**Lung Cancer Detection and Classification using 3D Convolutional Neural Network**

Aymane Mahri

*Faculty of Sciences and Technologies of Tangier*

Mohammed Amine Sabbahi

*Faculty of Sciences and Technologies of Tangier*

**Keywords**: Lung Cancer, Deep learning, Convolutional Neural Networks, Detection

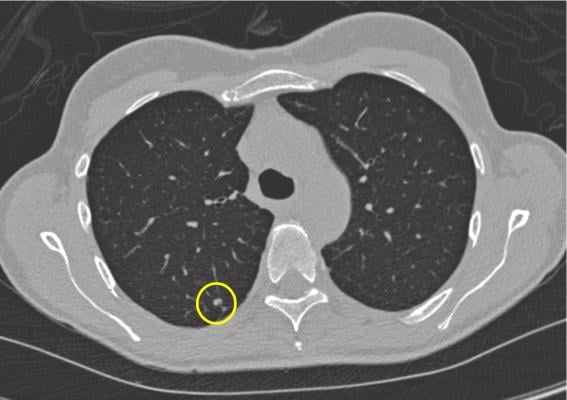
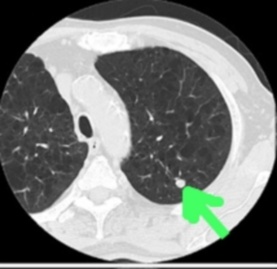
**Abstract**

This paper presents a computer-aided diagnosis system for lung cancer classification from CT scans with unmarked nodules, using a small sample of 50 patients from Kaggle Data Science Bowl 2017 dataset. Initial segmentation of lung tissue was done using thresholding, which was followed by a processing of regions by 3D CNNs for final classification. The system achieved 83.2% accuracy for the classic 3D CNN model, we also decided to implement another CNN model which was a combination of 2d and 3d convolutions for test, this model on the other hand achieved 83% accuracy.

**1. Introduction**

Lung cancer is by far the leading cause of cancer death in the US, accounting for about 1 in 5 of all cancer deaths. Each year, more people die of lung cancer than of [colon](https://www.cancer.org/cancer/types/colon-rectal-cancer.html), breast, and prostate cancers combined., ac- counting for over 234,000 cases, 125,000 deaths, and $12 billion in health care costs yearly in the U.S. [1]. It is also one of the deadliest cancers; overall, only 17% of people in the U.S. diagnosed with lung cancer survive five years after the diagnosis, and the survival rate is lower in developing countries. The stage of a cancer refers to how extensively it has metastasized. Stages 1 and 2 refer to cancers localized to the lungs and latter stages refer to cancers that have spread to other organs. Current diagnostic methods include biopsies and imaging, such as CT scans. Early detection of lung cancer (detection during the earlier stages) significantly improves the chances for survival, but it is also more difficult to detect early stages of lung cancer as there are fewer symptoms [1].

Our task is a binary classification problem to detect the presence of lung cancer in patient CT scans of lungs with and without early stage lung cancer. We aim to use methods from computer vision and deep learning, particularly 3D convolutional neural networks, to build an accurate classifier. An accurate lung cancer classifier could speed up and reduce costs of lung cancer screening, allowing for more widespread early detection and improved survival.



*Figure 1 & 2 :* 2D CT scan slice containing a small early stage lung cancer nodule.

The paper’s arrangement is as follows: Related work is summarized briefly in Section II. Dataset for this paper is described in Section III. The methods for segmentation are presented in section IV. The nodule segmentation is introduced in Section V based on U-Net architecture. Section VI presents 3D Convolutional Neural Network for nodule classification and patient classification. Our discussion and results are described in details in Section VII. Section VIII concludes the paper.

**2. Data**

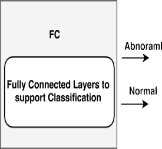
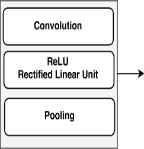
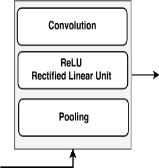
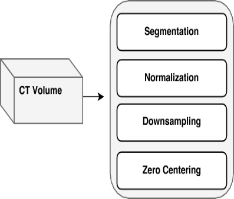
Our primary dataset is a small sample of the patient lung CT scan dataset from Kaggles Data Science Bowl (DSB) 2017 [13]. The dataset contains labeled data for 50 patients, which we divide into training set of size 40, and test set of size 10. For each patient, the data consists of CT scan data and a label (0 for no cancer, 1 for cancer). Note that the Kaggle dataset does not have labeled nodules. For each patient, the CT scan data consists of a variable number of images (typically around 100- 400, each image is an axial slice) of 512 512 pixels. The slices are provided in DICOM format. Around 70% of the provided labels in the Kaggle dataset are 0, so we used a weighted loss function in our malignancy classifier to address this imbalance.

×

Because the Kaggle dataset alone proved to be inadequate to accurately classify the validation set, we added another data that contained only lungs that were labelled as cancerous, the data was also in DICOM format, with studied made on 61 subjects. [14]

**3. Methodology**

Typical CAD systems for lung cancer have the following pipeline: image preprocessing, detection of cancerous nodule candidates, nodule candidate false positive reduction, malig- nancy prediction for each nodule candidate, and malignancy prediction for overall CT scan [15]. These pipelines have many phases, each of which is computationally expensive and requires well-labeled data during training. For example, the false positive reduction phase requires a dataset of labeled true and false nodule candidates, and the nodule malignancy prediction phase requires a dataset with nodules labeled with malignancy.



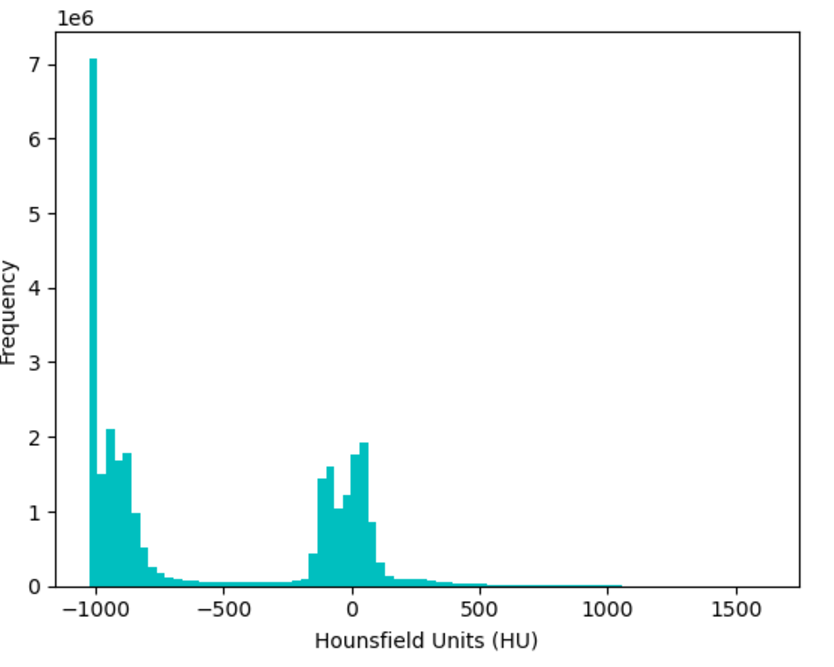
*Figure 3 :* Typical Pipeline for a CAD system

