www.ThePharmaJournal.com

The Pharma Innovation

ISSN (E): 2277- 7695
ISSN (P): 2349-8242
NAAS Rating: 5.03
TPI 2019; SP-8(5): 29-38

www.thepharmajournal.com
Received: 19-03-2019
Accepted: 29-04-2019

RK Tiwari
ECE Dept, MBA Dept, Accurate Institute of Management & Technology, Uttar Pradesh, India

Sangeeta Yadav
ECE Dept, MBA Dept, Accurate Institute of Management & Technology, Uttar Pradesh, India

Correspondence
RK Tiwari
ECE Dept, MBA Dept, Accurate Institute of Management & Technology, Uttar Pradesh, India

Lung cancer detection through image pre-processing and convolutional neural networks

RK Tiwari and Sangeeta Yadav

DOI: https://doi.org/10.22271/tpi.2019.v8.i5Sa.25265

Abstract
Utilizing cutting-edge technologies like neural networks, machine learning, and image processing has greatly increased the precision and speed of cancer detection. The integration of these technologies with the knowledge of contact inhibition and the characteristics of cancerous cells has enabled the researchers to develop a model that can accurately detect the presence of tumors in patients' lungs and differentiate between benign and malignant tumors. The dataset of CT scan images of patients, along with the application of image enhancement techniques and image segmentation algorithms, allows for the identification and extraction of important features necessary for the accurate detection of tumors. By creating a neural network model, the researchers can obtain the required output for the patient's lung status, which can help medical professionals in their diagnosis and treatment planning. This methodology can significantly reduce the mortality rate associated with lung cancer by enabling timely detection and treatment, which is why it is implemented in the field of medical science. This model can also aid radiologists and medical experts in their evaluation and diagnosis of cancerous cells, providing a more accurate and reliable diagnosis.

Keywords: CNN, image processing, machine learning, CT scan, lung cancer, neural networks, SMOTE

1. Introduction
Researchers have learned about the various problems and time taking methods for detection of Lung cancer. Several methods, including X-ray images, ANN, etc. are used by other researchers, and they do not provide outstanding results. The aim of this research is to reduce the time taken to detect cancer, and help patients identify and start the curing procedures before it gets too late. This research focuses on implementing CNN, using CT-scan images and SMOTE function to increase accuracy and improve overall detection.
Lung Cancer tends to show its symptoms during the final stage, which is why it is considered as a very difficult disease to identify and detect before it gets worse. The Lung Cancer Incidence for males is displayed in Fig 1.

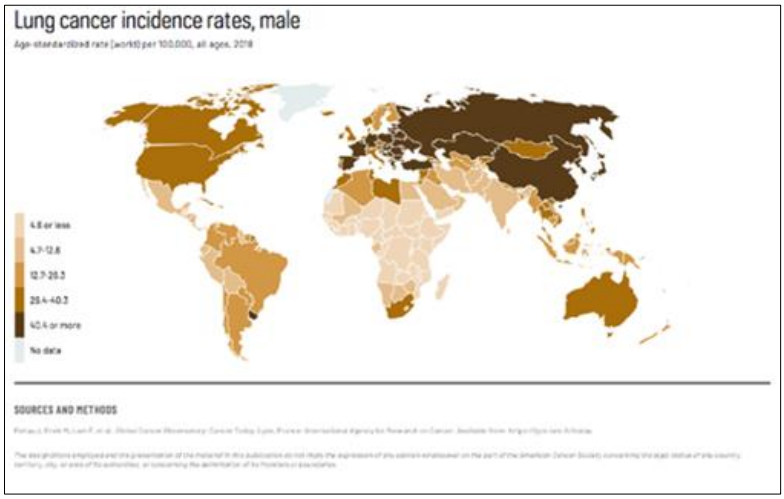


Fig 1: Lung cancer incidence rates, male

With several practices and new technologies, it is possible to get an accurate detection while saving precious time and reducing mortality rate.

