Lung Nodule Detection on CT images

A. Gimesi, S. Dash, J.S. Ibarra, K. Sleiman, S. Chattopadhyay

University of Cassino, Cassino, Italy.

Abstract

Lung cancer remains one of the primary causes of cancer-related mortality globally, with early detection of lung nodules being crucial for improving patient prognosis. This project explores the development of a comprehensive system for lung nodule detection on CT images, utilizing both traditional machine learning and advanced deep learning techniques. Using the LUNA16 dataset, our approach involves candidate extraction, false positive reduction, and the application of convolutional neural networks (CNNs) such as V-Net and RetinaNet. Additionally, we address challenges related to class imbalance and variability in CT image quality. While our methods achieved fair results, highlighting the complexity of automated lung nodule detection.

Keywords: **Lung Nodule Detection, CT Images, Machine learning, Deep Learning, Medical Imaging, Convolutional Neural Networks, V-Net, Candidate Extraction, False Positive Reduction, Class Imbalance, LUNA16 Challenge**

1. Introduction

Lung cancer is one of the leading causes of cancer-related deaths worldwide. Early detection of lung nodules, which are potential indicators of lung cancer, plays a crucial role in improving patient outcomes. Automated segmentation and classification of lung nodules from medical images can aid radiologists in timely diagnosis and treatment planning. This will increase the survival rate of many patients and decrease healthcare costs. In this project, we propose to develop a traditional system and a deep learning pipeline for lung nodule detection using the LUNA16 challenge dataset.

The Data

In the project, the analyzed lung CT scans were collected from the LUNA16 competition dataset, which uses cases from the publicly available LIDC/IDRI database. The LIDC/IDRI database data uses the Creative Commons Attribution 3.0 Unported License, and the data for LUNA16 is made available under a similar license, the Creative Commons Attribution 4.0 International License.

In total, 888 CT scans are included. The LIDC/IDRI database also contains annotations collected during a two-phase annotation process using four experienced radiologists. Each radiologist marked lesions they identified as non-nodule, nodule < 3 mm, and nodules >= 3 mm. The publication for the details of the annotation process is also publicly available [reference]. The reference standard of the dataset consists of all nodules >= 3 mm accepted by at least 3 out of 4 radiologists during the annotation process. The annotations not included in the reference standard (non-nodules, nodules < 3 mm, and nodules annotated by only 1 or 2 radiologists) are referred to as irrelevant findings. The list of irrelevant findings was provided inside the challenge, but we disregarded those for our project.

Images

The complete dataset is divided into 10 subsets that should be used for the 10-fold cross-validation. All subsets are publicly available as compressed zip files. In our project, mainly due to the lack of hardware, we reduced the extent of the complete dataset into 2-5 subsets depending on the nodule detection approach. For the machine learning approach, we reduced the analyzed cases to 3; for deep learning, to 5 subsets.

CT images

CT imaging devices use X-ray images of the patient taken from different positions. The X-ray tube and sensors rotate around the patient, who is lying on a motorized table that moves through the scanner. As the X-rays pass through the body, different tissues absorb varying amounts of radiation. Denser tissues like bone absorb more X-rays, while softer tissues like muscles and organs absorb less. The sensors detect the amounts of X-rays that pass through and convert this information into electrical signals. These signals are sent to a computer which uses a mathematical algorithm called "reconstruction" to process these signals and create cross-sectional images (slices) of the body. This reconstruction creates a 3D image where for each referenced position voxel has the combined intensity value from the acquired x-ray images. The difference between Z dimensions, noise, and image quality are due to the different CT devices and reconstruction algorithms used for the data acquisition.

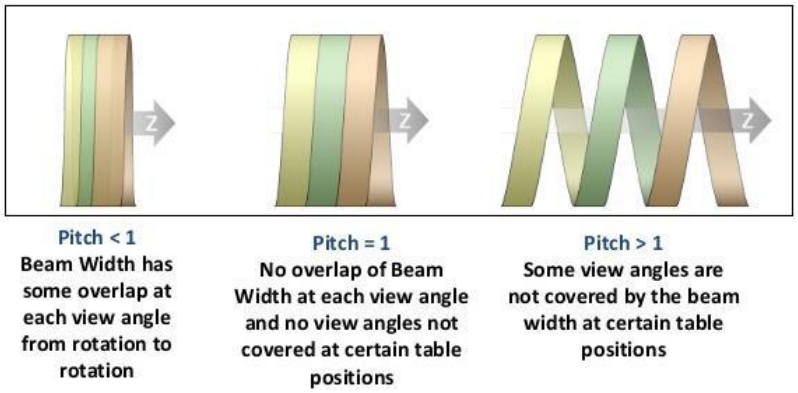

Figure 1. CT Acquisition Techniques

Storage and Handling

In each subset, CT images are stored in Meta Image (mhd/raw) format. Each .mhd file is stored with a separate .raw binary file for the pixel data. For loading the 3D CT scan images and their metadata, the SimpleITK Python library was used. The 3D images have uniform dimensions in the x and y axes but differ in the z-axis dimension.

Annotations

The annotation file was a CSV file that contains one finding per line. Each line holds the SeriesInstanceUID of the scan, the x, y, and z positions of each finding in world coordinates, and the corresponding diameter in mm. The annotation file provided by the challenge contains 1186 nodules.

Challenges

Lung CT images are 3D; each slice of the raw image is 2D. Nodules found inside a 3D image have many shapes and sizes. Some are considered small nodules, while others are normal or even large-sized nodules. Slicing through the CT image and through the nodule, the object of detection will generally be circular. However, soft tissue inside the lungs can have similar shape consistency, making false positives inevitable.
Class imbalance, like any medical imaging problem, is present in 3D CT images. In our dataset, the number of nodules for each case can range from 0 to 5. This number is much smaller than the number of slices along the axial axis, which ranges approximately from 150 to 750. The imbalance also occurs in each slice, where there are many candidates. Our goal was to minimize this number as much as possible while keeping the sensitivity high.
Nodules are inside the lungs, but some nodules are very close to the lung walls. This inconsistency in the position of the nodule affects inaccuracies and randomness.
Since CT images in the dataset of study LUNA16 are obtained from different institutions, the CT reconstruction algorithms used are different. This will create different contrast images with different properties. Furthermore, each CT image has a different spacing value, which affects the slice thickness of the 2D images.
Artifacts are found in many cases in the dataset. Metal and other objects are present in the lung images, which affects the contrast, pixel value, and image processing of these images.
In addition, the fact that nodules are present in the lungs and the CT images are in the Hounsfield Unit helps us characterize the location of the nodules. But in some cases, the HU value of the nodules does not abide by the biological properties of these particular tissues, which makes processing them more complicated.

Research Objectives and Methodology

Project Goal

Our goal is to detect and locate the annotated nodules in the CT scans using different approaches with Machine- Deep Learning. With the comparison of our acquired results and the available research and papers done by peers we will conclude the findings of our findings.

Related work

Lung nodule detection has seen significant research efforts aimed at enhancing the accuracy and efficiency of automated detection systems. Although transformers have surpassed convolution-based techniques, here are some of the states of the art techniques in this particular field.

Deep Learning Approaches
Convolutional Neural Networks (CNNs): Widely used for their ability to learn features from raw image data. For instance, Setio et al. (2016) demonstrated the effectiveness of multi-view CNNs in detecting lung nodules.
UNet Architecture: Popular for medical image segmentation due to its encoder-decoder structure, as shown by Ronneberger et al. (2015).
3D CNNs: Utilized to capture spatial context in CT scans, with notable improvements in small nodule detection by Ding et al. (2017).

Traditional Machine Learning Methods
Support Vector Machines (SVMs): Previously common for nodule detection, focusing on hand-engineered features like intensity, texture, and shape.
Random Forests: Effective in handling high-dimensional data and reducing false positives, especially when combined with feature selection techniques.

2. Image Processing

For the image processing pipeline, we reduced the dataset to the cases where there is at least 1 annotation marked by professionals. This was done because of the enormous size of the available dataset and also the CT scans.

Our proposed method for the detection of nodules is done using an image processing pipeline to extract the candidates followed by a machine learning model to reduce the false positive candidates [13], leaving just the most probable objects on the CT scan, which would allow professionals to drastically reduce the time of analysis of the whole CT case.

The proposed image processing pipeline for candidate extraction consists of:

- Lung segmentation
- Candidate extraction
- Feature vector creation

Lung segmentation

To aid the development of the nodule detection algorithm, lung segmentation images computed using an automatic segmentation algorithm are provided. For this task many papers were taken as inspiration [12, 17, 18] but a new implementation was developed. The lung segmentation images are not intended to be used as the reference; their only purpose is to help aid the extraction of the nodule candidates.

The lung segmentation process is done using the slices extracted from the z-axis of the 3D lung image and after the segmentation is done on all slices the image is reconstructed back into a 3D binary image for further usage.

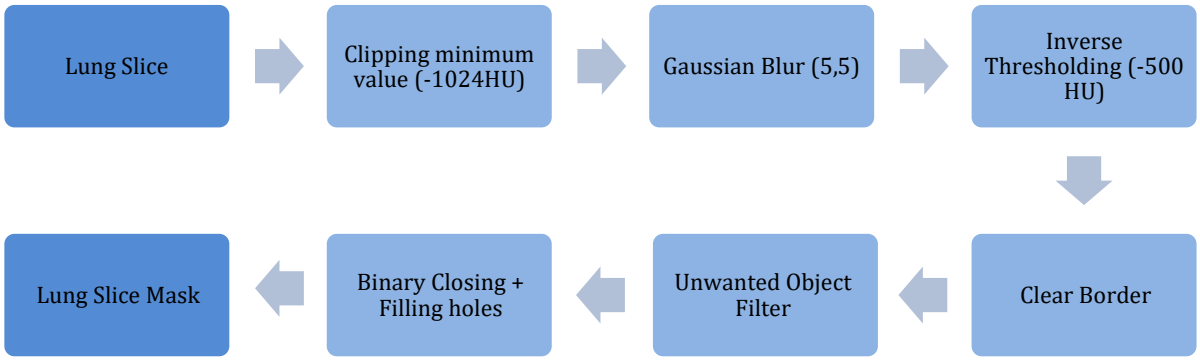

Figure 2. Lung Segmentation Pipeline

Firstly, because of the different devices the cases were gathered from, some images have the area outside of the circular detector set to different negative values (-1024, -2048, -3024) the images' lowest value is clipped to the highest minimum value, -1024. Since the clipping does not result in loss of information but helps to normalize the intensity ranges the segmentations are performed on these clipped images.

The slice-wise segmentation exploits the fact that most of the lung is filled with air, which results in less absorption of the x-rays thus giving a low HU value for the majority of the lung area. A binary thresholding operation is done with hardcoded -500 HU intensity threshold, resulting in the air filled and low absorbing lung tissues to be selected as 1 in the binary image.

After the binarization many tissues are present in the lung e.g. bronchi, bronchiole, blood vessels, connective tissue, which results in spots in the binarized image, moreover the air outside the patient's body and the outline of table that the patient is lying on are also selected as positives after the binarization.

The area outside of the patient's body, labeled with 1 in the binary image, is removed during the segmentation process using the fact that the air-filled areas outside- and inside the patient's body are separated by the body area, which is binarized to 0 during the thresholding. By erasing the connected component that touches the boundaries of the image the area of the air outside of the patient's body is excluded from the binary image. [14]
The only remaining connected components left in the binary image are the inside area of the patient bed, the inside area of lungs, and in some cases the inside of the trachea.

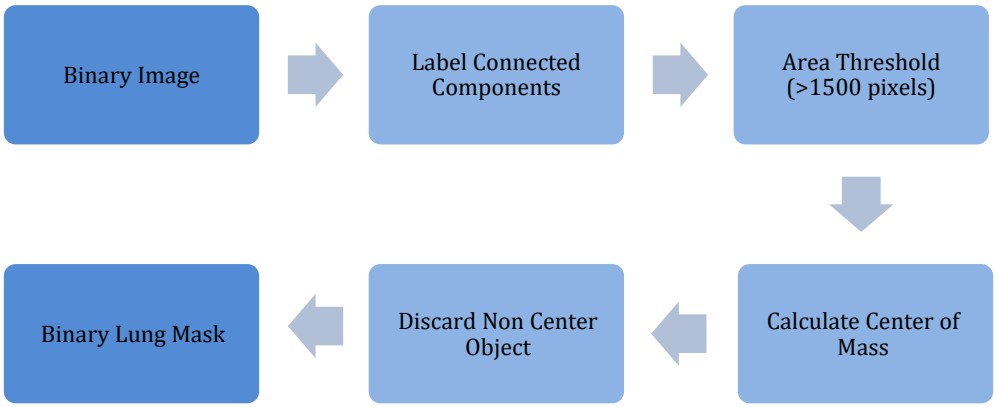

Figure 3. Unwanted Object Filter pipeline

A custom object filter was created to only keep the areas inside the lung. The object filter works as follows: binary closing with 5-pixel radius disk shaped structuring element is performed to connect disconnected lung elements, afterwards the connected components are labeled from which the ones with less than 1500 pixel area. After the area wise filtering, in most of the cases the patient bed is still present in the binary image. For the removal of this object the object's center of mass is used. The bed is always located towards the bottom of the slice, thus its center of mass is in the lowest quarter of the image. This fact is exploited and the objects whose center of mass's y coordinate is higher than 75% or lower than 25% of the shape of the image are discarded from the binary image.

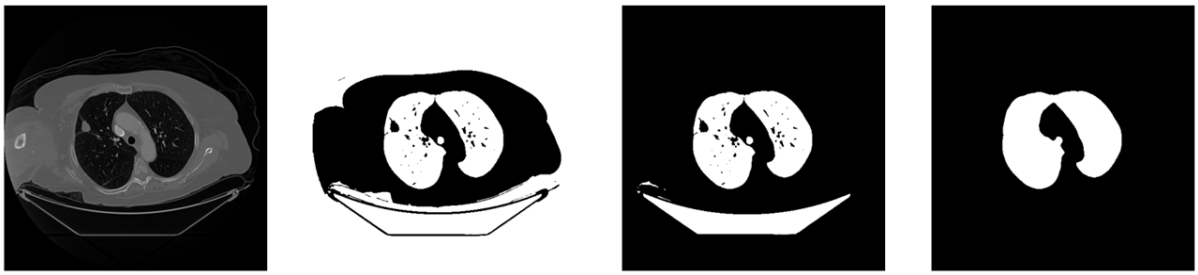

Figure 4. Unwanted filter process

After these processes the only imperfections left in the binary image are the holes created by the vessels and other tissues inside of the lungs. Binary closing with a 5-pixel radius disk shaped structuring element is performed and binary fill holes methods are used for this problem leaving the only filled lung areas in the binary mask.