*3.1 Data Visualisation :*

The proposed methodology focuses on processing and visualizing medical imaging data, particularly DICOM (Digital Imaging and Communications in Medicine) files, to derive actionable insights. The steps involve data loading, slice counting, preprocessing using Hounsfield Unit (HU) scaling, resampling to standardized dimensions, and 3D visualization.

The analysis begins with **loading and preprocessing the DICOM images**, which are stored in patient-specific folders. We iterates through each folder, count valid DICOM slices, and verifie their integrity. This step ensures that the dataset is well-prepared and highlights the number of slices available for each patient.

To load individual patient scans, We sort DICOM slices based on their spatial position (ImagePositionPatient). This ordering preserves the anatomical continuity of slices. Slice thickness is calculated using metadata fields like ImagePositionPatient or SliceLocation. Subsequently, we convert pixel values to Hounsfield Units (HU), enabling a consistent representation of tissue densities.

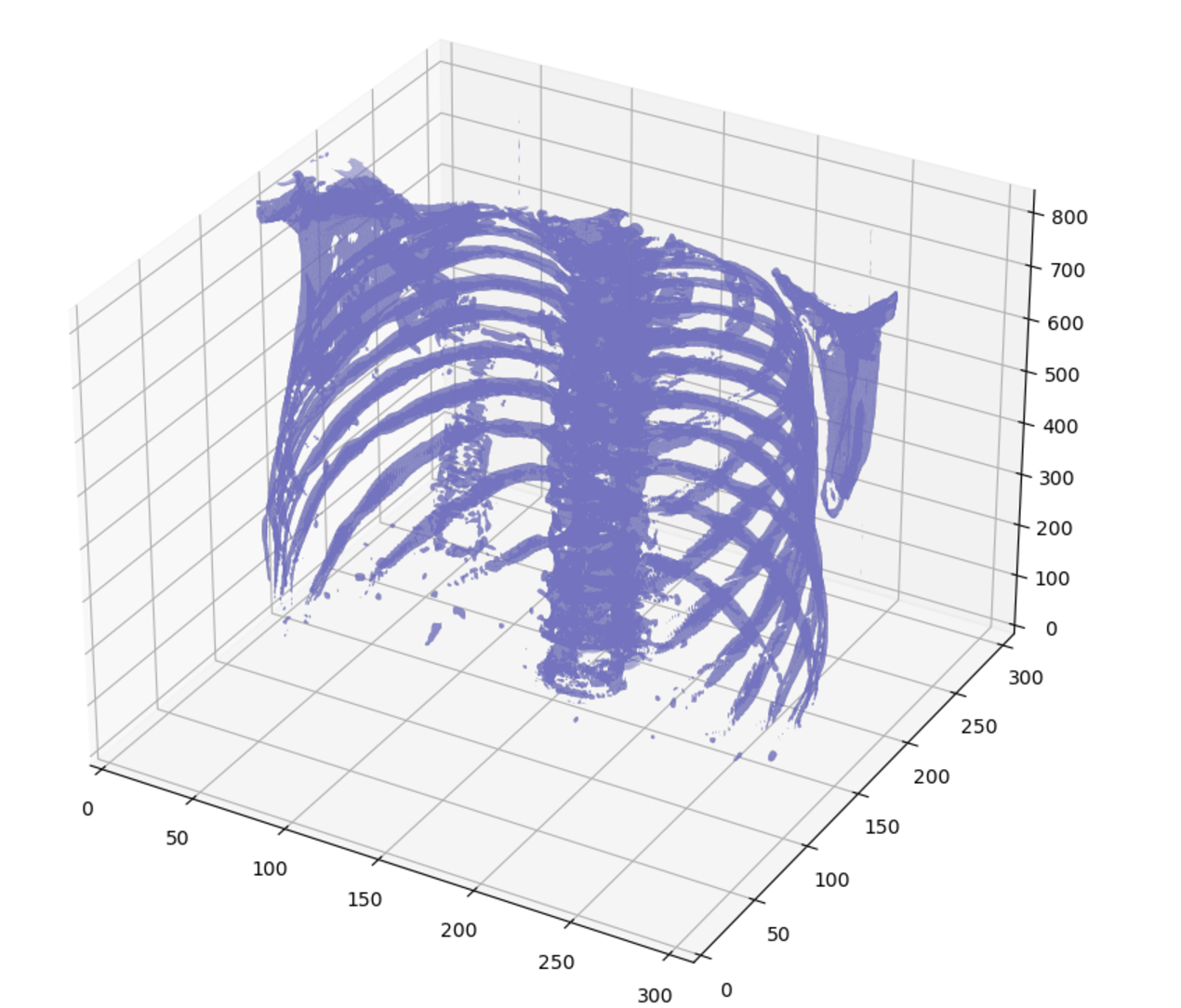


Resampling is performed to standardize the spacing between slices to 1mm x 1mm x 1mm. This uniformity is crucial for downstream tasks, including segmentation and 3D visualization, as it ensures that voxel dimensions are consistent across all scans.

