(None, 128, 128, 256)	0	conv2d_24[0][0]
concatenate_4 (Concatenate)	(None, 128, 128, 384)	0	up_sampling2d_4[0][0], conv2d_18[0][0]
conv2d_25 (Conv2D)	(None, 128, 128, 128)	442,496	concatenate_4[0][0]
conv2d_26 (Conv2D)	(None, 128, 128, 128)	147,584	conv2d_25[0][0]
up_sampling2d_5 (UpSampling2D)	(None, 256, 256, 128)	0	conv2d_26[0][0]
concatenate_5 (Concatenate)	(None, 256, 256, 192)	0	up_sampling2d_5[0][0], conv2d_16[0][0]
conv2d_27 (Conv2D)	(None, 256, 256, 64)	110,656	concatenate_5[0][0]
conv2d_28 (Conv2D)	(None, 256, 256, 64)	36,928	conv2d_27[0][0]
conv2d_29 (Conv2D)	(None, 256, 256, 1)	65	conv2d_28[0][0]

Total params: 7,781,761 (29.69 MB)

Trainable params: 7,781,761 (29.69 MB)

Evaluation Metrics

Loss Function-

The loss function quantifies how far the model's predictions are from the true tumor masks. We used a combination of Dice Loss and Focal Loss to balance overlap accuracy and handle difficult tumor regions. Minimizing this loss improves overall segmentation quality.

Dice Coefficient-

The Dice Coefficient measures the similarity between the predicted mask and the ground truth, ranging from 0 (no overlap) to 1 (perfect overlap). It is especially effective for handling class imbalance in tumor segmentation. Higher Dice scores indicate better model performance.

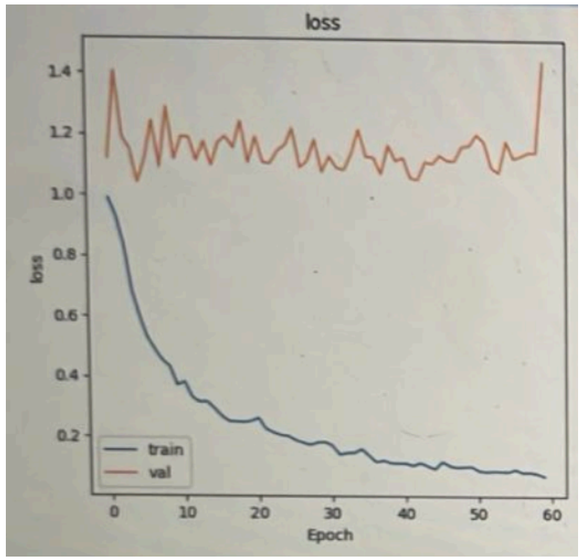
Intersection over Union (IoU)

IoU calculates the ratio of the overlap area to the combined area of prediction and ground truth masks. It provides a stricter evaluation than Dice, assessing how well the model outlines the tumor boundaries. IoU values closer to 1 signify more accurate segmentations.

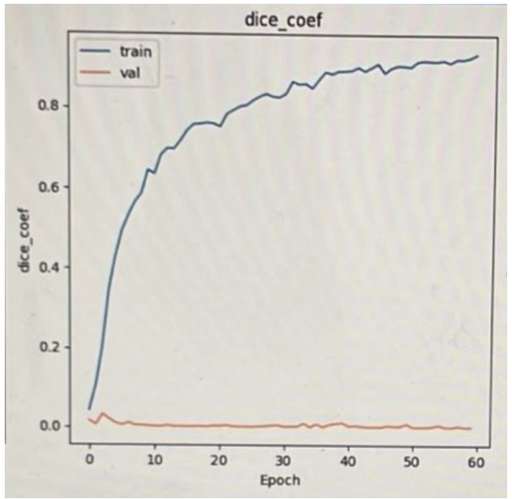
Threshold vs Dice Curve

This metric helps determine the best probability threshold to convert model outputs into binary masks. By evaluating Dice scores at different thresholds, we identify the cutoff that maximizes segmentation accuracy. It ensures optimal post-processing of predictions.