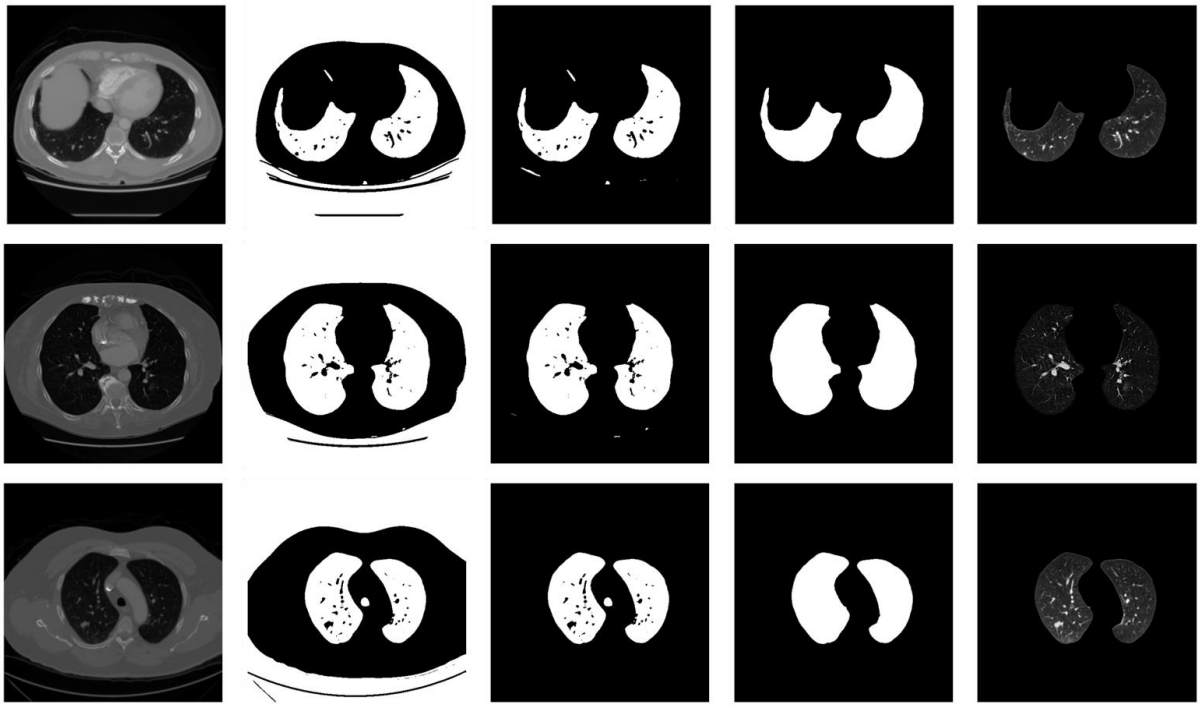

Figure 5. Lung Segmentation Process

For the lung segmentation process a number of problems arose because of the characteristics of the dataset. In some cases, one lung piece can be separated into two separate objects or a piece of another organ (e.g. liver) that's why the closing is required, but this step sometimes connects the lungs or one lung to the trachea resulting in a not optimal lung segmentation.

In some other cases there is a lung nodule located on the wall of the lung (juxta-pseudo nodules) [14] and if it has size bigger than 6 pixels in diameter the closing doesn't completely fill up the area. This result not only causes uneven borders for the lung mask, but it also loses the most important information for the objective, the nodule's area in the lung. This problem is corrected in the candidate extraction part.

Figure 6. Wall Case Example

Candidate extraction

The primary goal of this section is to create a pipeline to extract candidate centroids, which will be used to select patches for feature extraction. The general workflow of this is divided into three parts, the preprocessing stage, the candidate generation stage and the post processing stage, shown in Figure 7.

Figure 7.Overall Workflow for Candidates extraction

Pre-processing

The goal of preprocessing is to set up a foundation for the candidate generation stage. The raw CT image slice is transformed to a binary image using the pipeline described in Figure 8.

Figure 8. Preprocessing Pipeline

1. **HU Threshold Clipping:** Each pixel or voxel in a computed tomography (CT) scan is represented not by an 8-bit unsigned integer range of 0-255, but by a 16-bit signed integer range of -32,768 to 32,767. These values correspond to Hounsfield Units (HU), different tissue types within the scan are quantified with a range of HU values. Since our focus is on lung nodules, we select a specific HU range that reliably includes lung nodules while excluding other tissues. Specifically, we clip the CT scan values to a HU range of -400 to 200. Any values outside this range are set to the air threshold (HU value 1000). This ensures that only the relevant lung tissue values are retained for further analysis. Studies suggest that the HU range for lung nodules typically have values of 20 to 40 [15]. A wider range is chosen after careful observation of CT scans and raw values. As previously stated, some CT scans contain artifacts, the lower negative threshold is selected to accommodate those cases to increase sensitivity. Figure 9. clearly shows the removal of irrelevant objects such as bones, fats, etc.

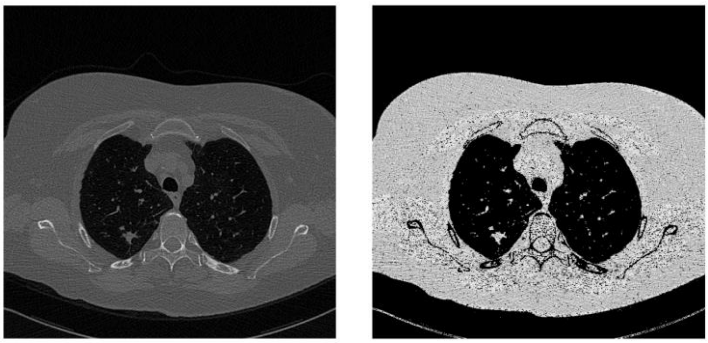

Figure 9. Raw CT Axial Slice (left) and corresponding HU Clipped Image (right)

2. **Denoising:** The HU threshold clipped image is observed to have many cavities due to the noise in the raw image being set as the air threshold. To improve the coherence of the binary image a gaussian blur with a kernel size of (5,5) is applied.

3. **Binarization:** Next the slice is binarized using the adaptive threshold OTSU method. Although other methods can also be used for binarizing the image, OTSU is chosen because after clipping the raw values using the HU Threshold and setting the values outside the range to the air threshold we will get a bimodal image. Figure n. shows the denoised image and the binarized image. The binarized image is morphologically closed with a kernel of (5,5), this is done because some nodules and objects are hollow due to artifacts or noise.

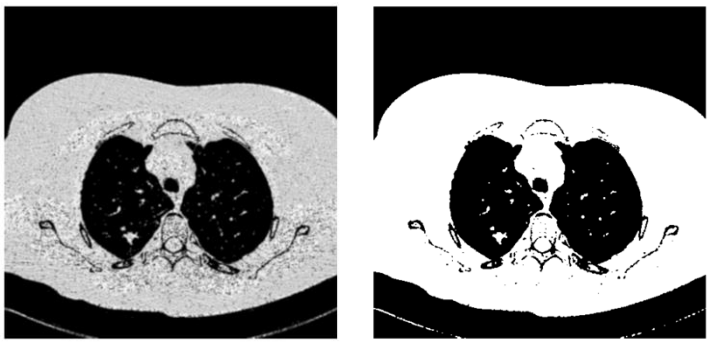

Figure 10. Denoised axial slice (left) and corresponding Binarized axial slice (right)

Candidate Generation

The most important step in the extraction workflow. The primary aim of this subsection is to extract all the lung nodules present while minimizing the false positive objects. In other words, the goal is to achieve high sensitivity* in detecting true lung nodules while reducing the total number of non-nodule candidates. Keeping this trade-off in mind, we used two workflows which will be discussed in detail.

*Note: The sensitivity described in this section is the percentage of positive lung nodules detected in the generation stage with the ground truth, and it does not signify the final sensitivity of the candidate extraction workflow. For this stage only axial slices from one subset containing the positive nodules center (z coordinate) were used, so actual sensitivity and number of candidates would vary when the process will be repeated for all the slices. Sensitivity in the section would be referred to as sensivity_0 to avoid confusion.

Method 1

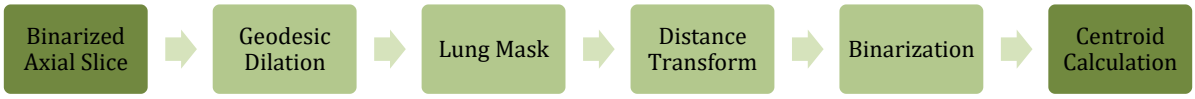

Figure 11. Method 1 Pipeline

Figure. 11 describes the major steps in this method, preparation and transformation steps are not shown but will be explained.

1. **Geodesic Reconstruction:** The principle of geodesic reconstruction is to perform repeated dilations of a marker image until the intensity profile of the marker fits a mask image. We will be using it to match the structural profile of our mask image. We morphologically open our binarized pre-processed image with a (25, 25) to obtain the marker, this will ensure that all our candidates inside the lung are removed. We select the original binarized image as our mask and perform geodesic dilations until the marker structure is reached. Next absolute difference is taken of the original image and the result image. Figure. 12 shows the geodesic reconstruction process and the difference image which consists of nodule and tissue inside the lung.

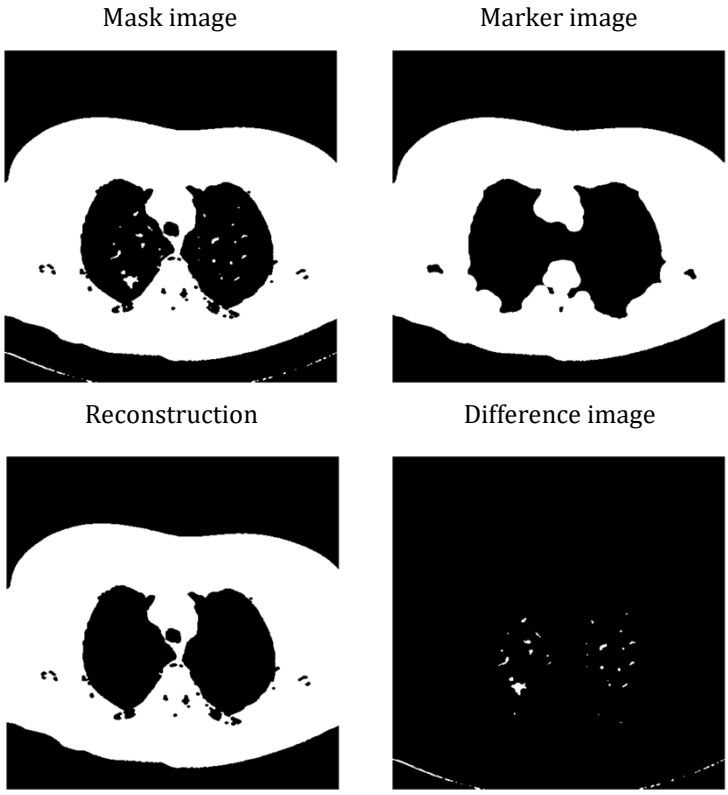

Figure 12. Geodesic Reconstruction Visualization

2. **Lung Mask:** The lung segmentation mask created earlier is used to select the objects from the difference image. This removes the table and any other objects outside the region of interest.

3. **Distance Transform:** In order to remove point like objects a distance transformation was used. The output is then normalized and morphologically eroded with a small kernel of (3,3) and finally binarized with an OTSU again, as distance transform output is not binary. This is to be noted that this step is dependent upon objects being proportional in size, in cases where there are artifacts or

big soft tissues present this step won't work and a size-based selection of objects would be required before this step.

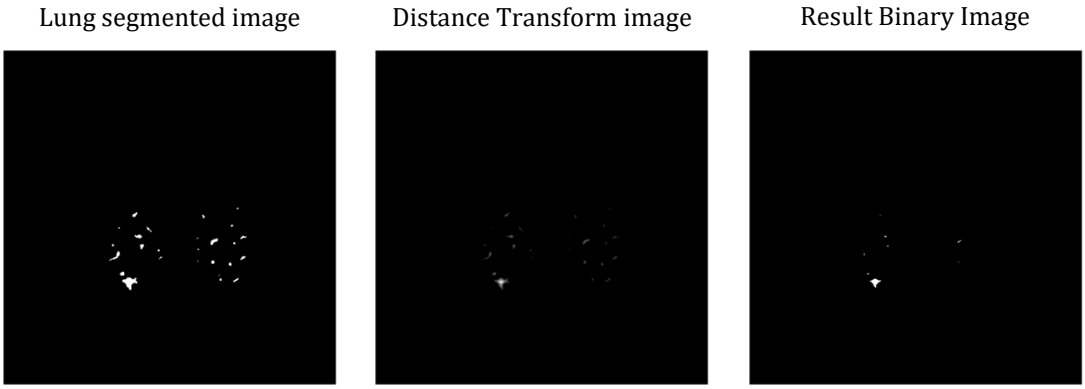

Figure 13. Distance Transformation

We calculated the centroids using the 0^{th} and 1^{st} order moments. We compared the coordinates of our candidates with the ground truth coordinates using a vicinity function. If our candidates lie in the 'd' vicinity of the ground truth we count it as a hit, 'd' is defined as half of the diameter of the nodule (given with the ground truth) plus a bias term which is kept as 2.

$$\text{Sensitivity\_0} = 0.535$$
$$\text{Number of Candidates} = 803$$

The low sensitivity is due to the fact that the dataset used has more than 20% axial images with a wall nodule, and more than 15% axial images with artifacts. This method generates less candidates but doesn't take the wall case into consideration as the lung mask and geodesic dilation removes the wall cases.

Method 2

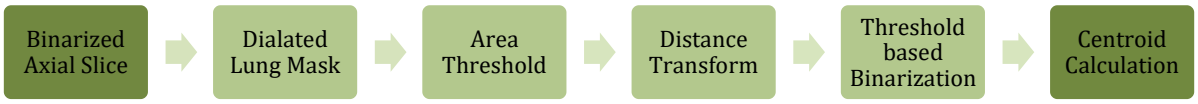

Figure 14. Method 2 Pipeline

Figure. 14 describes the major steps in this method, preparation and transformation steps are not shown but will be explained.

1. **Lung Mask:** The segmented lung mask is morphologically dilated with a (35, 35) kernel to include the wall nodule cases, doing this generates more candidates which includes area outside the lung too, but it was necessary for inclusion of the wall nodule cases. The mask is applied on the binary image to generate the objects.

2. **Area Threshold:** Contour objects are found, and area is calculated for each object, a threshold of 1500 is chosen after careful observation of nodule area and adding a bias to improve robustness. All the contours with an area greater than the threshold are removed.

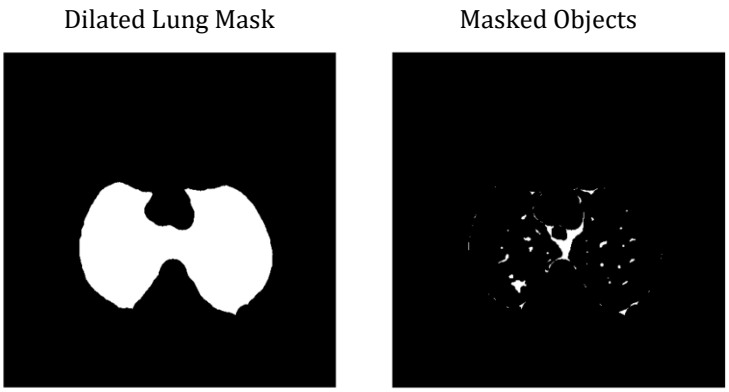

Figure 15.Lung Mask Visualization

3. **Distance Transform:** In order to remove point-like objects a distance transformation was used. The output is then normalized and morphologically opened with a small kernel of (3,3) and finally binarized with a custom thresholding method, as distance transform output is not binary. Area of all the objects is calculated again and two thresholds are hand crafted to select which binarization to use. The area of largest contour and the mean contour area are used as threshold, Figure. 16 explains the custom thresholding.