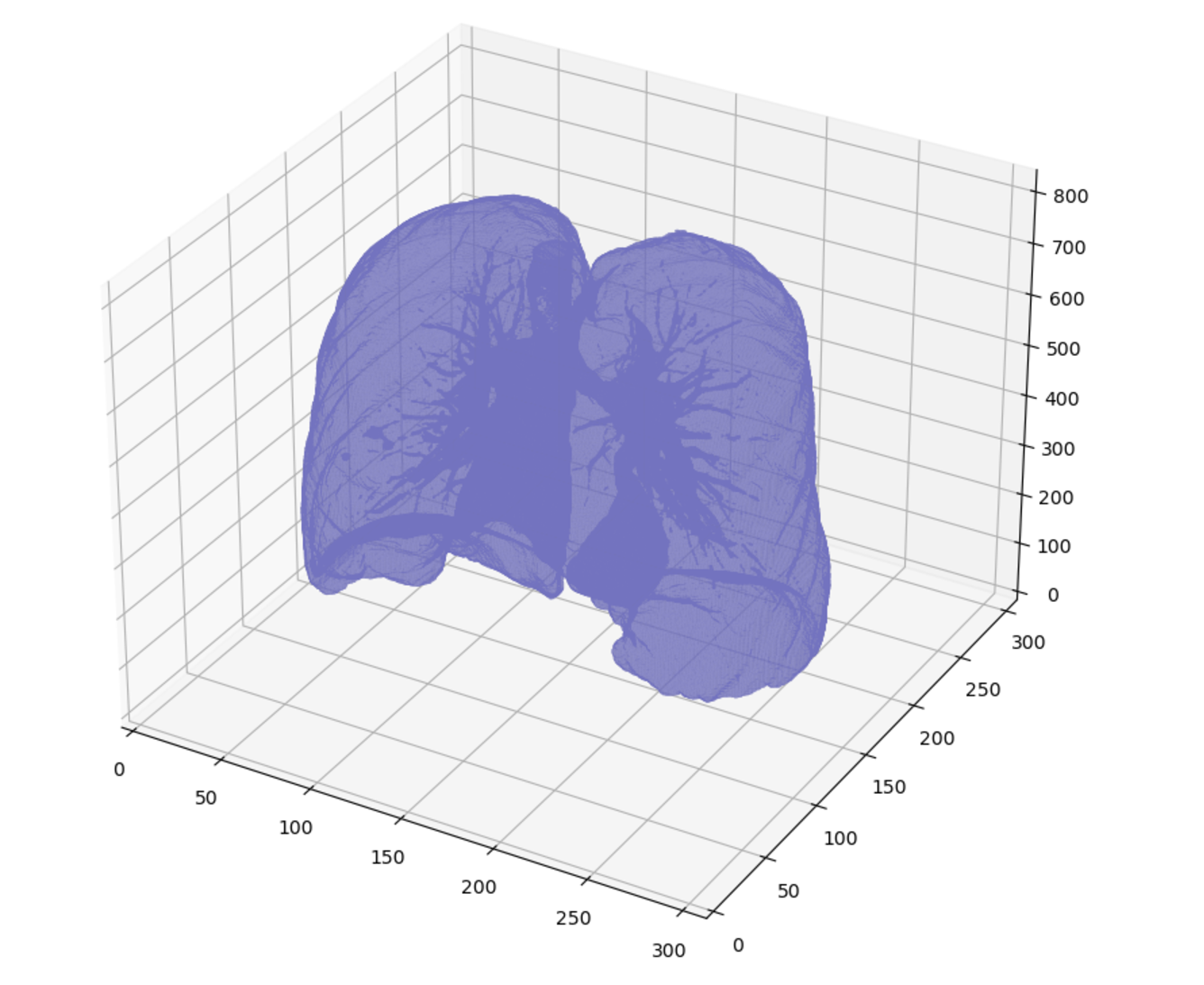
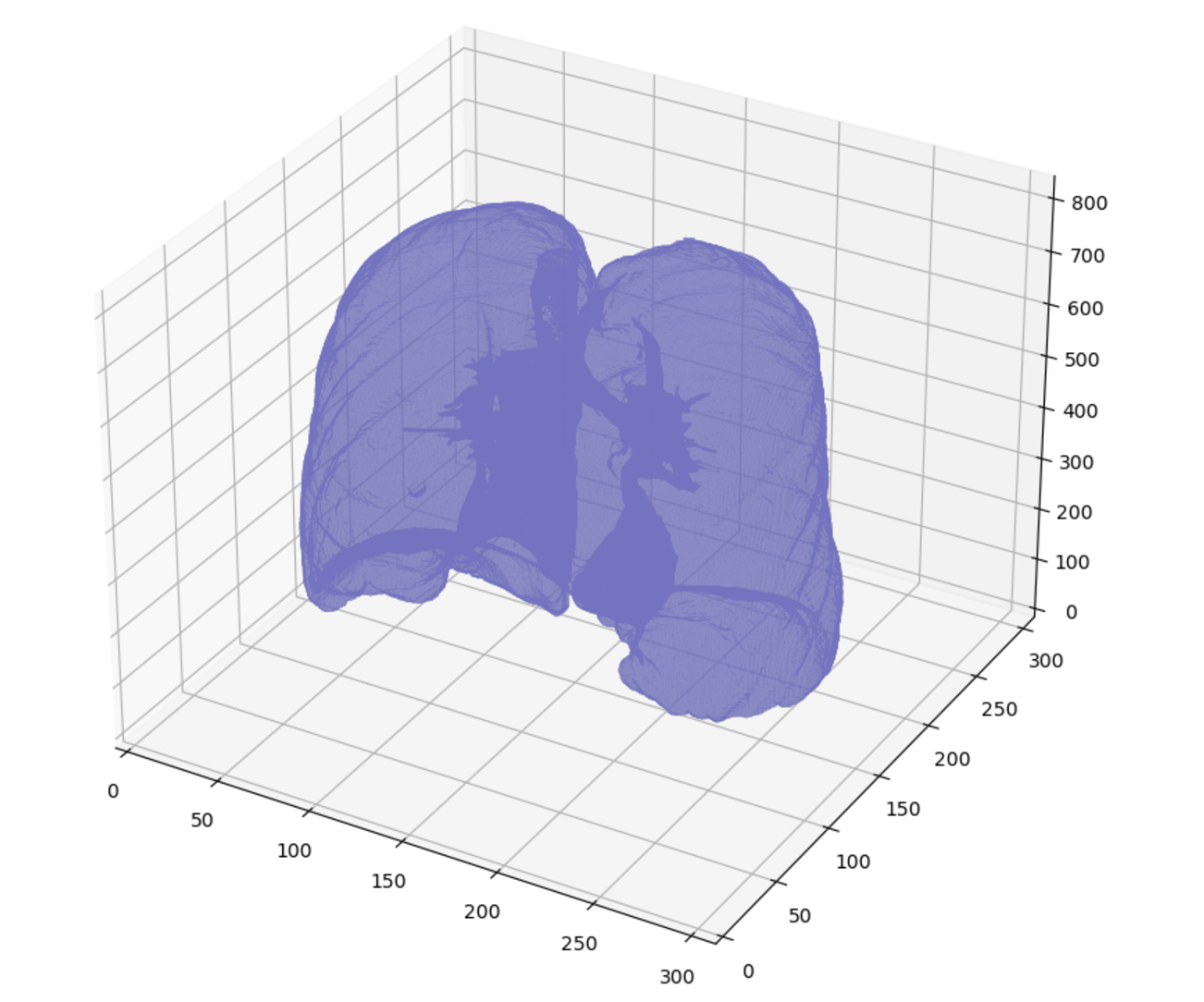
Shape before resampling (110, 512, 512)