Results

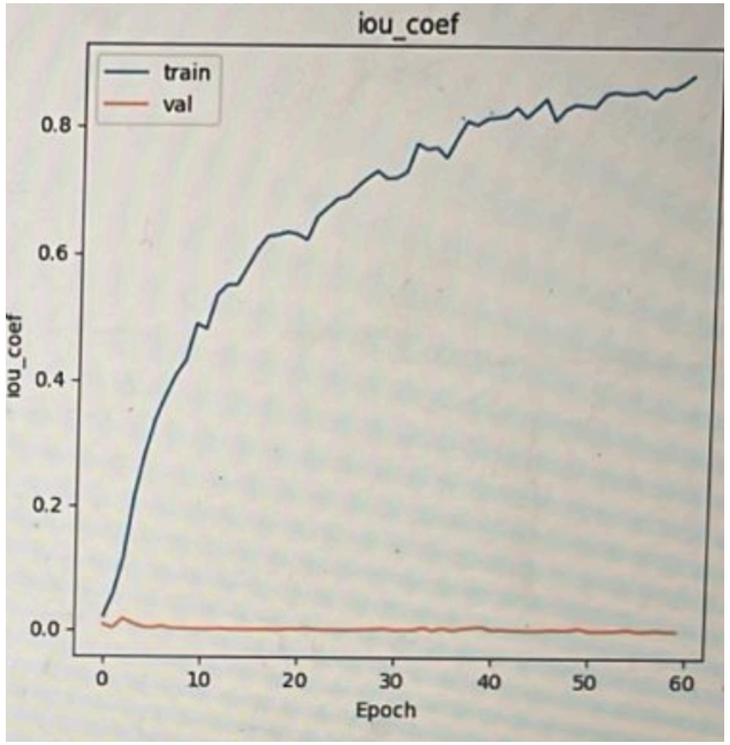


Loss over Epochs:
Training loss decreases steadily, showing the model is learning on training data. However, validation loss remains high and fluctuates, indicating poor generalization and possible divergence.



Dice Coefficient over Epochs:
Training Dice increases well, reaching close to 0.9, but validation Dice stays near zero throughout, meaning the model fails to segment correctly on validation data.

Results



IoU over Epochs:
Similar pattern—training IoU improves significantly, but validation IoU remains near zero, confirming no meaningful overlap on validation masks.

Clinical & Technological Impact

Efficiency Boost

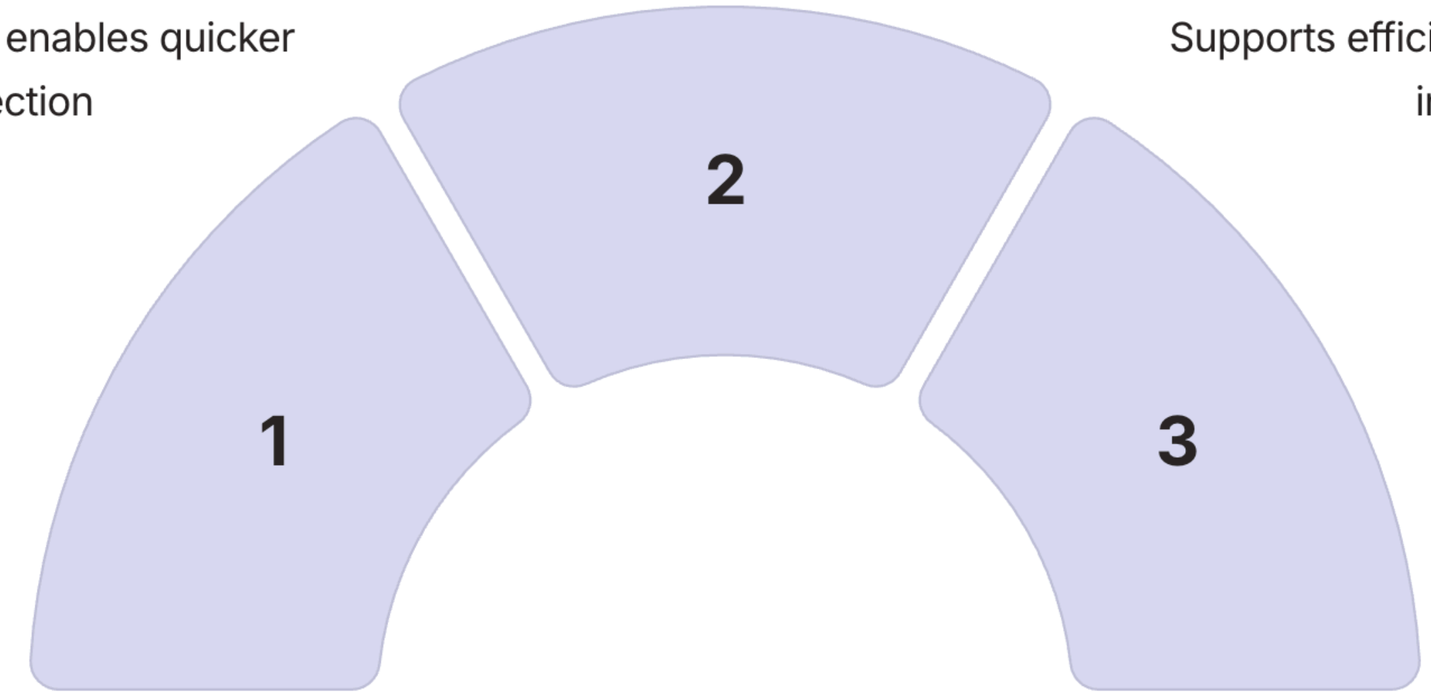
Reduces false positives, streamlines diagnostics