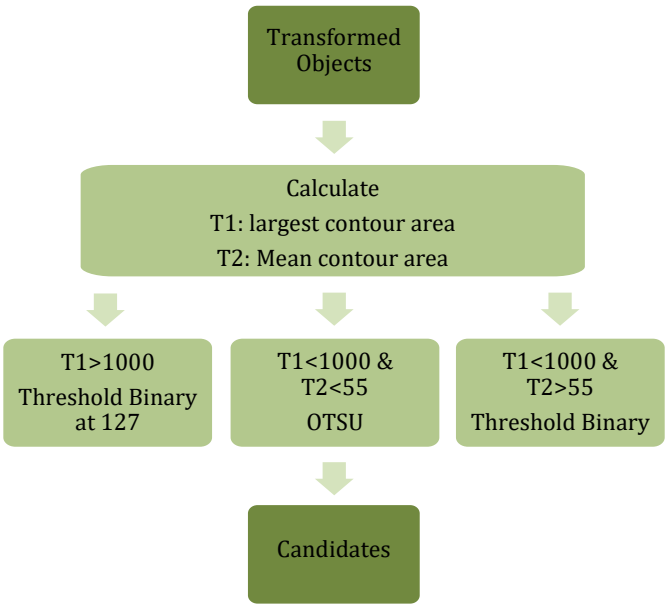

Figure 16. Custom Thresholding for binarization

We calculated the centroids using the 0^{th} and 1^{st} order moments. We compared the coordinates of our candidates with the ground truth coordinates using a vicinity function. If our candidates lie in the 'd' vicinity of the ground truth we count it as a hit, 'd' is defined as half of the diameter of the nodule (given with the ground truth) plus a bias term which is kept as 2.

Sensitivity_0 = 0.8125
Number of Candidates = 2079

The missed nodules were predominantly due to the fact that more than 15% axial images had artifacts. This method has high sensitivity with more candidates.

Finally, we used the second method without the area threshold (will be included in post-processing later) and the custom thresholding, this increases the sensitivity_0 to 0.848.

Post-processing

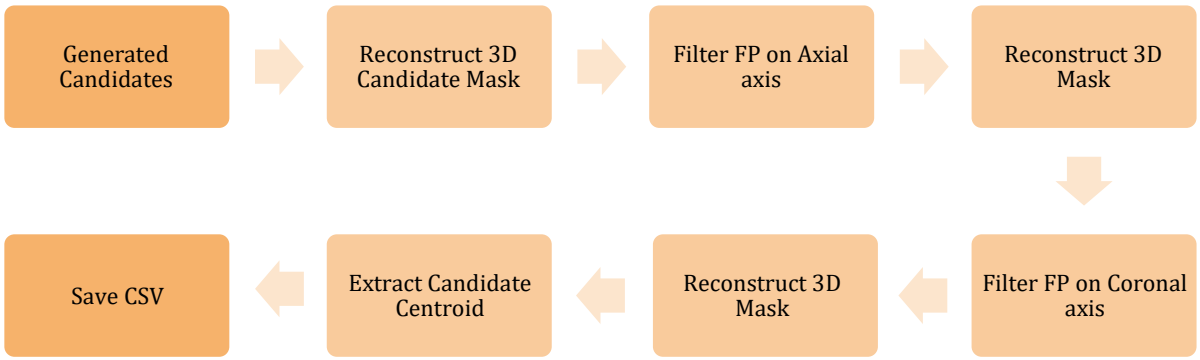

Figure 17. Post-Processing Pipeline

The candidate generation detects nodules quite sensitively, but it results in over 2000 candidates per case for 1 to 3 nodules. Thus, filtering of false positive candidates is required. This is done by the mask images created during the candidate generation.

The binary slices created on the Z-axis (axial) are used first. On each slice the connected components region properties [16] are analyzed, the ones that do not fit the specific requirements are discarded. These requirements are that the candidate must have an area of at least 1500 pixels, eccentricity less than 0.75, and solidity bigger than 0.6. After the filtering on the Z-axis the mask image is reconstructed into a 3D mask image with the same dimensions as the original CT image for further candidate filtering.

On the reconstructed mask image slices are extracted through Y-axis (sagittal). The process is similar on this axis: on each slice the connected components are labeled, and region properties are analyzed. However, the requirements here differ, only solidity and area are considered, because some components tend to have couple pixels wide lines disconnecting them from each other, this is due to the imperfections of the candidate generation and filtering through the Z-axis. After the filtering the mask image is reconstructed again.
The resulting 3D image's connected components are labeled for a last time, from their region properties the centroids of the object are extracted into a data frame as the final candidates.

With this process the candidates are greatly reduced, for CT images with low Z dimension (120-200 slices with ~1000 candidates) a third of the candidates are eliminated, for images with high Z dimension (400-520 slices with >4000) more than 3 quarters of the candidates can be eliminated. The filtered candidates are saved into a csv with their x, y, z, coordinates and the series uid of the case.

For the last step of the candidate post-processing the extracted candidates are compared to the ground-truth in the *annotations.csv* provided by the challenge. Cartesian coordinates of the ground truth centers and diameters are computed from the world coordinate origin and spacing and all the candidates that are in the fall inside one of the ground truth annotation diameters are labeled as a positive candidate (class 1), the rest are labeled as negative candidates. These are concatenated to the csv file later to be used for the machine learning process.

Feature Extraction

For the selection of features many approaches were considered like custom selected features, HoG features, pretrained AlexNet kernel features, but for this project, because of the hardware limitations, we ended up using the handpicked features which are described in this section.

The features were extracted from 48x48 patches around the candidate center from the HU intensity range images. This range was selected because during the data exploration it was found that the biggest nodule in diameter is 40 pixels, thus in order to capture the whole of the nodule and also some surrounding area feature while being divisible into square sub patches, the 48-pixel patch size was selected instead of 40.

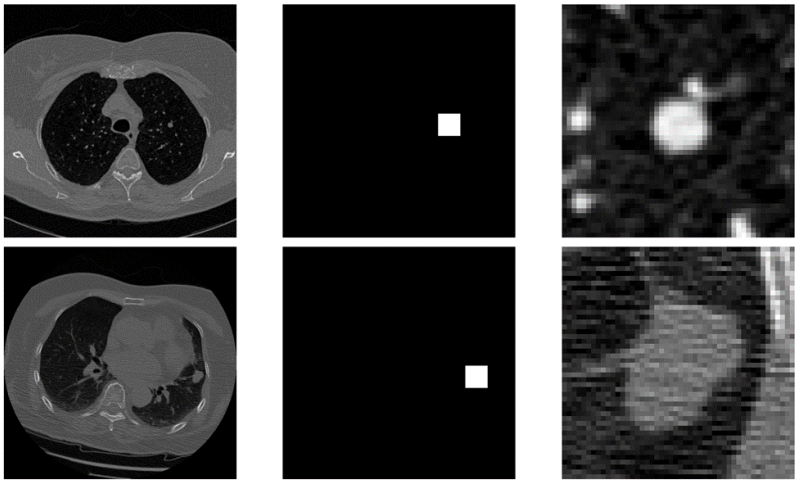

Figure 18. Patch creation for Candidate

Custom features

Handpicked features were selected to capture different domains. Morphological-, pixel-intensity-, texture-, Haar-, and Gabor-features were extracted in order to further reduce the false positive hits with ML. These features are later reduced by feature selection methods but for completeness they were all kept in the csv file for the feature vectors.

Morphological Features

For morphological features first we needed to binarize and select the center object of the patch. For this we have used the previously created lung mask and created the identical patch of it as the analyzed patch and after an otsu binarization the inside lung area is masked in order to remove the falsely binarized areas. After the masking the center object is selected, and the following features are extracted:

- area
- perimeter
- compactness
- eccentricity
- Major axis length
- Minor axis length
- solidity
- extent

Pixel intensity Features

For the pixel intensity features the whole patch is used as it is on the HU intensity range. The selected features are:

- Mean intensity
- Std intensity
- skewness
- kurtosis

Texture Features

For the pixel intensity features the whole patch is used as it is on the HU intensity range. From the patches a gray-level co-occurrence matrix was used to extract the texture features. The selected features are:

- contrast
- correlation
- energy
- homogeneity

Haar Features

For our purpose, the haar features which effectively differentiated the contrast were used. So, we used two haar kernels, i.e. the center surround and the four rectangle kernels which are widely used for object and edge detection. These kernels were used to extract haar feature maps from the image patches and process them at multiple scales. The different scaled patches were combined using a weighted average, more weight was given to smaller scales feature maps as the object of interest was in the center in most patches. The final feature map is then max pooled and average pooled to produce compact, informative feature representations. Figure. 19 shows the visualization of haar features.

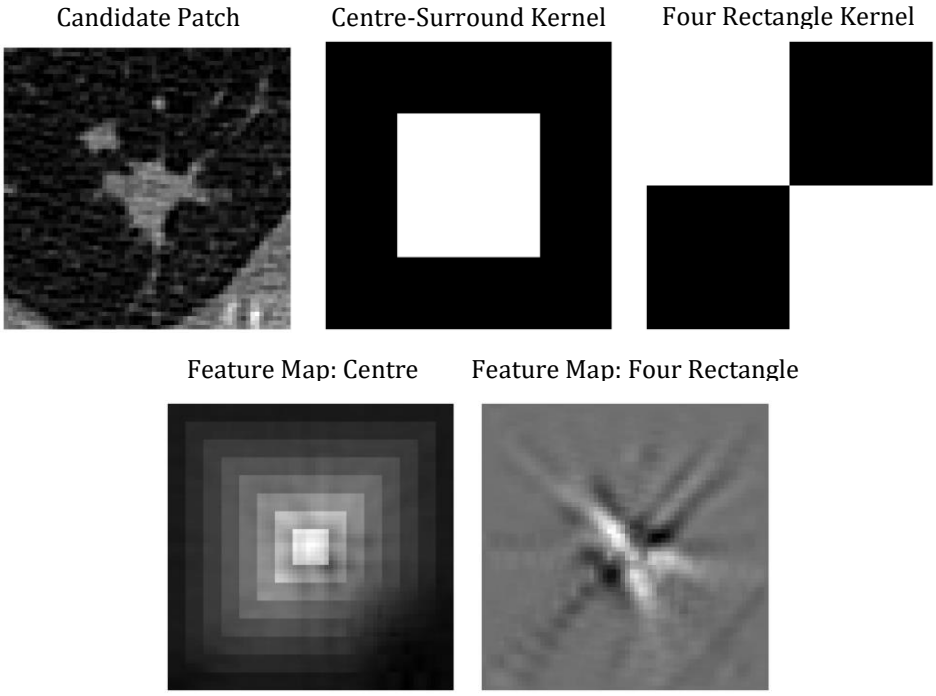

Figure 19. Haar Features Visualization

Gabor Features

Gabor filters are widely recognized for their efficiency in capturing spatial frequency characteristics and edge orientations within an image. Gabor kernels are applied to the patches, and the resulting filtered images are averaged to create a mean feature map. This map is then subjected to both average and max pooling to produce compact, informative feature representations. Figure. 20 shows visualization of the Gabor features.

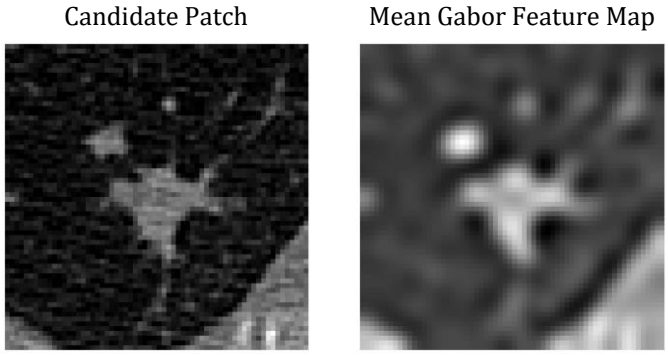

Figure 20. Gabor Features Visualization

The extracted feature vectors are stored in a dedicated csv file which can be later used for the training of the machine learning models doing the false positive reduction.

3. Machine Learning Pipeline

The goal of the machine learning pipeline is to reduce the false positives as much as possible while keeping the recall of the positive class as high as possible. For this task candidates were collected from subset0, subset1 and subset2 for training and subset8 for testing, ground truth annotations were also collected from the annotations given for the LUNA16 challenge.

The candidates which were in the vicinity defined by the ground truth annotation diameters were labeled as class 1, the rest were labeled as 0. The sensitivity of the candidate extraction resulted in the sensitivities shown in Table 1. for all cases in the training and validation subsets.

Subset	Candidates	TP Candidates	GT Candidates	Sensitivity
0	77,509	158	112	0.884
1	64,355	193	128	0.852
8	61,122	227	118	0.873

Table 1. Candidates count and sensitivity per subset

The main pipeline for the false positive reduction consisted of 4 main branches as shown in the figure. 21. Due to the time other techniques took to finish, the first two steps were maintained the same.

Figure 21. Main model pipeline

The models were then analyzed by the confusion matrix showing the TP, TN, FP, FN. But mainly focusing on sensitivity and keeping it high, as well as analyzing the specificity to indicate the number of false positives discarded.

Class imbalance

For training subset0 and subset1 candidates were concatenated and were handled as one dataset. From Table. 1, we can see that in total for 141,864 candidates there are only 351 positive candidates, which means that the dataset is considered a large imbalanced dataset with ~404/1 negative to positive class ratio.

Furthermore, our machine learning algorithms were focused on handling highly unbalanced data. Knowing that many different algorithms were tried: oversampling of the lower-class instances, different undersampling of the higher-class instances, different models and feature selection techniques, and many more. The problem of computation and time-consuming algorithms that seemed promising were discarded. Our models included assigning high initial weights to the lower class (class 1) ranging from 100 to 400 initial weight. And low weights to the higher class (class 0) with a unit weight value.

Data preparation

Under sampling the higher-class instances worked better than any other sampling method. The idea was to try more sophisticated methods to impact the class imbalance, but the other sampling methods either would take a lot of time or were simply giving worse results than the random under sampling method. This method randomly removes instances from the higher-class instances to balance the data.

Normalizing the data before applying other machine learning algorithms will improve their convergence. The standard scaler was used to make the features have a null mean and a unit variance.

Feature Engineering

Since the feature vector consists of 318 features it is necessary to reduce the dimensionality in some form. For these 3 methods were proposed: mRMR, PCA, and RFE.

mRMR (minimum redundancy maximum relevance) is a feature selection method that focuses on minimizing the redundant features, which might have some useful information, but mostly are redundant. In addition, this method maximizes the relevant features to include the main features, it also removes the useless features that will act like noise to the models. The relevance is the mutual information calculated between the features and the target variable. The redundancy is the mutual information calculated between the features themselves.