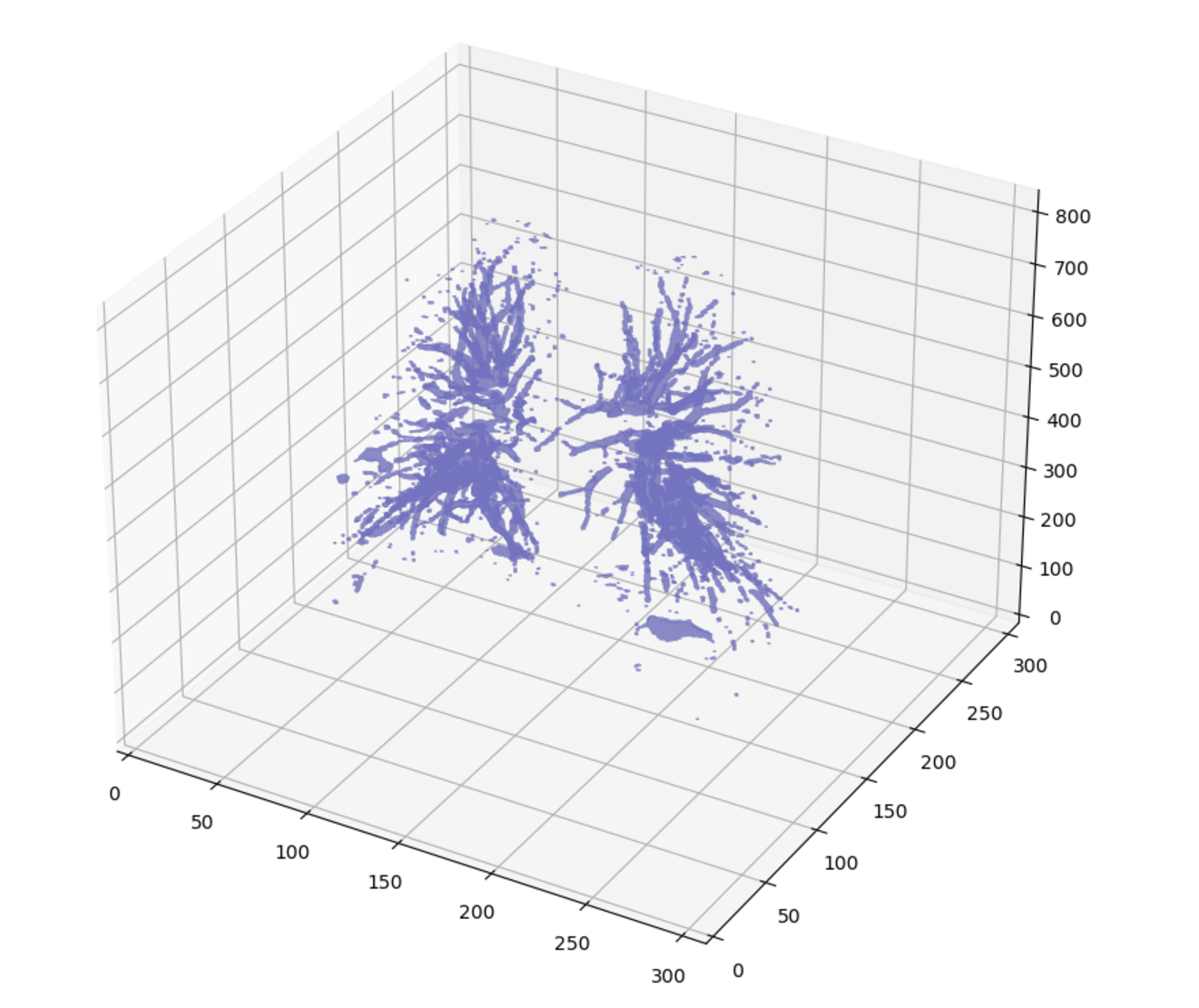
Shape after resampling (825, 306, 306)

The processed volumetric data is visualized using the **Marching Cubes algorithm**, which extracts a 3D surface mesh based on a defined HU threshold. For instance, isolating lung structures involves thresholding at a value that corresponds to air-filled regions. The resulting 3D plot renders the segmented structures, allowing for a clear understanding of spatial anatomy.



The segmentation process is refined by applying a binary threshold to identify the lung region, filling air pockets within the body, and removing unwanted regions. The final segmented volume distinctly separates the lung from surrounding tissues.





The methodology described provides a pipeline for loading, preprocessing, resampling, and visualizing medical images in 3D. The ability to generate detailed visualizations aids in diagnostic analysis and validates the accuracy of the segmentation process. Each plot included demonstrates a critical stage in the workflow, offering both technical insight and visual representation of the processed data.

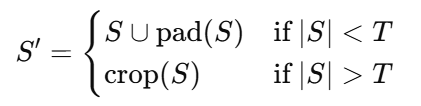
*3.2 Data Processing :*

The preprocessing of medical imaging data is critical for ensuring uniformity and robustness in machine learning models. This chapter outlines the systematic steps involved in preparing DICOM images from patients’ medical records for classification tasks. The dataset consists of volumetric scans, which vary in slice count and dimensions. The objective is to standardize the data by adjusting slice counts, resizing dimensions, and splitting the dataset into training and validation subsets.

*3.2.1 Data Standardization :*

Volumetric scans for different patients often have varying numbers of slices. To address this, we implemented a helper function, adjust\_slices, which ensures that each patient’s data is represented by a fixed number of slices. If the total slices in a scan are fewer than the target count, the function pads the volume by repeating slices using NumPy’s pad method. For scans exceeding the target count, the function crops the middle section, maintaining anatomical consistency. This uniformity is crucial for reducing variability in the input dimensions of machine learning models.

Mathematical formulation for slice adjustment:



Where S is the original slice set, T is the target slice count, S′ is the adjusted slice set.

*3.2.2 Resizing slices :*

Each slice is resized to a uniform dimension using OpenCV's resize function. The target dimensions, 128×128, are chosen to balance computational efficiency and feature preservation. This step transforms pixel arrays into a standardized grid, ensuring compatibility across the dataset.

*3.2.3 Data Integration and Labeling:*

Patient data is paired with corresponding diagnostic labels. Labels, extracted from a metadata CSV file, indicate the presence or absence of cancer. A dictionary maps patient IDs to their labels, facilitating efficient lookup during preprocessing.