Early Diagnosis

Accurate segmentation enables quicker disease detection

Automated Triage

Supports efficient patient prioritization in hospitals



Challenges

No Ground Truth Masks Available

The dataset lacked expert-drawn contours, so we had to create elliptical pseudo-masks based on coordinate midpoints — introducing approximation errors.

Loss Instability

Standard loss functions) failed to converge reliably on pseudo-masks. We had to adopt a custom **Dice + Focal loss** function to balance precision and recall.

Cross-Institutional Data Variability if we scale up

CT scan protocols, resolutions, and scanner vendors vary across hospitals which may affect model generalization.

```
create_ellipse_mask(shape, center, axes):
mask = np.zeros(shape, dtype=np.uint8)
center = (np.clip(center[0], 0, shape[1] - 1), np.clip(center[1], 0, shape[0] - 1))
cv2.ellipse(mask, center=center, axes=axes, angle=0, startAngle=0, endAngle=360, color=255
return mask
```

```
# ---------- Loss Functions ----------
def dice_loss(y_true, y_pred):
    y_true_f = tf.keras.backend.flatten(y_true)
    y_pred_f = tf.keras.backend.flatten(y_pred)
    intersection = tf.keras.backend.sum(y_true_f * y_pred_f)
    return 1 - (2. * intersection + 1e-7) / (tf.keras.backend.sum

def focal_loss(gamma=2., alpha=0.25):
    def focal(y_true, y_pred):
        y_pred = tf.keras.backend.clip(y_pred, tf.keras.backend.e
        cross_entropy = -y_true * tf.math.log(y_pred) - (1 - y_tr
        weight = alpha * tf.pow(1 - y_pred, gamma) * y_true + (1 -
        return tf.reduce_mean(weight * cross_entropy)
    return focal

def combined_loss(y_true, y_pred):
    return dice_loss(y_true, y_pred) + focal_loss()(y_true, y_pre
```

References

- Armato, S.G. et al. (2008). The Lung Image Database Consortium (LIDC) and Image Database Resource Initiative (IDRI): A completed reference database of lung nodules on CT scans. Medical Physics, 38(2), 915–931. [DOI: 10.1118/1.3528204]

- Sun, W., Zheng, B., & Qian, W. (2017). Automatic feature learning using multichannel ROI based on deep structured algorithms for computerized lung cancer diagnosis. Computers in Biology and Medicine, 89, 530–539. https://doi.org/10.1016/j.compbiomed.2017.04.006

- TransSegNet & MinClassNet Study – Proposing transformer-based segmentation and a lightweight CNN classifier for lung cancer detection.

- https://www.geeksforgeeks.org/lung-cancer-detection-using-convolutional-neural-network-cnn/
- Kumar, D. et al. (2023). Lung cancer diagnosis using 3D multi-layer CNN on TCIA dataset.

- Wenqing Sun et al. (2017). Computers in Biology and Medicine, 89, 530–539.

- National Lung Screening Trial Research Team. (2011). New England Journal of Medicine, 365(5), 395–409.

- Aerts, H.J. et al. (2014). Decoding tumour phenotype. Nature Communications, 5, 4006. [DOI: 10.1038/ncomms5006]

- Shen, W. et al. (2015). Multi-scale convolutional neural networks for lung nodule classification, 1, 588–599.

- RIDER Lung CT Dataset. https://www.cancerimagingarchive.net

- Segmentation Models Library – https://github.com/qubvel/segmentation_models