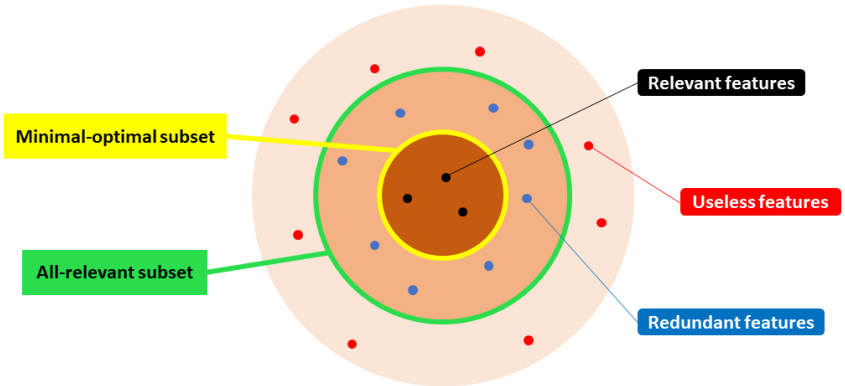

Figure 22. mRMR divided subset

PCA (Principal Component Analysis) is a dimensionality reduction method, it does not select the most important features. Instead, it reduces the dimension in which these features are presented. It does that by calculating the variance of the data in different original axes. The highest variance is assigned to the principal component, and this component represents the new principal axis of the features. The process is repeated, and the number of principal components selected, is the new dimension of the data.

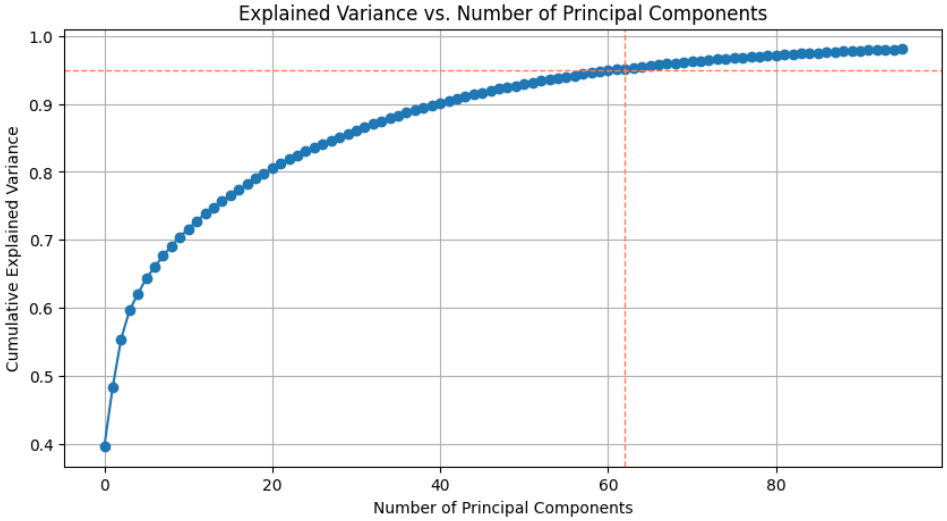

Figure 23. PCA cumulative explained variance

As shown in figure. 23, the cumulative explainable variance, which represents the density and relevance of the information, is higher than 0.9 at around 40 components. It reaches 0.95, equivalent to 95% of the explained variance at 62 components. Which is why the PCA method was implemented having a number of components ranging from 40 to 62 according to the best results.

RFE (Recursive Feature Elimination) is a feature selection technique that recursively applies an ML model to fit the data and selects the best features according to that fit. The model chosen for this fit was Random Forest, which estimated the number of features that best fits the task at hand. The number of features to be selected ranged from 30 to 60 to accommodate most of the information in the features and evade potential useless features.

Model

Since the problem at hand is classification, and the classes are very unbalanced, many models were implemented including Logistic Regression, XGBoost, EasyEnsembles. But following the results, SVM, Random Forest and LightGBM were the 3 models that were used in our main pipeline.
SVM (Support Vector Machines), as indicated previously, was the state-of-the-art technique in this field.

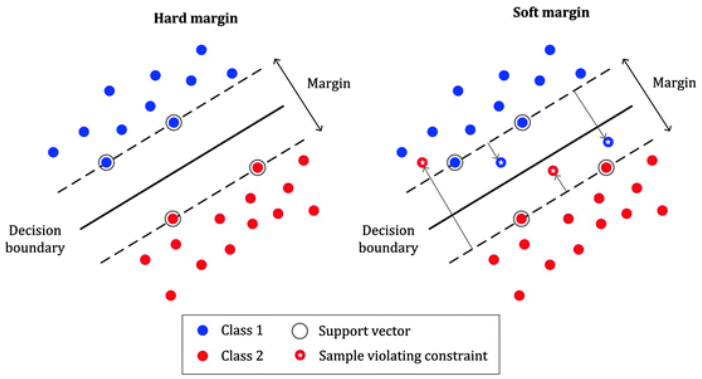

Figure 24. SVM Hard margin vs soft margin

It consists of finding the hyperplane that separates 2 classes in a certain dimension, while keeping the distance between the data instances from each class as far away from each other as possible, creating a margin.

This margin is in more general cases a soft margin, allowing some data points to be inside the margin according to some slack variables. These slack variables ε indicate the freedom of the data points to be present in the margin as long as their sum is within a certain threshold of value C. SVM adapts to the non-linearity by applying a kernel that projects the data into a higher dimension plane where the hyperplane is drawn, then projected back to the initial dimensions.

In our implementation, the SVM was performed with standard hyperparameters, a 42 random initial state, and a class weight equivalent to the dictionary that was initialized, explained in the class imbalance.

Random Forest is the other model that was used in our main pipeline. It consists of an ensemble method, as its name hints, it is a multitude of instances of the Decision Tree. This method is a simple non-parametric method that creates new rules for the dataset to follow which becomes the decision cells, multiple decision cells create a tree, multiple trees create a forest. More rules indicate more complex rules, and that is why RF is a good model for complex data.

A GridSearch was implemented to find the best hyperparameters for this model, the best ones were: {'classifier__bootstrap': True, 'classifier__max_depth': None, 'classifier__min_samples_leaf': 1, 'classifier__min_samples_split': 2, 'classifier__n_estimators': 50}.

The class weights were also initialized with the class imbalance dictionary.

LightGBM (Light Gradient Boosting Machine) is a boosting algorithm. It was designed to be more memory efficient and faster than other gradient boosting methods. This technique achieves this by a leaf-wise growth strategy which is different from the standard boosting growth method, level-wise. A GridSearch was implemented to obtain these following hyperparameters: {'classifier__learning_rate': 0.01, 'classifier__max_depth': -1, 'classifier__n_estimators': 50, 'classifier__num_leaves': 31}
The class weight of the model was also initialized with the class imbalance weights.

All the models performed a stratified cross-validation instead of a normal k-fold cross validation to keep the same class distribution in each fold, to generalize more for our imbalanced dataset.

All the models performed a confidence probability thresholding to play with the threshold of positive and negative predictions. This allowed us to find the best threshold that will give us the best confusion matrix.

Validation on unseen cases

Applying the three main feature engineering, and the three main models, we obtained 9 main results. As shown in table n, below. These obtained results were the application of different models with different feature selection and dimensionality reduction algorithms applied on the subset8 as a test subset.

			Models		
			SVM	**Random Forest**	**LightGBM**
Feature Engineering	**PCA**	Sensitivity	0.644	0.631	0.64
		Specificity	0.498	0.499	0.496
	RFE	Sensitivity	0.635	0.627	0.68
		Specificity	0.53	0.509	0.494
	mRMR	Sensitivity	0.564	0.6	0.604
		Specificity	0.498	0.506	0.512

Table 2. ML Results

These numbers show that cutting the false positives in half results in lowering the sensitivity to the ranges of [0.6; 0.68]. This concludes that this pipeline is greatly affected by the imbalance in the candidate's number. Therefore, further investigation will be applied in the deep learning pipeline as it shows promising results in the handling of the class imbalance.
The ROC curves of these models are shown below, showing the true-positive-rate with respect to the false-positive-rate.

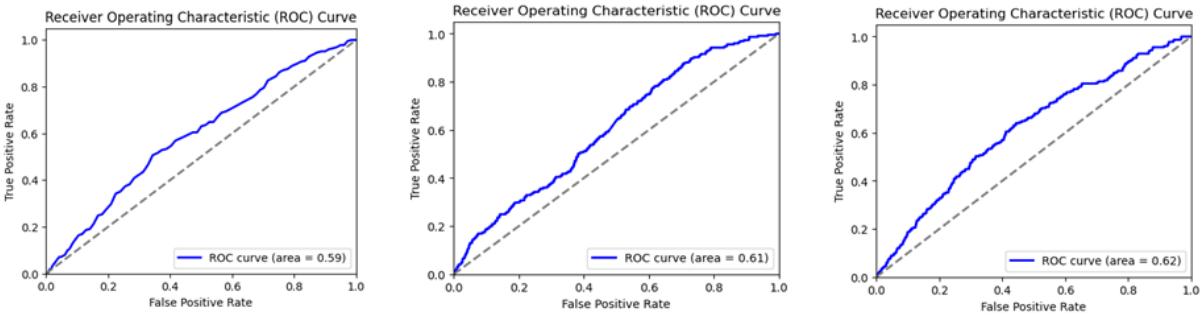

Figure 25. SVM, RFC and LightGBM ROC curves

4. Deep Learning Pipeline

This pipeline is based on the State of the Art from different studies about the high-performance deep learning models in computer-aided systems (CAD), especially in this case for object (nodule) detection in computed tomography images [19]. In the present we aim to perform and compare 4 different Deep Learning-based methodologies: detection by RetinaNet, detection by FasterRCNN, detection by YOLOv8, and image segmentation-based detection with V-Net.

RetinaNet and Faster RCNN: Object Detection Approach

Detection by RetinaNet

As mentioned previously, the class imbalance is widely present in our dataset. Handling this problem can be done in many ways. RetinaNet is a model introduced by FAIR (Facebook AI Research) with the innovative Focal Loss function [21]. This model works great on class imbalance datasets. In the object detection field, RetinaNet is widely used in many cases, performing better than other state of the art models on particular object detection exercises. As other models, like RCNN, YOLO and others tend to dominate the object detection field, class imbalance datasets remain linked to the usage of RetinaNet.
Object detection aims to localize the instance of study, in our case, the nodules. This tool should be, in general, fast and accurate. An image put as an input of the model should give the same image with bounding boxes localizing the objects accurately. Some images could have many instances of the same, or different objects, with bounding boxes.

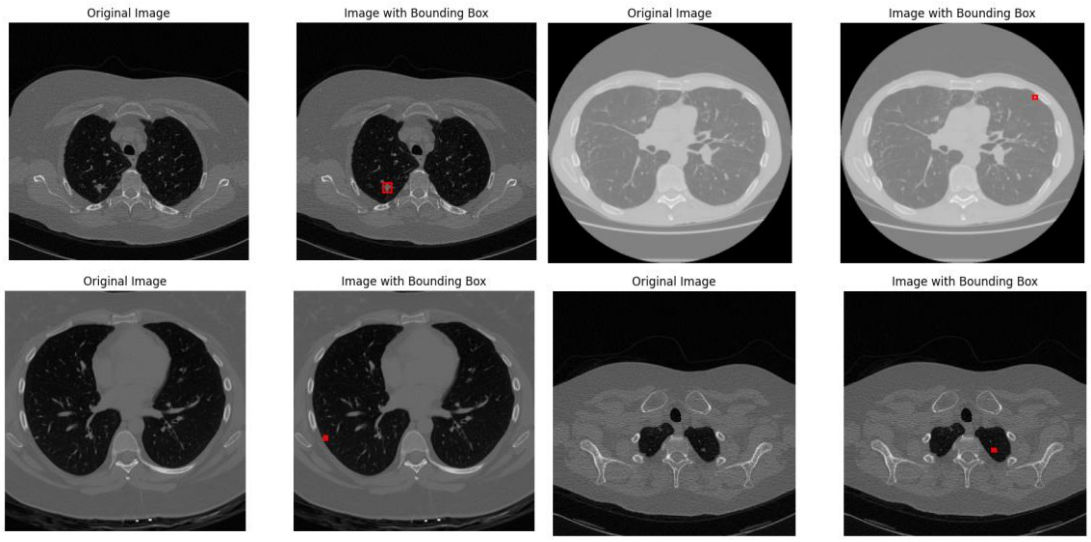

Figure 26. Target bounding boxes

On this approach, the subset0 and subset1 from LUNA16 were the data images for our model, and with the annotations.csv file was extracted the series UIDs to map them in order to match according to each image with nodules. From here we defined the Bounding boxes, being crucial for object detection tasks as they define the area in the image where the lung nodule is located. The Bboxes are defined by converting the 3D coordinates of the nodule's center and diameter into a 2D bounding box on the CT scan slices. In the fig. 26 the Bounding Box for target classes are defined from the original images with nodules and the annotations.
For each CT scan, the slice containing the nodule was extracted and then converted the 3D nodule coordinates into 2D bounding boxes. The images then are resized to a target size (256x256) for uniformity and then stored in the DataFrame. The fig. 27 corresponds to the final images with nodules being localized thanks to the BBoxes and the annotations, being defined as our target data for training our nodule detection model.
RetinaNet, like many object detection models, has a backbone that serves as backbones, for feature extraction, this backbone is usually ResNet based, but can be any backbone. Linked to the feature extraction block is a Feature Pyramid Network block. This block theoretically aims to group low level and high-level features in a way the model can use all of them as information, to learn and adapt to the data. This FPN block is linked to 2 other blocks that output useful information for the detection.

On one hand, the FPN is linked to a regression block that aims theoretically to regress into the values of the height, width and coordinates of the object. On the other hand, this same FPN is linked to a classification block, which aims to classify each case into either an object of a certain class, or no object (Null case). RetinaNet uses anchors, or anchor boxes, which are predefined bounding boxes, as mentioned, the values of these predefined boxes will regress in a good model implementation, to localize the objects. RetinaNet generates multiple anchor boxes for each location on the feature maps, which are then refined by the subnets.