*3.2.4 Dataset Construction:*

For each patient folder:

*DICOM Loading:* DICOM files are read using the pydicom library, extracting raw pixel arrays. Error handling ensures faulty files do not disrupt the pipeline.

*Slice Processing:* Slices are resized and adjusted to the target slice count.

*Dataset Assembly:* Processed volumes are appended to a data array, while labels are appended to a target array.

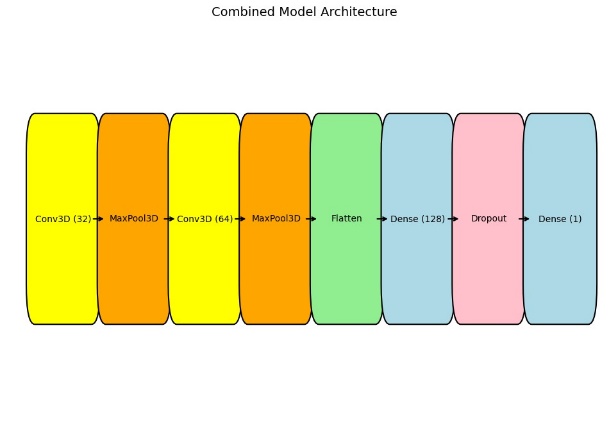
The resultant dataset has the shape (N,64,128,128, where N is the number of patients, 64 is the fixed slice count, and 128×128 represents the image dimensions. Labels are stored as a N-dimensional array.

*3.2.5 Dataset Splitting:*

The preprocessed dataset is split into training and validation subsets using an 70-30 ratio, ensuring stratification to maintain class balance. The train\_test\_split method from sklearn is employed, with a random seed for reproducibility.

*3.3 Classic 3D CNN model :*

The first model we used is classic 3D CNN model with the following architecture :



*Figure 4 :* Architecture 3d CNN classic

In this study, we utilized a 3D Convolutional Neural Network (CNN) model for feature extraction and classification tasks. The architecture of the model is as follows:

*Conv3D Layer:* The first layer is a 3D convolutional layer that applies 32 filters to the input data, with a kernel size of 3x3x3. This layer processes the input volume with dimensions (None, 62, 126, 126, 1) and outputs a feature map of shape (None, 62, 126, 126, 32). This layer has 896 parameters.

*MaxPooling3D Layer:* After the first convolution, a 3D max pooling operation with pool size 2x2x2 is applied. This reduces the spatial dimensions of the feature map from (None, 62, 126, 126, 32) to (None, 31, 63, 63, 32), retaining the depth (number of channels).

*Conv3D\_1 Layer:* A second 3D convolutional layer is applied with 64 filters of size 3x3x3. This layer processes the output from the previous max-pooling layer, resulting in an output shape of (None, 29, 61, 61, 64). This layer has 55,360 parameters.

*MaxPooling3D\_1 Layer*: Another max-pooling operation is applied with the same pool size of 2x2x2, reducing the dimensions of the feature map to (None, 14, 30, 30, 64).

*Flatten Layer:* The flattened layer reshapes the 3D feature maps into a 1D vector of size 806,400, which is the input for the dense layers.

*Dense Layer:* A fully connected (dense) layer with 128 units is applied, which enables the model to learn complex relationships between the features extracted by the convolutional layers. This layer has 103,219,328 parameters.

*Dropout Layer:* A dropout layer with a rate of 50% is applied to reduce overfitting by randomly setting half of the activations to zero during training.

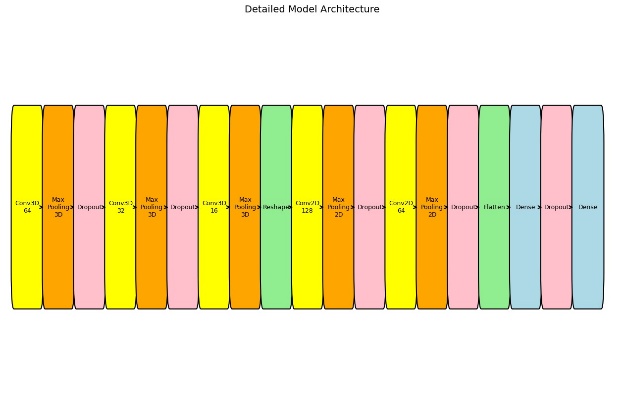
*Dense\_1 Layer:* The final dense layer consists of a single neuron, making it suitable for binary classification tasks. This layer has 129 parameters.

The total number of parameters in the model is 103,275,713, with all of them being trainable, which amounts to approximately 393.97 MB of memory.

This model architecture is designed to extract hierarchical spatial features from volumetric data through multiple convolutional layers, followed by a fully connected layer for classification. The dropout layer further improves generalization by mitigating overfitting.

*3.4 Combined 2D/3D CNN model :*

Next we used a combined 2D/3D CNN model with the following architecture :



*Figure 5 :* Architecture 2d/3d CNN combined

The model architecture presented here combines both 3D and 2D convolutional layers, designed to extract hierarchical features from volumetric data while transitioning to more traditional image-like 2D features for classification tasks. This combination leverages the strengths of 3D convolutions for spatial and temporal (or volumetric) data processing, followed by 2D convolutions to capture finer details and patterns after dimensionality reduction.

***Conv3D Layer (conv3d\_3):*** The first layer applies 32 filters of size 3x3x3 to the input data, which results in an output shape of (None, 64, 128, 128, 32). This 3D convolution captures spatial features in the depth, height, and width dimensions, typically useful for processing volumetric or video-like data.

***MaxPooling3D Layer (max\_pooling3d\_3):*** Max pooling with a pool size of 2x2x2 reduces the spatial resolution of the feature map to (None, 32, 64, 64, 32), helping to down-sample the feature maps and reduce computational complexity.

***Dropout (dropout\_5):*** This layer helps mitigate overfitting by randomly setting a fraction (typically 50%) of the input units to zero during training.

***Conv3D Layer (conv3d\_4):*** A second 3D convolution with 64 filters is applied, processing the previous feature maps to extract higher-level spatial patterns. The output shape after this layer is (None, 32, 64, 64, 64).

***MaxPooling3D Layer (max\_pooling3d\_4):*** Max pooling is applied again, reducing the feature map size to (None, 16, 32, 32, 64), further reducing spatial dimensions.

***Dropout (dropout\_6):*** Another dropout layer to reduce overfitting, applied after the second 3D convolution.

***Conv3D Layer (conv3d\_5):*** A third 3D convolution with 128 filters is applied, processing the feature maps to capture even more abstract spatial patterns. The output shape is (None, 16, 32, 32, 128).