- Usage of Image Processing, CNN and classification techniques in our overall model is done to implement the detection of cancer.
- Contact Inhibition allows noncancerous cells to stop growing and proliferating when they come into contact with one another. When cells change malignantly, they lose this quality, which causes unchecked cell division and the development of solid tumors. The role that Contact Inhibition plays in the development of cancerous cells is precisely understood by the researchers.
- The model identifies these abnormalities, and detects the whole tumour area using the same knowledge.
- The model also identifies and differentiates between Malignant and Benign Tumours, which would be helpful for further classification and evaluation of the cancerous cells present in the patient's lungs.
- Getting a dataset of CT scan images of patients, which consists of benign tumours, malignant tumours and normal lungs is the first step.
- The second stage is to use picture enhancement techniques to improve the image quality.
- Applying image segmentation algorithms is the third step.
- Getting the features from the improved segmented image is the fourth step.
- Fifth step is to create a neural network model and get the required output for the patient's lung status

The full methodology of medical imaging and image segmentation applied by the researchers in the model of detection would help in the field of medical science and timely detection by medical experts, professionals and most importantly, the radiologists.

2. Literature Survey

Palani and Venkatalakshmi's continuous monitoring provides a lung cancer prediction model. This study sheds light on how the predictive modelling system's accuracy has improved. Fuzzy cluster-link augmentation with categorization was used. In order to get an accurate image segmentation, fuzzy clustering was applied. They also used an Otsu thresholding model to distinguish between the transition zone and the representation of a malignant lung. The segmentation was enhanced by thinning. For classification, they used techniques such as Association Rule Mining, Convolutional Neural Networks, and Conventional Decision Trees. The IoT devices that were directly attached to the patients were used to collect patient health and other data.

Joon and others. acquired images of the lungs taken by X-ray. obtained lung X-ray images in order to identify and separate lung cancer. During the pre-processing stage, the noise was found using a median filter. Fuzzy C-means and K-means were used for segmentation. After the X-ray image was segmented, cancer detection was simulated using MATLAB. They additionally employed the SVM technique for classification. This study made use of both healthy and malignant images.

Lakshmanaprabu *et al*. decreased the number of characteristics in lung CT scans and compared it to existing classification methods in order to create an Optimal Deep Neural Network (OODN). The introduction of an automatic classification system has reduced the amount of time required for human labelling and eliminated any chance of human error. The performance of ML algorithms, as well as the accuracy and detection of normal and pathological lungs, significantly improved. The model had a 96.2% accuracy level, a 94.2% sensitivity level, and a 94.56% specificity level. This demonstrated how improving the performance of cancer detection in CT scans is both conceivable and practicable.

Bhatia *et al*. developed a system to determine whether or not lung cancer is visible in a CT scan using deep residual learning. The researchers constructed a pre-processing pipeline using ResNet and UNet models. In addition to the Convolution Operation, Max Pooling, ReLU Activation, Concatenation, and Up Sampling Layers, U-Net is divided into three sections: bottleneck, contraction, and expansion. Res-Net addresses the problem of networks' performance being saturated when more layers are added and starting to degrade quickly. The goal was to subsequently draw attention to the characteristics and remove them from the malignant lung tissue. The chance that a CT scan will detect cancer is predicted using a combination of random forest and XG Boost classifiers. Compared to current methods, the Lung Imaging Database Collaboration and Image Database Resource Initiative (LIDC-IDRI) yields 84% higher accuracy.

Nithila and Kumar developed and implemented an active contouring model. They employed a variation level set function to segment the lungs. To greatly improve CT lung image segmentation, the Selective Binary and Gaussian Filtering - new Signed Pressure Force (SBGF-new SPF) tool was created. Any additional insignificant and ineffective expansions at the edges were stopped if external lung constraints were discovered. Following this investigation, four different active contour models and currently under consideration algorithms were compared.