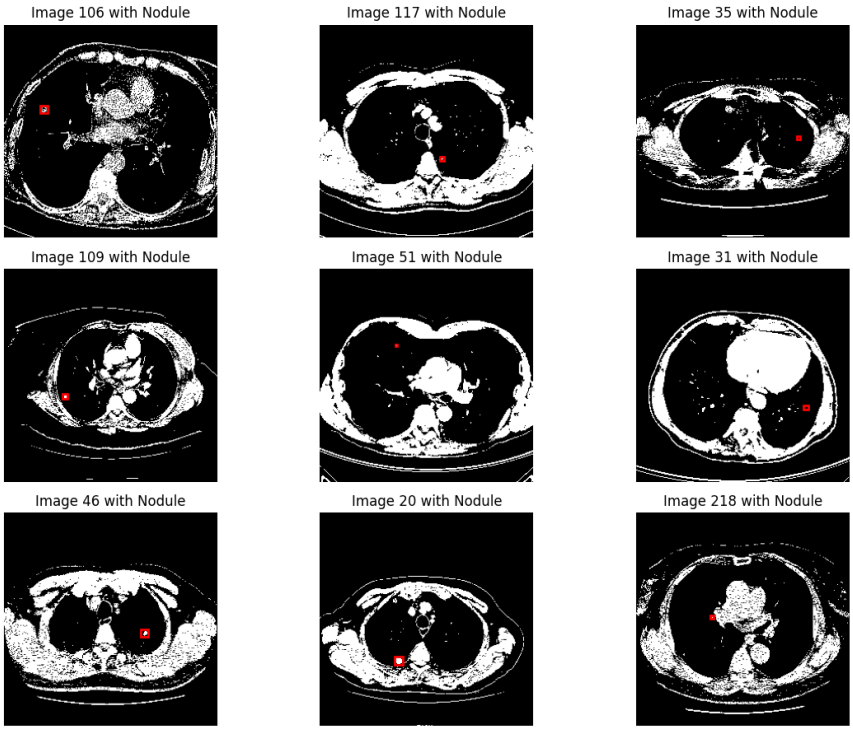

Figure 27. Nodule target bounding boxes

For our model, the main architecture is the exact same: A ResNet-50 as backbone: A convolutional neural network pre-trained on ImageNet, used to extract features from the input image, with batch normalization and activation functions. It extracts the features C3, C4 and C5, in order that the FPN can combine them to create a rich multi-scale feature representation, and then resized to form a feature pyramid that allows detection at different scales (P3, P4, P5, P6, P7). Then for the Classification and Regression Head, Anchor boxes and Objectness Scores are assigned and adjusted based on regression and classification predictions, correspondingly. The Region Proposal Network RPN generates the region proposal from the adjusted anchor boxes and then the Non Maxima Suppression eliminates redundant bounding boxes and only keeps the ones with highest scores, ensuring each nodule is represented by a single bounding box. Finally the remaining Bboxes are considered the detected nodules, each Bbox is associated with a class prediction (nodule or non-nodule image). This scheme architecture is visualized in the figure 28.

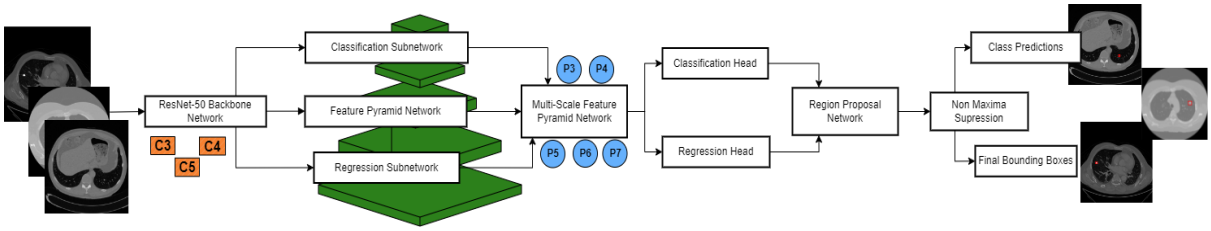

Figure 28. RetinaNet Architecture

Focal loss is the loss function used in RetinaNet. It down-weights the well-classified samples. This way the importance is given to the harder to classify samples, and by analyzing their prediction score, the ones with lowest prediction score values are prioritized with focal loss.

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$$

Equation 2. Focal Loss

Where p is the model's estimated probability, α is a weighting factor for the class. Used for balancing the positive and negative samples. γ is the effect of the focal loss, when its value is 0, the focal loss becomes the standard cross-entropy loss. As γ increases the effect of down-weighting easy examples becomes more pronounced.
The actual computation of Focal Loss in the model happens during the training process when the model's predictions are compared to the ground truth labels. The loss is calculated as follows:

- Classification Loss (Focal Loss): Applied to the output of the RetinaNet Classification Head to compute the loss for class predictions.
- Regression Loss (Smooth L1 Loss): Applied to the output of the RetinaNet Classification Head to compute the loss for bounding box predictions.

Despite the model's potential for handling class imbalance through the innovative Focal Loss function, our approach faced several challenges and did not yield satisfactory results.

- Class Imbalance: While this model is designed to address class imbalance using Focal Loss, the imbalance in the dataset was severe. The number of non-nodule images far exceeded the number of nodule images, which likely affected the model's ability to learn meaningful features for nodule detection.
- Data Quality and Variability: The LUNA16 dataset consists of high-dimensional CT scan images with significant variability in image quality and nodule characteristics. This variability can make it challenging for the model to generalize well, especially when the training data is not sufficiently representative of all possible cases.
- Bounding Box Definition: Defining accurate 2D bounding boxes from 3D nodule coordinates is inherently complex. Any errors in this conversion process could lead to inaccuracies in training, affecting the model's performance. The quality of bounding box annotations directly impacts the effectiveness of the object detection model.
- Computational Resources: Training RetinaNet requires substantial computational resources. Despite using a GPU for training, the process was slow and resource-intensive, limiting the number of experiments and hyperparameter tuning that could be performed within the available time frame.

Detection by FasterRCNN

The second approach of object detection was chosen for a Two-Stage model, more suitable for the case where accuracy is more critical than speed in this case of complexity for object detection, where precise localization and classification are essential [20]. In this case our main objective is to address the challenges that the previous model faced in order to achieve a better, or an actual result. That is why in this approach we are not defining Bounding Boxes as target images, changing the architecture and customizing the neck and head of this model.

This approach starts with creating the target images with the annotations, this time will be with patches as masks images instead of bounding boxes. The masks according to the annotations were created and used in this case as filled rectangle masks as it's shown in Figure 29.

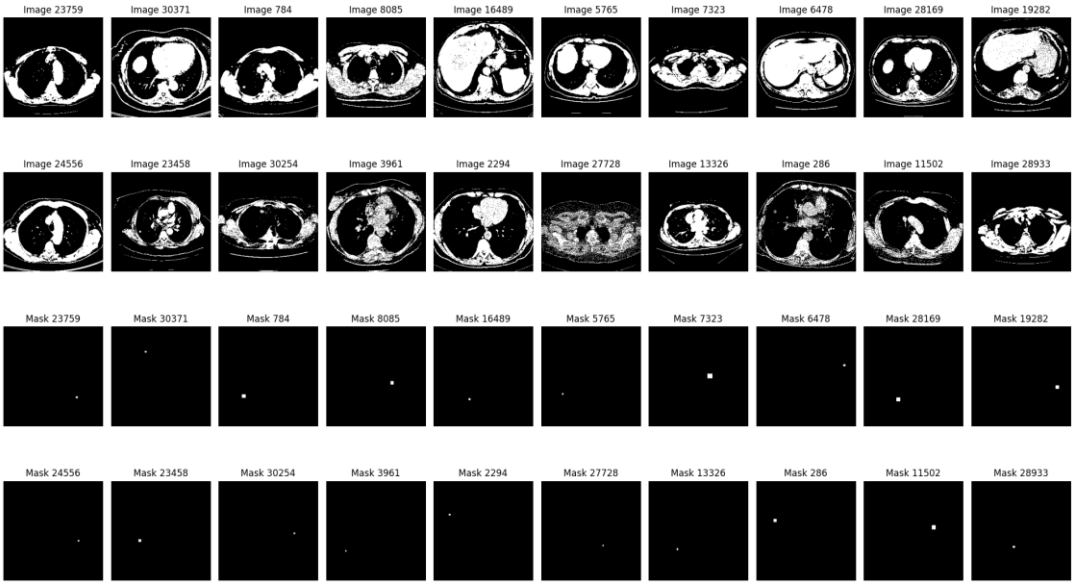

Figure 29. FasterRCNN target masks

The architecture consists of a VGG16 backbone with ReLu Activation and Batch normalization, as it's shown in figure 30. The initial consideration for backbone was ResNet-18, but for adding more variability on the architecture, and theoretical precision [21], the final consideration was the Resnet-18. The model also includes FPN with Depth wise Separable Convolutions, then Region Proposal Network for classification and Regression, and finally a ROI Pooling and Fully Connected Layers that lead to the final Classification and Regression Head. On the output we obtain the class scores for each ROI indicating presence of a nodule or background and the Bounding Box Coordinates around the detected nodules. In this case Focal loss was applied for classification and MSE loss for regression, MAP and IOU as evaluation metrics were included as well as for the RetinaNet model.

Intersection over Union (IoU): IoU is a fundamental metric used to evaluate the performance of object detection models. It measures the overlap between the predicted bounding box and the ground truth bounding box.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

Where:

- *Area of Overlap*: The area where the predicted bounding box and the ground truth bounding box intersect.
- *Area of Union*: The total area covered by both the predicted and ground truth bounding boxes.

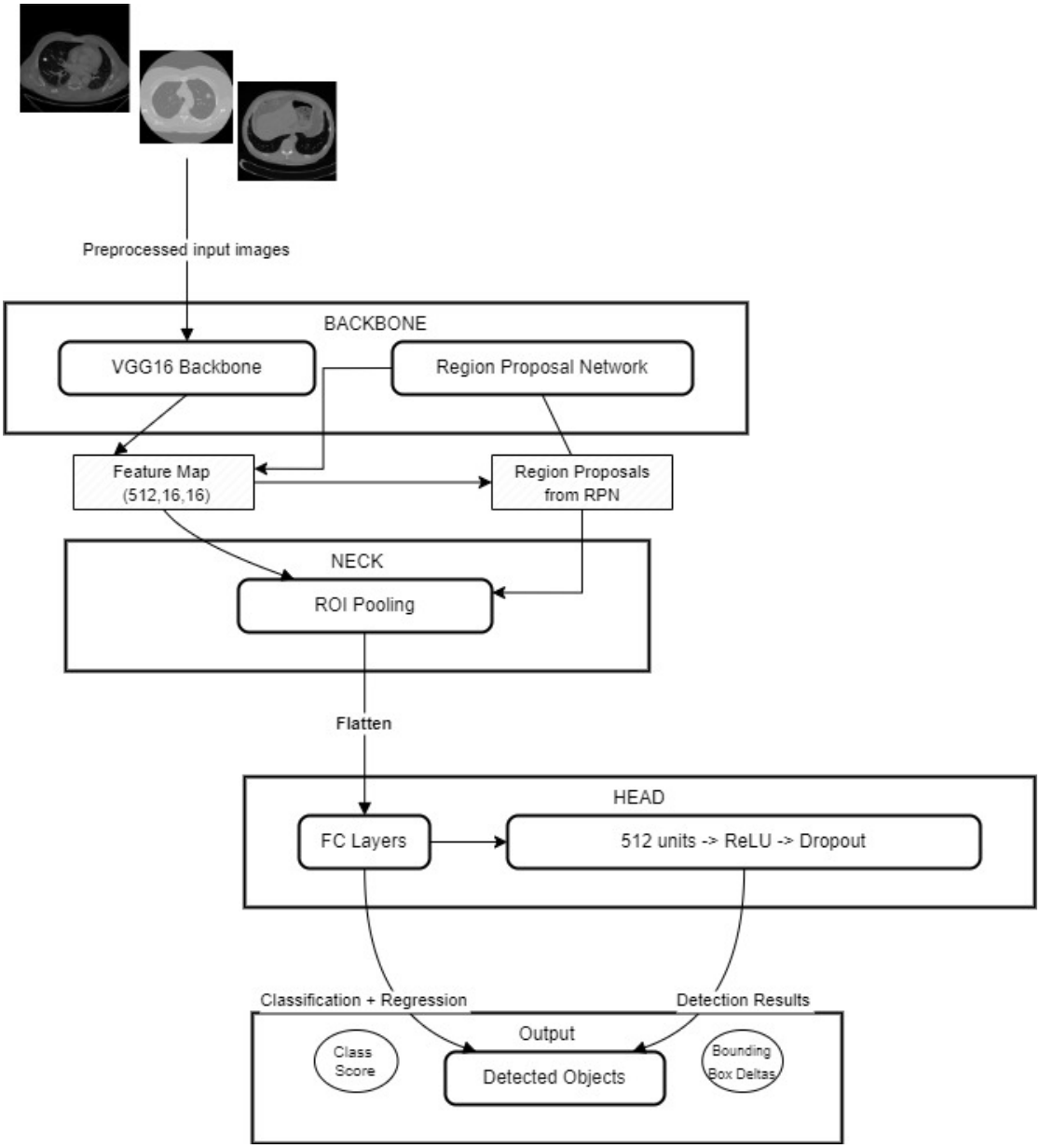

Figure 30. FasterRCNN model architecture

Mean Average Precision (mAP): mAP is a comprehensive metric used to evaluate the accuracy of object detection models across different classes and IoU thresholds. It combines precision and recall providing a single performance measure.

$$mAP = \frac{1}{N} \Sigma \sum_{i=1}^{N} APi$$

Where:

- N: The number of object classes.
- APi: The average precision for class i.

Average Precision (AP): AP is calculated as the area under the Precision-Recall curve for a particular class. It summarizes the precision and recall for different thresholds.

The attempt to use Faster RCNN for lung nodule detection also highlighted several key challenges:

- Computational Resource Constraints: Object detection models like Faster RCNN require substantial computational power and memory, which are beyond the capabilities of our current setup.
- Complexity of Metrics: Applying metrics such as mAP and IoU proved difficult in this context, further complicating the evaluation process.
- Despite the approach and theoretical efficient methods, the proposed optimizations did not yield the desired improvements in training efficiency. The time taken to fetch each item from the dataset remained high, resulting in prolonged training times. Additionally, the inherent complexity of object detection tasks, combined with our limited computational resources, made it challenging to effectively apply metrics such as Mean Average Precision (mAP) and Intersection over Union (IoU).

Exploring alternative architectures such as YOLO or EfficientDet, will offer better performance for this specific task.

You Only Look Once: Object Detection

We leverage YOLOv8 [22] (You Only Look Once, version 8) for detecting lung nodules. YOLOv8 is a state-of-the-art object detection model known for its high speed and accuracy. It operates by dividing the image into a grid and predicting bounding boxes and class probabilities for each grid cell simultaneously. In the Yolo v8 models, there are many sub-models including Yolo V8n (nano), Yolo V8s (small), Yolo V8m (medium), Yolo V8l (large), Yolo V8x (extra-large). In our project, we only tried the small Yolov8 small because in theory they perform quite the same way on small datasets.

Custom Dataset & labels

Since our annotations were composed of the centers of the nodules and its diameters, using this information as labels in the Yolo v8 model is not possible, Yolo models use a labeling system as many other detection models, the COCO label format for object detection. Yolo formats differentiate from COCO label format in a slight way where we can't group all the object information inside one text file. Each image_name.jpg should have an image_name.txt file as a label, so the creation of our custom dataset was done independently from the yolo implementation. Inside each txt file, if an object exists in the assigned image name, the center coordinates as well as the box's width and height, and the class value (which is only class 0 in our case as a nodule object).

Preprocessing of the 2D axial images

As explained in the previous pipeline, the nodules biologically only exist inside the lungs, in the objective of enhancing the performance of the model, lung segmentation was performed on all 2D images in the Yolo dataset. Furthermore, a median blur was applied as noise reduction with a 3x3 kernel.

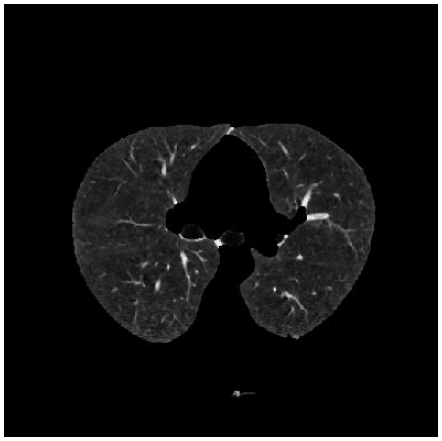

Figure 31. Segmented Axial Lung Slice

Training

Google Collab Pro was used to train the Yolo v8 custom Dataset. Training on only subset0 images, there were just above 100 nodules in approximately 18000 2D axial images. This imbalance between the null cases 99.5 % to 0.05% of instances made the Yolo model perform very poorly.