***MaxPooling3D Layer (max\_pooling3d\_5):*** Max pooling reduces the spatial size to (None, 8, 16, 16, 128).

***Reshape Layer (reshape\_1):*** At this point, the 3D feature maps are reshaped to (None, 128, 16, 128) to facilitate the transition from 3D convolutions to 2D convolutions.

***Conv2D Layer (conv2d\_2):*** The reshaped 3D data is passed through a 2D convolutional layer with 128 filters, extracting local patterns in the 2D spatial domain. The output shape is (None, 128, 16, 128).

***MaxPooling2D Layer (max\_pooling2d\_2):*** Max pooling reduces the output size to (None, 64, 8, 128), further compressing the spatial dimensions.

***Dropout (dropout\_7):*** Dropout is applied again after the first 2D convolution to prevent overfitting.

***Conv2D Layer (conv2d\_3):*** A second 2D convolution is applied with 256 filters, which captures even finer spatial features. The output shape is (None, 64, 8, 256).

***MaxPooling2D Layer (max\_pooling2d\_3):*** Max pooling reduces the feature map size to (None, 32, 4, 256), which prepares the data for flattening.

***Dropout (dropout\_8):*** Another dropout layer to reduce overfitting.

***Flatten Layer (flatten\_1):*** The feature map is flattened into a 1D vector of size 32,768, which is the input for the fully connected layers.

***Dense Layer (dense\_2):*** A fully connected layer with 512 units is applied, allowing the model to learn complex relationships between the extracted features. This layer has 16,777,728 parameters.

***Dropout (dropout\_9):*** Dropout is applied again after the dense layer to prevent overfitting.

***Dense Layer (dense\_3):*** The final dense layer consists of a single neuron, suitable for binary classification tasks (e.g., classification of a single class or two classes). This layer has 513 parameters.

This combination allows the model to capture both fine-grained 2D features and larger-scale volumetric features, making it particularly effective for tasks where both spatial depth and 2D patterns are important.

**4. Results**

*4.1 Classic 3D CNN model :*

Here is the following results and observations :

*4.1.1 Training Metrics:*

AUC: The AUC fluctuates between 0.8742 and 0.9270, showing strong performance in distinguishing between classes but some minor instability.

Precision: Ranges from 0.8603 to 0.9077 — Precision is good, indicating the model's ability to avoid false positives.

Recall: Ranges from 0.7207 to 0.8178 — Recall is gradually improving, which means the model is identifying more of the positive cases.

Accuracy: Ranges from 0.7964 to 0.8329 — Accuracy is steadily improving, reflecting a better overall classification on the training data.

Loss: Decreasing from 0.4253 to 0.3371 — The loss is decreasing, suggesting that the model is learning and fitting the training data better over time.

*4.1.2 Validation Metrics:*

AUC: The validation AUC fluctuates between 0.8409 and 0.8598 — It remains strong but slightly lower than the training AUC, indicating that the model has a decent ability to generalize.

Precision: Mostly 1.0000 but fluctuates (0.8750 to 1.0000) — Precision is very high, indicating that the model is making very few false positive predictions on the validation set.

Recall: Ranges from 0.5833 to 0.7500 — Recall is relatively low but consistent. It still indicates that the model misses a significant portion of the true positive cases.

Accuracy: 0.7391 to 0.8261 — The validation accuracy fluctuates, but it's still somewhat lower than the training accuracy, indicating room for improvement in generalization.

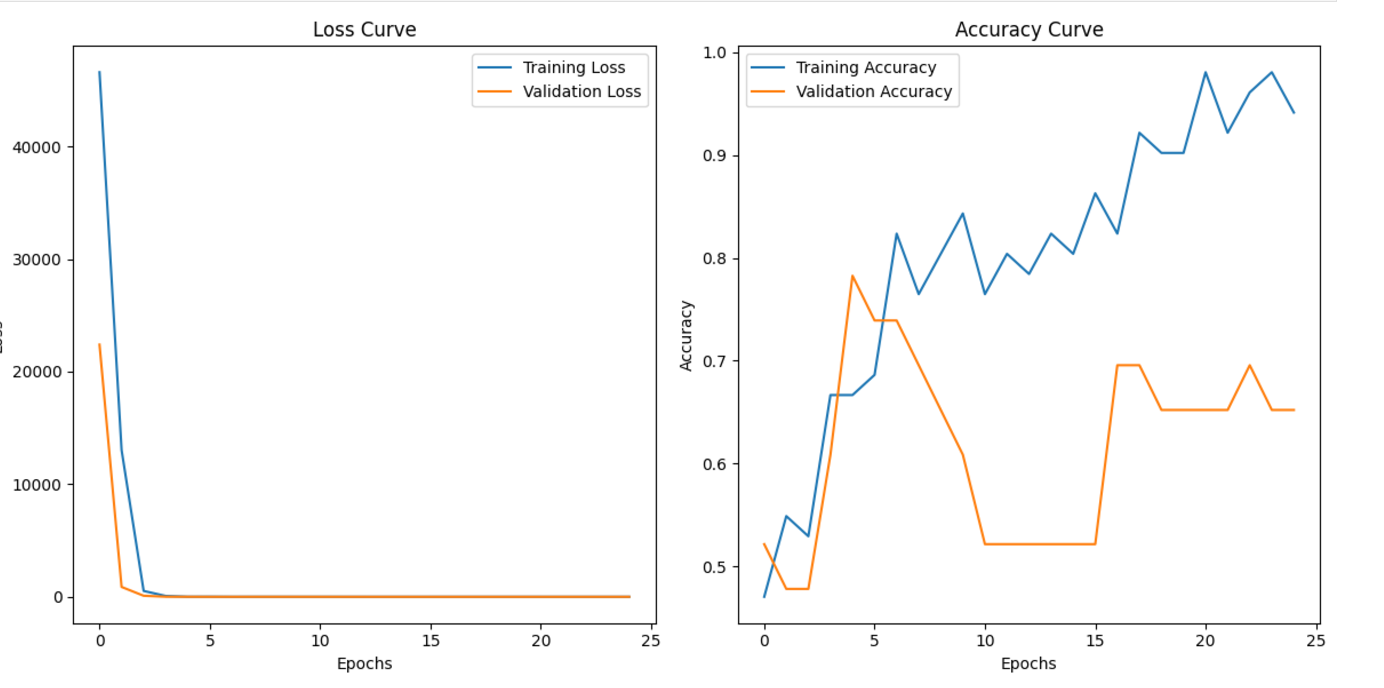
Loss: The validation loss remains relatively stable, fluctuating around 0.5825 to 0.5950 — It indicates that the model is not significantly overfitting or underfitting on the validation set, but there's still some room for improvement.

*4.1.3 Key Observations:*

Model Stability: The training metrics show good consistency, with AUC, precision, and recall improving over time. However, the validation metrics show some instability, particularly in recall, which remains lower than expected.

Precision-Recall Trade-off: The model maintains high precision on the validation set, but recall still lags behind. This suggests that while the model avoids false positives well, it struggles with false negatives.

Overfitting/Generalization: While the model is performing well on training data (good accuracy and low loss), the validation performance is still not optimal, especially with respect to recall and accuracy. The model may be overfitting to the training data to some extent.



*Figure 6 :* Results 3d CNN classic

*4.1.4 Conclusion:*

The model is performing well overall, with strong precision and decent recall, but the performance gap between training and validation still exists. The focus should be on improving recall and reducing overfitting to enhance generalization.

*4.2 Combined 2D/3D CNN model :*

This model was trained over **30 epochs.** Key metrics such as **AUC (Area Under the Curve), Precision, Recall, Accuracy,** and **Loss** were monitored on both training and validation datasets. Below is an analysis of the model's progression during training and its validation performance.

*4.2.1 Training Metrics:*

AUC: The training AUC shows fluctuation, improving from 0.8971 (Epoch 28) to a strong 0.9270 (Epoch 30). This indicates the model's growing ability to distinguish between classes.

Precision: Precision consistently improves, starting at 0.8396 (Epoch 28) and reaching 0.9077 (Epoch 30). This reflects the model's ability to reduce false positives.

Recall: Recall varies from 0.7616 (Epoch 28) to 0.7300 (Epoch 30), showing slight instability but remaining relatively strong, indicating the model's capability to detect positive cases.

Accuracy: Accuracy trends upward, increasing from 0.8068 (Epoch 28) to 0.8303 (Epoch 30), demonstrating overall improvement in correct classifications.

Loss: The training loss decreases from 0.4186 to 0.3371, confirming the model's improved fit to the training data.

*4.2.2 Validation Metrics:*

AUC: The validation AUC remains consistently strong, fluctuating between 0.8523 and 0.8598, demonstrating stable generalization performance.

Precision: Validation precision is excellent, maintaining high values (1.0000 in Epochs 28 and 29 and 0.8889 in Epoch 30), indicating minimal false positives.

Recall: Recall ranges from 0.5833 to 0.6667, showing improvement by the final epoch but still suggesting the model misses some true positive cases.

Accuracy: Validation accuracy remains steady at 0.7826 across the epochs, indicating consistent performance.

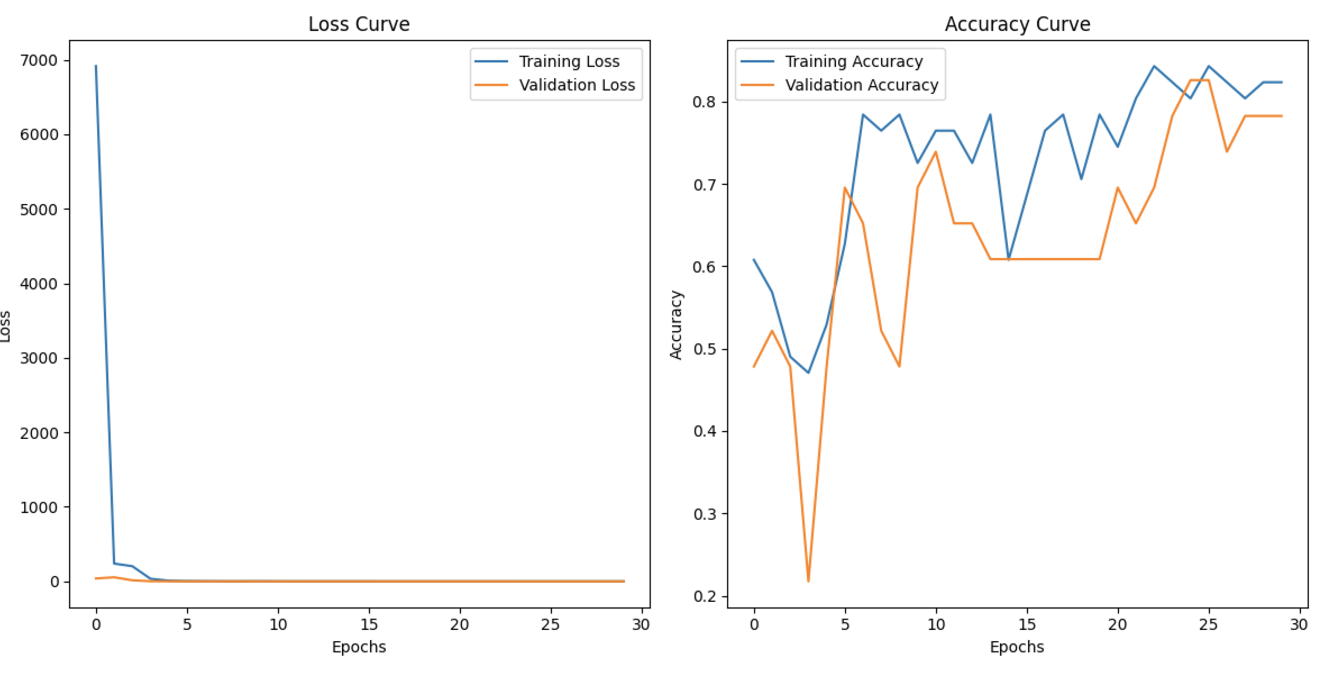
Loss: Validation loss shows slight variation between 0.5825 and 0.5950, which may suggest minor overfitting or noise.

*4.2.3 Observations:*

Training Stability: Training metrics show consistent improvement, with the model learning and fitting the data effectively.

Validation Challenges: The validation metrics, particularly recall, highlight the need to address missed positive cases while maintaining precision.

Overfitting Signs: The gap between training and validation loss, along with a slight dip in validation precision and recall, may indicate overfitting.



*Figure 7 :* Results 2d/3d CNN combined

*4.2.4 Conclusion:*

The model demonstrates strong performance with high precision and improving AUC but needs further fine-tuning to enhance recall and ensure balanced generalization between training and validation sets.

**4. Conclusion**

In this study, we developed a computer-aided diagnosis (CAD) system for lung cancer classification using CT scan data and 3D convolutional neural networks (CNNs). Despite achieving an 83.2% accuracy with the classic 3D CNN model and experimenting with a combined 2D/3D CNN model, there are several areas where improvements can be made to enhance the model's performance and generalization capabilities.