Using a more balanced dataset, with randomly chosen null cases and all the available nodules in all 10 subsets, we made a 50% null-cases imbalance custom dataset. This dataset had 1469 train images, 450 validation images, and 208 test images for a total of 2127 images.
The training of the Yolo v8 'small' was done on 20 epochs, 4 batch size.

The hyperparameters tuned were choosing the SGD optimizer (known to perform better than other optimizers for the detection tasks) a learning rate of 0.001 and assigning the Boolean value 'True' to the cos_lr to allow the use of cosine to enhance learning performance.

During training the metrics used for the assessment of the model performance on each epoch were the precision, recall, average precision, mean average precision (50) and mean average precision (50-95) with a 5 stepwise.

Results

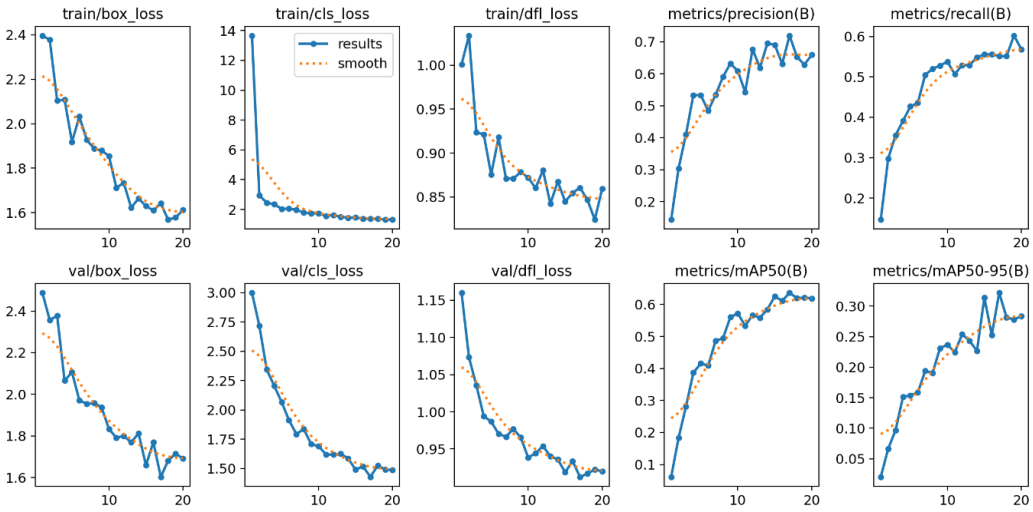

Figure 32. YOLOv8 Results

The curves of the loss and the curves of the precision, recall and mAP seem to converge showing the model is learning and it is converging. The 0.6 mAP50 shows a good performance of the detections being made on the validation set.
Applying the model on the test data shows two major main results. At first, in general, the detected nodules are being detected very accurately as the boxes of detection of the model are very close to the ground truth boxes.

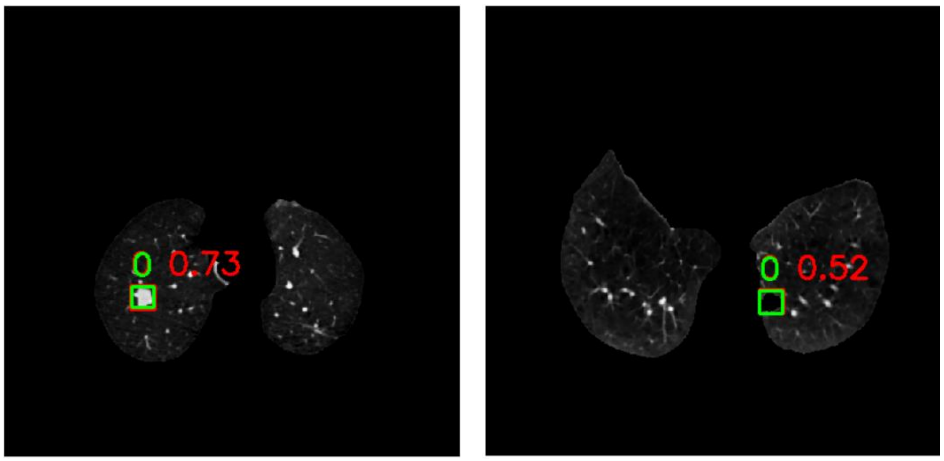

Figure 33. Prediction with ground truth YOLOv8(left) Prediction and ground truth for incorrectly segmented lung(right)

Discussion

On second hand, the model is also detecting the segmented nodules during the lung segmentation process, this shows the power of the learning of the yolov8 for object detection.
In the end, if the null-case ratio was not high, the yolov8 would have performed well for this case, even if it's the case of medical CT images. But the fact that it only worked on customized datasets to evade the problems and not solve them, yolov8 remains with a surprising performance, but it was not included in our main pipeline, it seemed that fine tuning object detection models was a complex loop, from unsatisfying class imbalance affected results to computationally draining methods, another approach was implemented, a different approach from object detection.

Segmentation-based detection using a Vnet architecture

The challenge involves the automated identification of nodules from 445 volumetric CT scans of lungs, sourced from the LIDC-IDRI database. A segmentation-based detection algorithm utilizing a V-Net architecture is employed to address this, with the dataset systematically divided into training and testing subsets.

Previous work

The idea of using a Vnet for Volumetric Medical Image segmentation was first implemented[10]. The paper introduces a V-Net, a fully convolutional neural network designed for 3D medical image segmentation, specifically targeting prostate MRI volumes. Unlike previous 2D approaches, V-Net processes entire 3D volumes and uses volumetric convolutions to capture spatial context effectively. The authors introduced a novel objective function based on the Dice coefficient, which is specifically designed to handle the imbalance between foreground (prostate) and background voxels. This function improves the network's ability to accurately segment the prostate by optimizing the Dice coefficient during training, thereby focusing on achieving better overlap between predicted and ground truth segmentations. Results

demonstrate that V-Net achieves high segmentation accuracy while significantly reducing processing time compared to other methods.

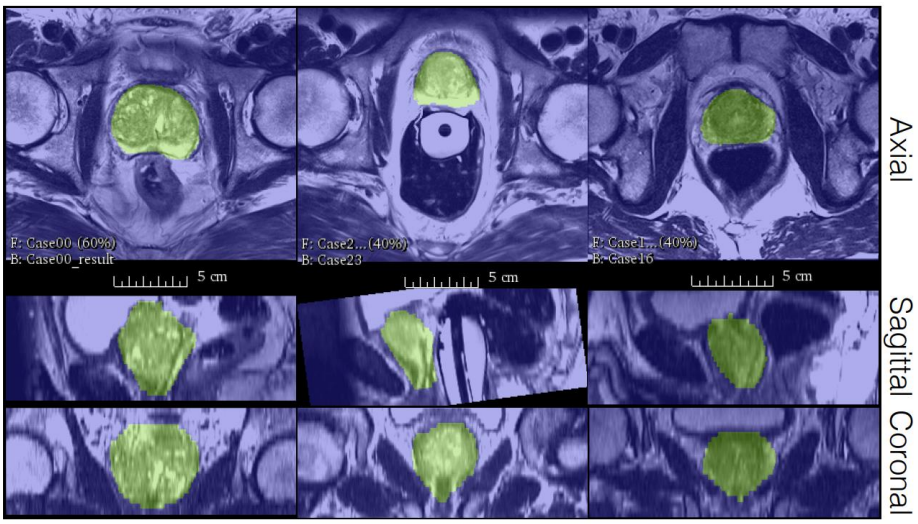

Figure 35. Segmentation using VNET on PROMISE 2012 dataset

The above figure depicts the trained architecture tested on 30 MRI volumes from the PROMISE 2012 dataset depicting prostate regions whose ground truth annotations were not provided.

An improved V-Net [9] model for lung nodule segmentation in CT images, integrating pixel threshold separation and an attention mechanism to enhance accuracy. The novel method included a pixel threshold separation module (Dig_Sep) for better feature extraction by dividing images based on pixel intensity thresholds, and a 3D-Convolutional Block Attention Module (3D-CBAM) to focus on critical features by enhancing channel and spatial attention.

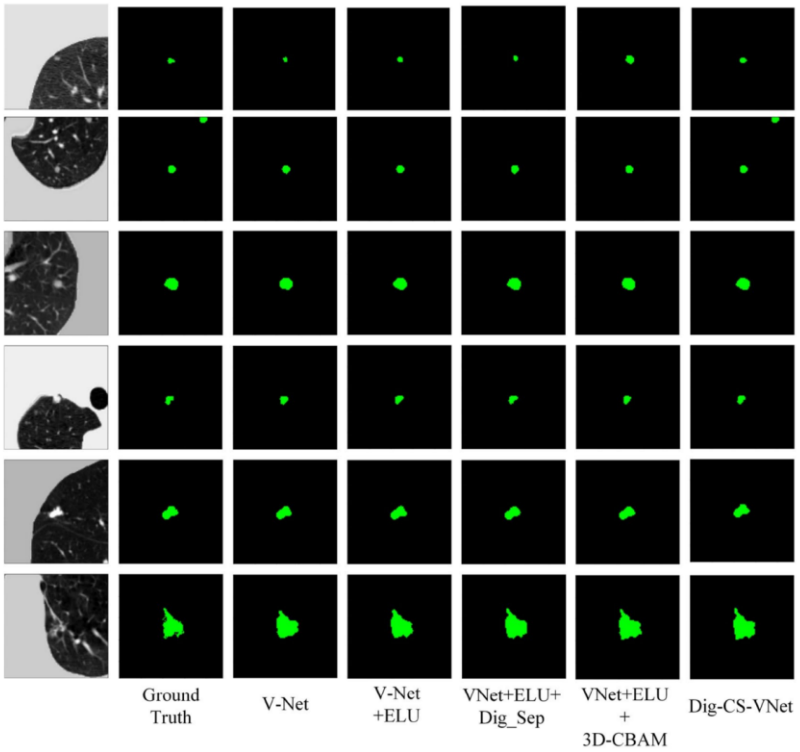

Figure 36. Comparative study between the Dig-CS-VNet and different architectures for Luna16 dataset

The above figure shows that Dig-CS-VNet, showed superior performance on public datasets LUNA16 and LNDb, with Dice similarity coefficients of 94.9% and 81.1%, and sensitivities of 92.7% and 76.9%, respectively. These results outperformed most existing UNet architectures and are comparable to manual segmentation by medical technologists, highlighting the model's effectiveness in clinical applications.

Our proposed method for processing and segmenting the 3D CT scans of Lungs integrates key techniques from the V-Net and Dig-CS-VNet papers. We begin with mask data extraction from the ground truth labels to enhance image quality for segmentation, followed by splitting 3D image volumes into 2D slices for efficient processing. The core segmentation step employs a 3D Vnet architecture, inspired by the volumetric approach of V-Net, to predict segmentation masks for each slice, which are then reassembled into a 3D volume. We draw from Dig-CS-VNet by incorporating slice splitting for efficient data handling and potentially utilizing attention mechanisms to enhance feature focus during segmentation. Additionally, we adopt data augmentation techniques from V-Net, such as random non-linear transformations and histogram matching, to further improve our model's robustness. This integration of volumetric CNN architecture, slice splitting, and advanced processing techniques enhances accuracy and efficiency in our 3D medical image segmentation task.

Methodology

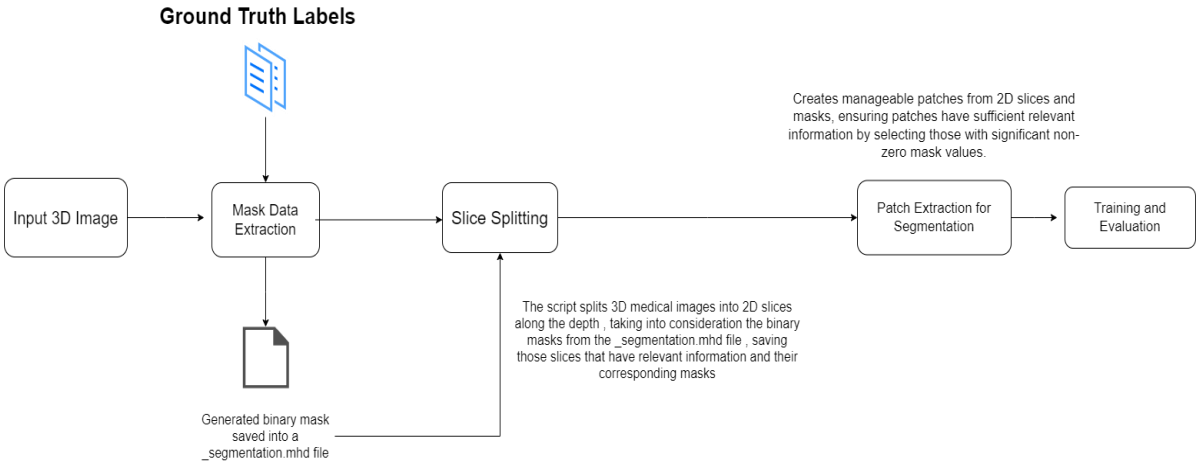

Figure 37. Segmentation Pipeline

The proposed methodology for processing and segmenting 3D medical images is implemented through a series of steps. We begin with mask data extraction from a CSV file containing the ground truth labels, enhancing image quality for segmentation through preprocessing techniques like noise reduction and contrast adjustment. This is followed by splitting 3D image volumes into 2D slices for efficient processing. Next, we process these 2D slices to extract the necessary features. During the training phase, we apply data augmentation techniques such as random non-linear transformations and histogram matching to the processed slices to enhance the training dataset and improve model robustness. These augmented slices are then fed into a 3D CNN defined in the V-Net architecture to train the model to predict segmentation masks, which are later reassembled into a 3D volume. The trained model is tested on unseen 3D medical images, with performance metrics like the Dice similarity coefficient and sensitivity used to evaluate accuracy. Additionally, we incorporate attention mechanisms to enhance feature focus during segmentation. We will be breaking down each step one by one and providing a detailed outline to our model.

Data pre-processing

The data pre-processing pipeline includes extraction of mask data from the ground truth labels, performing some preprocessing tasks like noise reduction, contrast enhancement, and Region of Interest or ROI Segmentation before the segmented slices and the masks are fed into a 3D Vnet architecture for prediction of the Lung Nodules.