First, the relatively small dataset of 50 patients limited the model’s ability to generalize well. A larger, more diverse dataset would allow the model to capture a broader range of lung cancer cases and improve its robustness. One potential avenue is incorporating the LUNA 16 dataset, which contains a larger set of lung CT scans with labeled nodules. This would provide more comprehensive training data, especially for nodule detection and classification, potentially increasing the model's accuracy, particularly for early-stage cancers.

Furthermore, the current models could benefit from exploring transfer learning with pre-trained models on large medical image datasets. Fine-tuning these models on our lung cancer dataset might lead to better feature extraction, especially in scenarios where labeled data is scarce.

In terms of model architecture, while both the classic 3D CNN and the combined 2D/3D CNN models performed reasonably well, further experimentation with more advanced architectures like 3D DenseNets or 3D ResNets could improve feature extraction and classification accuracy. Additionally, exploring hybrid models that combine CNNs with recurrent neural networks (RNNs) or attention-based mechanisms could help improve the model's sensitivity to subtle patterns in volumetric CT data.

In conclusion, while our model demonstrates a promising approach to lung cancer classification from CT scans, there remains significant room for improvement. By leveraging larger and more diverse datasets, advanced segmentation techniques, transfer learning, and deeper network architectures, we can push the performance further and make this system more reliable and accurate for clinical use.

**5. Acknowledgements**

We would like to express our sincere gratitude to our project supervisor, Pr. Faouzi TAYALATI, for providing us with the opportunity to work on this project. Their guidance, valuable insights, and unwavering support were instrumental in the successful completion of this research. We would also like to extend our heartfelt thanks to our collaborators, Aymane Mahri and Mohamed Amine Sabbahi, for their dedicated contributions, collaboration, and teamwork throughout the project. Together, we were able to achieve the goals of this research and overcome challenges that arose during the process.

**6. References**

1. W.-J. Choi and T.-S. Choi, “Automated pulmonary nodule detection system in computed tomography images: A hierarchical block classifi- cation approach,” *Entropy*, vol. 15, no. 2, pp. 507–523, 2013.
2. A. Chon, N. Balachandar, and P. Lu, “Deep convolutional neural networks for lung cancer detection,” tech. rep., Stanford University, 2017
3. Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision.,” in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 253–256, IEEE, 2010.
4. K. Alex, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems 25 (NIPS 2012)* (F. Pereira, C. J. C. Burges,
5. H. Suk, S. Lee, and D. Shen, “Hierarchical feature representation and multimodal fusion with deep learning for AD/MCI diagnosis,” *NeuroImage*, vol. 101, pp. 569–582, 2014.
6. G. Wu, M. Kim, Q. Wang, Y. Gao, S. Liao, and D. Shen, “Unsupervised deep feature learning for deformable registration of mr brain images.,” *Medical Image Computing and Computer-Assisted Intervention*, vol. 16, no. Pt 2, pp. 649–656, 2013.
7. Y. Xu, T. Mo, Q. Feng, P. Zhong, M. Lai, and E. I. Chang, “Deep learn- ing of feature representation with multiple instance learning for medical image analysis,” in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 1626–1630, 2014.
8. D. Kumar, A. Wong, and D. A. Clausi, “Lung nodule classification using deep features in ct images,” in *2015 12th Conference on Computer and Robot Vision*, pp. 133–138, June 2015.
9. Y. Bar, I. Diamant, L. Wolf, S. Lieberman, E. Konen, and H. Greenspan, “Chest pathology detection using deep learning with non-medical train- ing,” *Proceedings - International Symposium on Biomedical Imaging*, vol. 2015-July, pp. 294–297, 2015.
10. W. Sun, B. Zheng, and W. Qian, “Computer aided lung cancer diagnosis with deep learning algorithms,” in *SPIE Medical Imaging*, vol. 9785, pp. 97850Z–97850Z, International Society for Optics and Photonics, 2016.
11. J. Tan, Y. Huo, Z. Liang, and L. Li, “A comparison study on the effect of false positive reduction in deep learning based detection for juxtapleural lung nodules: Cnn vs dnn,” in *Proceedings of the Symposium on Modeling and Simulation in Medicine*, MSM ’17, (San Diego, CA, USA), pp. 8:1–8:8, Society for Computer Simulation International, 2017.
12. R. Golan, C. Jacob, and J. Denzinger, “Lung nodule detection in ct im- ages using deep convolutional neural networks,” in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 243–250, July 2016.
13. Kaggle, “Data science bowl 2017.” [https://www.kaggle.com/c/data-](http://www.kaggle.com/c/data-) science-bowl-2017/data, 2017.
14. LUNA16, “Lung nodule analysis 2016.” https://luna16.grand- challenge.org/, 2017.
15. M. Firmino, A. Morais, R. Mendoa, M. Dantas, H. Hekis, and

R. Valentim, “Computer-aided detection system for lung cancer in computed tomography scans: Review and future prospects,” *BioMedical Engineering OnLine*, vol. 13, p. 41, 2014.

1. S. Hawkins, H. Wang, Y. Liu, A. Garcia, O. Stringfield, H. Krewer,

Q. Li, D. Cherezov, R. A. Gatenby, Y. Balagurunathan, D. Goldgof,

M. B. Schabath, L. Hall, and R. J. Gillies, “Predicting malignant nodules from screening ct scans,” *Journal of Thoracic Oncology*, vol. 11, no. 12, pp. 2120–2128, 2016.

1. M. S. AL-TARAWNEH, “Lung cancer detection using image processing techniques,” *Leonardo Electronic Journal of Practices and Technolo- gies*, pp. 147–158, June 2012.
2. O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional net- works for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015.
3. M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le,

P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, “On rectified linear units for speech processing,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3517– 3521, May 2013.

1. A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in Proc. ICML, vol. 30, 2013.
2. L. Bottou, Large-Scale Machine Learning with Stochastic Gradient Descent, pp. 177–186. Augest 2010.
3. I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in Proceedings of the 30th International Conference on International Conference on Machine Learning, ICML’13, pp. 1139–1147, JMLR.org, 2013.
4. H. Han, L. Li, H. Wang, H. Zhang, W. Moore, and Z. Liang, “A novel computer-aided detection system for pulmonary nodule identification in ct images,” in Proceedings of SPIE Medical Imaging Conferenc, vol. 9035, 2014.
5. LungCT-Diagnosis,“LungCT-Diagnosis.” https://www.cancerimagingarchive.net/collection/lungct-diagnosis/