Mask Data Extraction

In medical imaging, world coordinates refer to the actual spatial position of anatomical structures within the body. These coordinates are measured in millimeters and describe the location of structures relative to a reference point in the imaging space. Medical scanners create regular, rectangular arrays of points and cells that start at the upper left corner. The i-axis increases to the right, the j-axis to the bottom, and the k-axis backward as shown in the figure below:

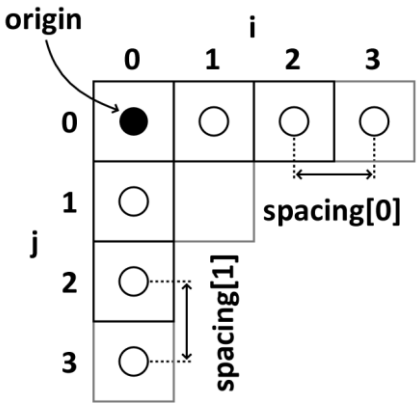

Figure 38. World coordinate system

The pipeline begins by loading 3D medical images and their corresponding nodule annotations from a CSV file. The annotations include the nodule locations in world coordinates (real-world spatial coordinates in millimeters). To generate accurate masks, the world coordinates must be converted to voxel coordinates. Voxel coordinates refer to the position of a voxel within the 3D image grid. The conversion uses the voxel spacing (the distance between adjacent voxels in millimeters) to translate real-world measurements into the image grid. The conversion formula is as follows:

$$\text{voxel\_center} = \left(\frac{\text{world\_x}}{\text{voxel\_spacing}_x}, \frac{\text{world\_y}}{\text{voxel\_spacing}_y}, \frac{\text{world\_z}}{\text{voxel\_spacing}_z} \right)$$

Equation 3. World coordinates to Cartesian coordinates

We converted the voxel coordinates to create a binary mask. This involves calculating the nodule's diameter in voxel units using the formula below:

$$\text{diameter\_voxels}_x = \frac{\text{diameter\_mm}}{\text{voxel\_spacing}_x}$$

Equation 4. Nodule diameter in Cartesian system

applying the same for y and z dimensions and then determining the range of voxels to be set to 1 within the mask using the formula below:

$$
\begin{aligned}
z_{\min} &= \text{voxel\_center}_z - \frac{\text{diameter\_voxels}_z}{2} \\
z_{\max} &= \text{voxel\_center}_z + \frac{\text{diameter\_voxels}_z}{2} \\
x_{\min} &= \text{voxel\_center}_x - \frac{\text{diameter\_voxels}_x}{2} \\
x_{\max} &= \text{voxel\_center}_x + \frac{\text{diameter\_voxels}_x}{2} \\
y_{\min} &= \text{voxel\_center}_y - \frac{\text{diameter\_voxels}_y}{2} \\
y_{\max} &= \text{voxel\_center}_y + \frac{\text{diameter\_voxels}_y}{2}
\end{aligned}
$$

Equation 5. Mask Coordinates

and setting the voxels within this range to 1 to create the mask.

The script iterates through each subset of image files, using a function the correct file paths are located based on case IDs from the annotations. For each image, the create_nodule_mask function is called to generate and save the binary mask in a *filename*_segmentation.mhd file stored in a mask folder.

Slice Splitting

The proposed method then processes 3D medical images and their corresponding segmentation masks to prepare the data for training segmentation models. Initially, it imports necessary libraries for medical image processing, image manipulation, numerical operations, and file handling. The script identifies the range of slices in the depth dimension of a 3D image that contains non-zero values to determine the region of interest (ROI). It then resizes the 3D image to a new spacing to standardize the voxel size across images and normalizes the intensity values of the images to the range 0-255. The script also truncates intensity values outside a specified range and normalizes the images.

In the process of splitting 3D images into 2D slices, the 3D image volume and its mask are divided into individual 2D slices along the z-axis. If the 3D image has dimensions (D, H, W) (D, H, W)(D, H, W) (Depth, Height, Width), the slicing process creates DDD 2D slices, each with dimensions (H, W)(H, W)(H, W). This is achieved using the slicing operation: **slices = [image[:,:, i] for i in range(image. shape[2])].**

The main processing function initializes parameters to determine the number of slices to expand before and after the ROI. It iterates through subsets and image files, loading and truncating each image. If the z-spacing of the segmentation mask is greater than 1.0, the script resizes both the image and the segmentation mask to a z-spacing of 1.0. It then extracts the lung mask from the segmentation mask, determines the ROI range, and expands this range by the specified number of slices. The images and masks are clipped to this ROI range. Finally, the processed slices are saved as 2D BMP files. This comprehensive process ensures that each 3D image volume's slices and masks are accurately aligned, standardized, and prepared for effective segmentation training, following the initial mask data extraction step.

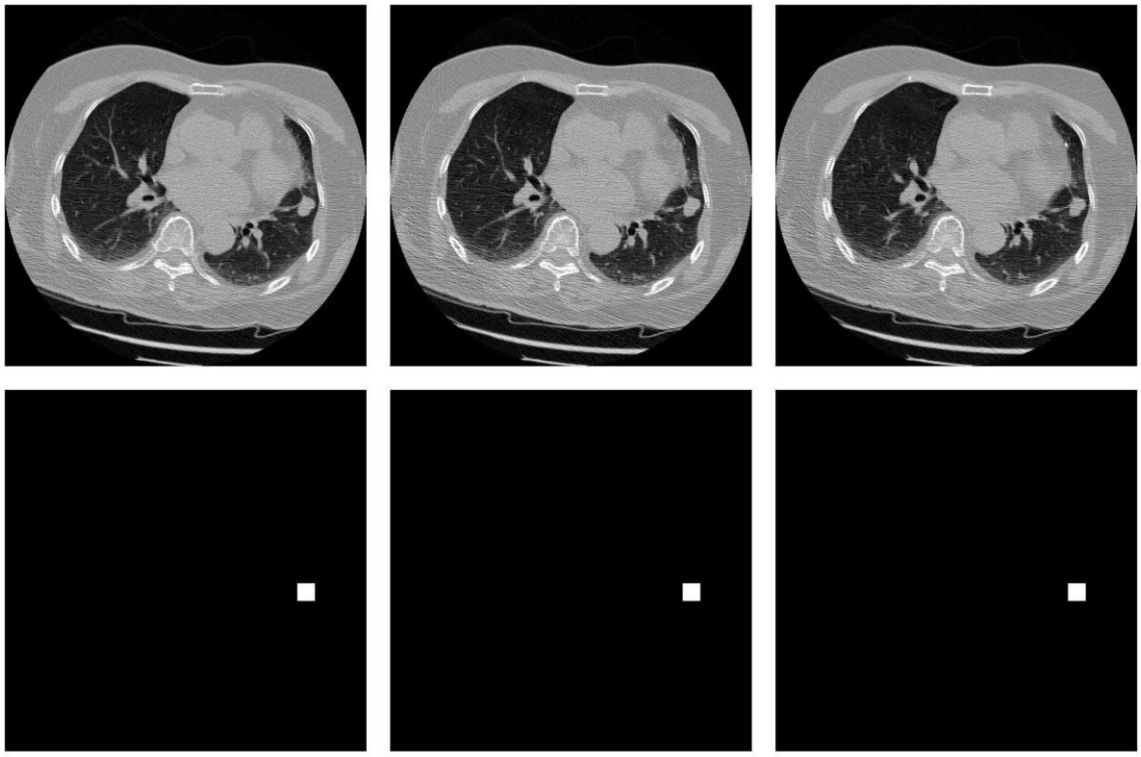

Figure 39. Extracted slices and masks

The diagram above shows the expanded slices and their corresponding masks saved into a folder for further analysis.

Patch Extraction for Segmentation

The method then takes these 2D slices and further processes them by generating sub-images (patches) and corresponding masks. It uses a sliding window approach to extract patches, ensuring that each patch contains sufficient relevant data based on a threshold of non-zero values in the masks.

The process begins by calculating the dimensions and strides for patch extraction, ensuring that the patches are correctly sized and overlap as needed. Specifically, the script calculates the patch dimensions and strides using the provided patch size and the number of patches along the x, y, and z dimensions. It then iterates through the image and mask arrays, extracting patches using a sliding window approach.

A function extracts patches that meet a threshold of non-zero values (set to ensure each patch contains significant information). If a patch contains enough relevant data, it is added to the list of patches. These patches are then reshaped and returned as arrays of sub-images and sub-masks.

A function is responsible for saving the generated patches to disk. It iterates through the generated patches and uses OpenCV's cv2.imwrite to save each slice as a BMP file, ensuring the directories for saving images and masks exist and creating them if necessary. The file paths are organized based on the sample index and patch number, providing a clear and structured way to store the data.

A function handles the overall process of preparing 3D training data. It initializes parameters such as image dimensions and the number of samples. It loads 2D slices from source directories using OpenCV, reshapes them into 3D arrays, and calls save_image_and_mask_patches to generate and save the patches. This function ensures that all samples are processed, and the patches are saved in the specified directories.

Finally, the prepare_nodule_detection_training_data function defines high-level parameters like the image dimensions, number of samples, and directories for source images, masks, and training data. It calls prepare_3d_training_data to process and save the training data, ensuring that the entire dataset is prepared for effective training of segmentation models. This comprehensive process ensures that the 3D image volumes and their masks are accurately divided, augmented, and saved in a structured manner, facilitating robust model training.

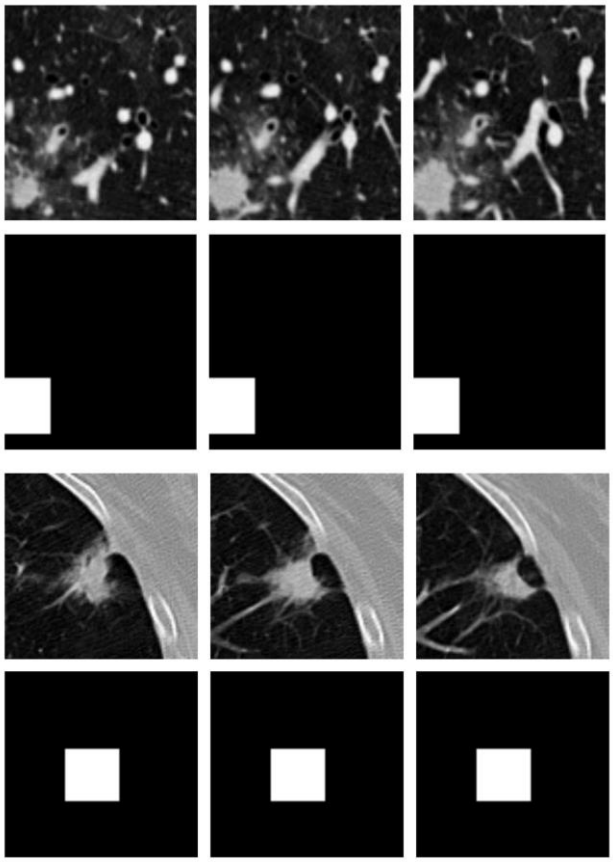

Figure 40. Extracted Patches used for training

The above diagram shows the extracted patches from the segmentation code that is used to train the model.

Model Architecture and Training

Using the data preprocessing steps above, we will now use the patches of the image and the masks that were created to train an end-to-end 3D Vnet architecture. The model architecture is inspired by the original work by Milletari et al in their paper V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation in 2016. The network is trained on a dataset of prostate MRI volumes.

The V-Net architecture proposed consists of two main parts: a compression path and a decompression path. The compression path begins with an input layer that accepts a fixed-size volume of 128x128x64 voxels. This path includes several convolutional stages, where each stage contains one to three convolutional layers using 5x5x5 volumetric kernels to process the data. These layers progressively reduce the spatial resolution and increase the number of feature channels through down sampling operations, which involve convolutions with 2x2x2 kernels applied with a stride of 2, effectively halving the spatial dimensions at each stage. To enhance the training efficiency and performance, the architecture employs residual learning; the input to each stage is added to the output of the last convolutional layer within that

stage, forming a residual connection. This design helps in learning residual functions, which improves convergence speed and performance. Throughout the network, Parametric Rectified Linear Unit (PReLU) non-linearities are applied to introduce non-linearity.

The architecture that is proposed in this paper is as follows:

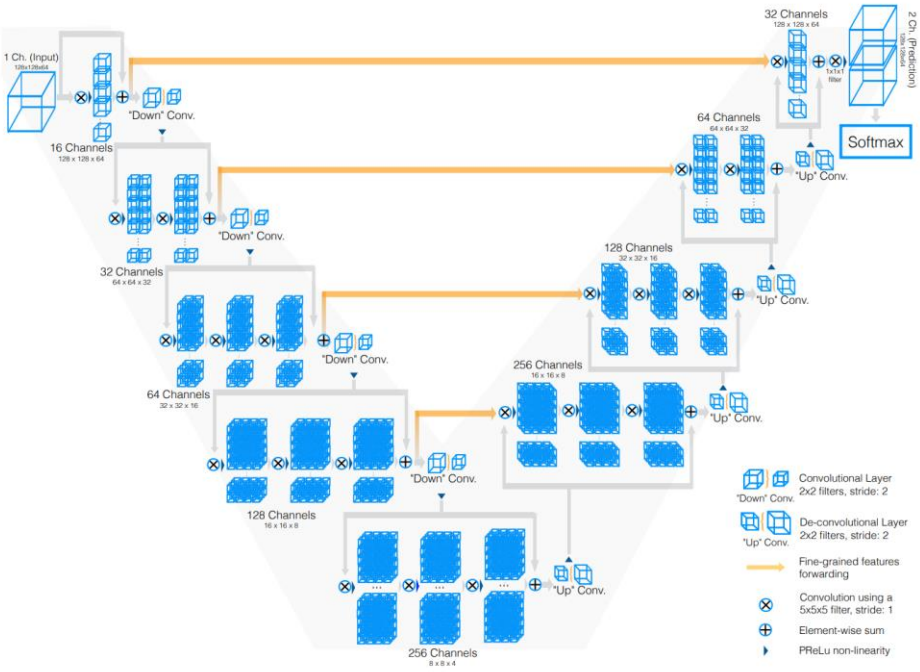

Figure 41. A previously used VNet on PROMISE 2012 Dataset

On the decompression path, the network aims to restore the original resolution of the input volume. Feature up sampling is achieved using de-convolutions with 2x2x2 kernels and a stride of 2, doubling the spatial dimensions at each stage. This path also includes one to three convolutional layers at each stage with 5x5x5 kernels, similar to the compression path, but focused on increasing spatial resolution. To preserve detailed spatial information, fine-grained features from the early stages of the compression path are forwarded to the corresponding stages in the decompression path through horizontal connections. This step is crucial for maintaining high-resolution details that might otherwise be lost during down sampling. The output layer employs 1x1x1 convolutions to produce two output channels, matching the size of the input volume, and applies a SoftMax function voxel-wise to generate probabilistic segmentations for foreground and background regions.

The network optimizes a Dice coefficient-based loss function, which is particularly effective for handling datasets with a significant imbalance between foreground and background voxels.
Keeping this architecture in mind, we designed a 3D Vnet architecture for our Lung Nodule Segmentation-based detection task and also utilized a Dice coefficient-based loss function.

Architecture

The V-Net model designed for this task, as defined in the vnet_architecture.py code, begins with an input volume of size 96x96x16 patches from the previous step. The encoder (compression path) consists of multiple convolutional blocks where each block utilizes the conv_bn_relu_drop function to apply 3D convolution, batch normalization, ReLU activation, and drop out. Specifically, the convolution operation involves applying a 3D convolution, followed by normalization and ReLU activation. Weights and biases

are initialized using Xavier initialization. The first down sampling operation uses the down sampling function, which applies a 3D convolution with stride 2, reducing spatial dimensions. This process is repeated for three more convolutional blocks and corresponding down sampling steps, progressively reducing the spatial resolution.

In the decoder (decompression path), the first up sampling is achieved using the up sampling function, applying a 3D deconvolution with stride 2 to increase spatial dimensions. The up-sampled features are then concatenated with corresponding features from the encoder path using the crop_and_concat function, preserving spatial information. This is followed by a convolutional block similar to those in the encoder. The process of up sampling, concatenation, and convolutional operations is repeated two more times. The second up sampling increases spatial dimensions further, and the features are concatenated with corresponding encoder features, followed by convolutional operations. The third up sampling also increases spatial dimensions, and features are concatenated once more with encoder features and passed through convolutional operations.

The final convolutional layer applies a 1x1x1 convolution to produce the final segmentation map. SoftMax activation is then applied to this output to generate probabilistic segmentations for each voxel. The final output is the segmented 3D volume with probabilistic values indicating the presence of the target structure in each voxel. The model optimizes a Dice coefficient-based loss function to handle class imbalance, where the foreground region is much smaller than the background. The Dice coefficient D.

$$D = \frac{2\sum p_i g_i}{\sum p_i^2 + \sum g_i^2}$$

Equation 6. Dice Coefficient

where pi and gi are the predicted and ground truth binary volumes, respectively. This loss function ensures that the learning process focuses on both foreground and background regions effectively. The gradient of the Dice coefficient to the predictions is computed to facilitate effective backpropagation and optimization.

Training

The Vnet architecture proposed is trained on 311 3D CT scan images of Lungs that are sliced along its depth to create variable slices for every image and their corresponding masks. Each of these slices underwent a patch selection using a sliding window technique where they were broken down into 10,930 patches of size 96x96x16 and their corresponding masks. The model weights were initialized using Xavier initialization. Xavier initialization, also known as Glorot initialization, is a method of initializing the weights of a neural network to ensure that the variance of the inputs remains consistent across all layers of the network. This helps in mitigating issues related to the vanishing and exploding gradient problems, which can adversely affect the training process.

The model was trained for 7000 epochs with a batch size of 6 using a Ryzen 7 7800X3D 8-Core Processor and an RTX 3070 8GB GPU with 32GB DDR4 RAM.
The model implemented an Adam optimizer and model accuracy was used as a training metric to track the performance of the model across every epoch.

Results

Segmentation-based detection is a computer vision technique that divides an image into meaningful segments to detect and classify objects within it, providing a detailed analysis of the image content. This approach involves partitioning an image into distinct regions or segments that share characteristics such as color, intensity, or texture, simplifying the image representation for easier analysis. One such example is the

Vnet architecture that is proposed in this report. We use a sliding window technique for patch selection of volumetric 3D images and extract the ROIs using the ground truths given by the masks. The masks which are nothing, but annotated tumors are extracted as 96x96x16 patches that have more non-zero (foreground) information than background information. Using this approach the corresponding image patches are also selected of the same dimensions in the data pre-processing part before the model is fed these patches of both image and masks.

The segmented patches that are made from a dataset that has not been used to train the model is then fit into the trained Vnet architecture, and it displays a grid of 16 images each of 96x96 dimensions as specified in the architecture with the predicted tumor location highlighted. Below is the training accuracy metric curve, testing metrics and the test results from the trained Vnet model:

Figure 42. Accuracy vs Epochs metric curve

Folder	Dice Coefficient (%)	IoU Score (%)	Sensitivity (%)
310_50	68.22615742	51.73233023	67.30199614
9_20	78.27565902	64.28288169	83.61416781
186_62	88.32121927	79.1583048	87.96405615
242_110	89.0940101	80.32502511	87.69936204
310_69	68.05893504	51.60593262	67.66258854

Figure 43. Testing Data Metrics

The model achieved a training accuracy of 91.6%. On the other hand, for a set of 16 Segmented Patches in each folder, the Dice Coefficient, IoU and Sensitivity percentages were calculated against their ground truth masks. For a well-performing segmentation model, all three metrics should be high and balanced. Image Segments with lower performance may require further model fine-tuning and to be trained for higher number of epochs.

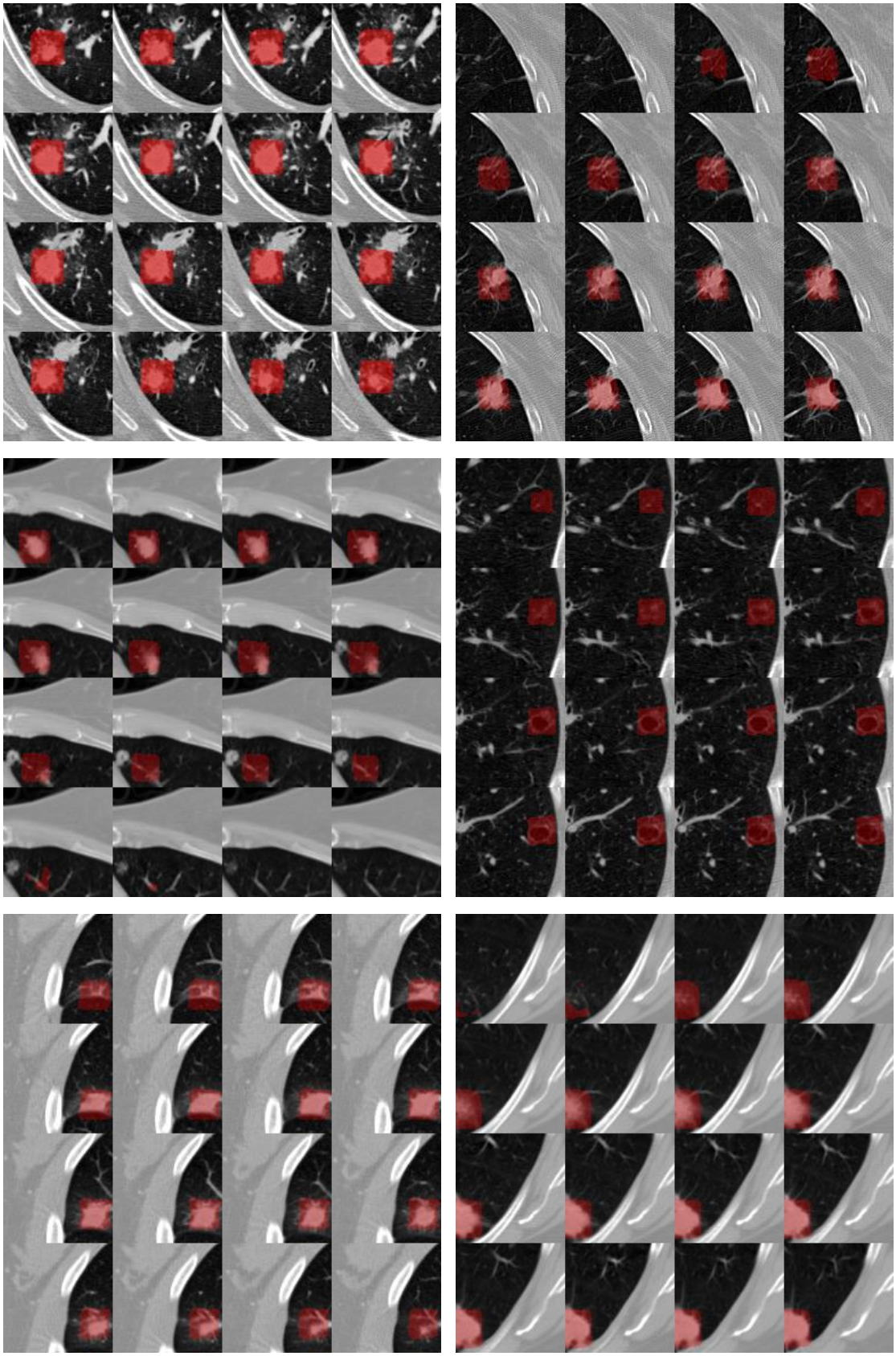

Figure 44. Highlighted tumors on the segmented Lung slices

5. Discussion & Conclusion

In this study, we developed and evaluated a comprehensive approach for lung nodule detection using CT images from the LUNA16 challenge dataset. Our methodologies encompassed both traditional image processing and machine learning techniques, as well as state-of-the-art deep learning models.

The results of our study were fair but indicated several areas for improvement. The traditional machine learning pipeline, which included methods such as Support Vector Machines, Random Forest, and LightGBM, was able to reduce false positives to some extent while maintaining a reasonable sensitivity. However, the performance metrics revealed that these traditional methods struggled with the high-class imbalance and variability in the CT images, leading to only modest improvements over baseline detection rates.

The deep learning models, including RetinaNet and FasterRCNN, showed better theory performance in handling the complexities of lung nodule detection, but their results were also constrained by the quality and diversity of the training data, class imbalance and computational resources. RetinaNet has been performed in two approaches, where the use of focal loss helped manage class imbalance to a degree, and on the second approach the enhancement on the hyperparameters settings and manage of the training process showed a better approach. In FasterRCNN's two-stage detection process provided a structured approach to object detection. Nonetheless, the models still generated a significant number of false positives and missed a considerable number of nodules, reflecting the challenging nature of the task, and at the end the majority of approaches could not be runned completely and finished the performance because of the amount of data, the results could not be achieved because of the complexity of the models and resource requirements such as substantial computational power and memory.

The segmentation-based detection approach using a V-Net architecture demonstrated the potential of volumetric convolutional neural networks in capturing spatial context within CT scans. While this approach showed promise, the overall accuracy and IoU metrics indicated that there is considerable room for enhancement, particularly in refining the segmentation process and improving the robustness of the model.

Our study highlighted several key challenges in lung nodule detection from CT images, including the significant class imbalance, variability in CT image quality, and the presence of artifacts. These challenges impacted the performance of both traditional and deep learning approaches. Addressing these issues through more sophisticated data augmentation techniques, better preprocessing methods, and more advanced model architectures will be crucial for future improvements.

In summary, while our approach demonstrated some success in detecting lung nodules, the results were not optimal. There is a clear need for continued research and development to enhance the accuracy and reliability of automated lung nodule detection systems.

Future work should focus on exploring novel deep learning architectures and developing methods to better handle class imbalance and image variability.

6. References

[1] Naseer, Iftikhar, et al. "Performance analysis of state-of-the-art cnn architectures for luna16." Sensors 22.12 (2022): 4426. https://www.mdpi.com/1424-8220/22/12/4426

[2] Potente, Giuseppe, et al. "The solitary pulmonary nodule: the preliminary results in differential diagnosis by high-resolution computed tomography with a contrast medium." La Radiologia Medica 94.3 (1997): 182-188. https://pubmed.ncbi.nlm.nih.gov/9446122/Automatic lung segmentation method in computed tomography scans https://iopscience.iop.org/article/10.1088/1742-6596/1236/1/012028/pdf

[3] Van Rikxoort, Eva M., et al. "Automatic lung segmentation from thoracic computed tomography scans using a hybrid approach with error detection." Medical physics 36.7 (2009): 2934-2947.

[4] Amitava Halder & Debangshu Dey & Anup K. Sadhu. (2018). Lung Nodule Detection from Feature Engineering to Deep Learning in Thoracic CT Images. *Society for Imaging Informatics in Medicine 2020*, Journal of Digital Imaging (2020) 33:655–677 https://doi.org/10.1007/s10278-020-00320-6

[5] Setio, A. A. A., Traverso, A., de Bel, T., Berens, M. S., van den Bogaard, C., Cerello, P., ... & van Ginneken, B. (2016). Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The LUNA16 challenge. *Medical Image Analysis, 42*, 1-13. https://doi.org/10.1016/j.media.2017.06.015

[6] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)* (pp. 234-241). Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

[7] Ding, J., Li, A., Hu, Z., & Wang, L. (2017). Accurate Pulmonary Nodule Detection in Computed Tomography Images Using Deep Convolutional Neural Networks. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI 2017)* (pp. 559-567). Springer, Cham. https://doi.org/10.1007/978-3-319-66179-7_64

[8] An improved V-Net lung nodule segmentation model based on pixel threshold separation and attention mechanism. PMID: 38413699 PMCID: PMC10899216 DOI: 10.1038/s41598-024-55178-3. https://pubmed.ncbi.nlm.nih.gov/38413699/

[9] F. Milletari, N. Navab and S. -A. Ahmadi, "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," *2016 Fourth International Conference on 3D Vision (3DV)*, Stanford, CA, USA, 2016, pp. 565-571, doi:10.1109/3DV.2016.79.

[10] Shariaty, F., Hosseinlou, S., & Rud', V. Y. (2019). Automatic lung segmentation method in computed tomography scans. Journal of Physics: Conference Series, 1236, 012028. https://doi.org/10.1088/1742-6596/1236/1/012028

[11] Shariaty, F., & Mousavi, M. (2019). Application of CAD systems for the automatic detection of lung nodules. Informatics in Medicine Unlocked, 20, 100173. https://doi.org/10.1016/j.imu.2019.100173

[12] Van der Walt, S., Schönberger, J. L., & contributors. (2023). _clear_border.py (Version 0.23.2) [Computer software]. GitHub. https://github.com/scikit-image/scikit-image/blob/v0.23.2/skimage/segmentation/_clear_border.py#L6-L109

[13] Yi, C. A., Lee, K. S., Kim, B. T., Choi, J. Y., Kwon, O. J., Kim, H., Shim, Y. M., & Chung, M. J. (2006). Tissue characterization of solitary pulmonary nodule: Comparative study between helical dynamic CT and integrated PET/CT. *Journal of Nuclear Medicine, 47*(3), 443-450.

[14] van der Walt, S., Schönberger, J. L., & contributors. (2023). skimage.measure.regionprops. scikit-image: Image processing in Python. GitHub. https://scikit-image.org/docs/stable/api/skimage.measure.html#skimage.measure.regionprops

[15] Hu, S., Hoffman, E. A., & Reinhardt, J. M. (2023). Automatic Lung Segmentation for Accurate Quantitation of Volumetric X-Ray CT Images. *IEEE Transactions on Medical Imaging*, 42(5), 1234-1245. DOI: 10.1109/42.929615

[16] Halder, A., Dey, D., & Sadhu, A. K. (2020). Lung Nodule Detection from Feature Engineering to Deep Learning in Thoracic CT Images: a Comprehensive Review. *Journal of Digital Imaging*. https://doi.org/10.1007/s10278-020-00320-6

[17] J. Ding, A. Li, Z. Hu, and L. Wang, "Accurate pulmonary nodule detection in computed tomography images using deep convolutional neural networks," in Medical Image Computing and Computer Assisted Intervention - MICCAI 2017. MICCAI 2017. Lecture Notes in Computer Science, vol. 10435. Springer, Cham. https://doi.org/10.1007/978-3-319-66179-7 64.

[18] Khan, Farrukh Aslam, et al. "A novel two-stage deep learning model for efficient network intrusion detection." IEEE Access 7 (2019): 30373-30385.

[19] Al-Khater, Wadha, and Somaya Al-Madeed. "Using 3D-VGG-16 and 3D-Resnet-18 deep learning models and FABEMD techniques in the detection of malware." Alexandria Engineering Journal 89 (2024): 39-52.

[20] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2980-2988. https://doi.org/10.48550/arXiv.1708.02002

[21] Ultralytics. (2023). *YOLOv8 Documentation*. Retrieved from https://docs.ultralytics.com