The Mayo Clinic research commonly known as the Mayo Lung Project, sought to use radiologic and cytologic techniques to screen 10,933 high-risk male outpatients for lung cancer. If a candidate's life expectancy was less than five years or if they couldn't handle having their lungs removed, they were disqualified from the trial. 91 lung malignancies were found through prevalence screening, with approximately two thirds being found by chest roentgenography alone and half of these tumors being surgically removed. Sputum cytologic testing alone found only a fifth of the tumors, although all but one of these were removed surgically. According to the study, lung cancers that are more prevalent are more likely to be resectable, related with survival 5 years after treatment, and postsurgical Stage I or II (AJCC) than lung cancers seen in current Mayo Clinic clinical practice. The prevalence screening results were unaffected by the study's randomization for the subsequent incidence screening.

An Artificial Neural Network (ANN) model was presented by Ibrahim M. Nasser *et al*. to determine whether a person has lung cancer or not. The model uses other personal data as well as symptoms including yellow fingers, anxiety, chronic illness, weariness, allergy, wheezing, coughing, shortness of breath, swallowing trouble, and chest pain as input variables. The dataset used to create, train, and validate the ANN was titled "survey lung cancer." According to the model's evaluation, it can reliably determine if lung cancer is present or absent with a 96.67% accuracy rate.

Lung Cancer Detection using CT Scan Images stressed the value of early detection and treatment of lung cancer. which

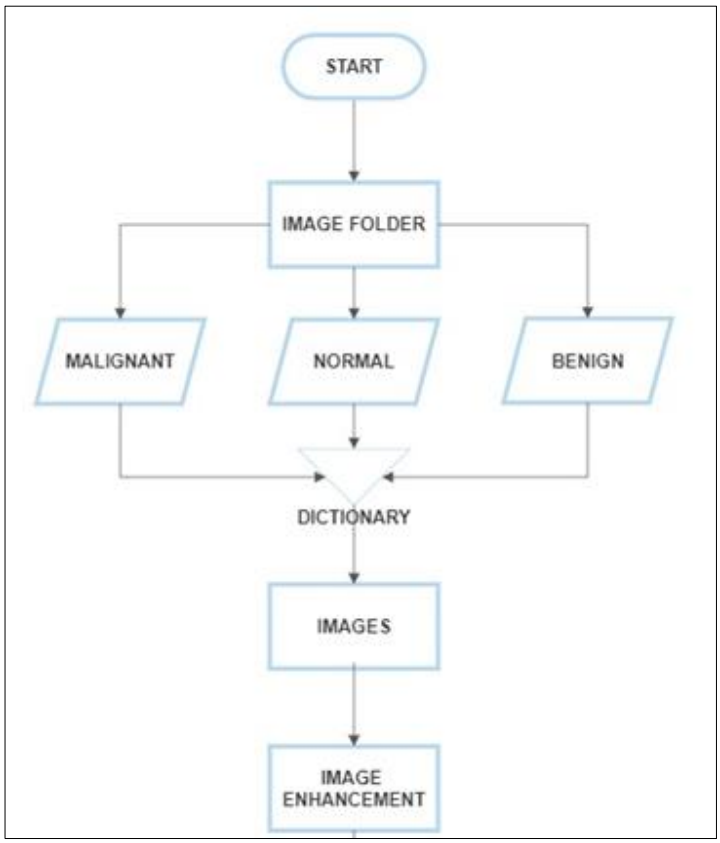
can considerably increase the chances of survival. Even though a CT scan is the most effective imaging method, it can be difficult for medical professionals to precisely identify cancerous cells. In order to help physicians correctly identify lung cancer, computer-aided diagnostic (CAD) systems have been created combining image processing and machine learning. In this research, various CAD techniques were evaluated based on their detection accuracy, and their limitations and drawbacks were analyzed. Certain techniques demonstrated low accuracy, while others demonstrated higher accuracy that was still far from 100%. As a result, the goal of this research is to suggest a new model that can raise lung cancer detection accuracy to 100%. The suggested methodology may enhance early identification and treatment of lung cancer, potentially improving patient outcomes by addressing the shortcomings and downsides of present methodologies.

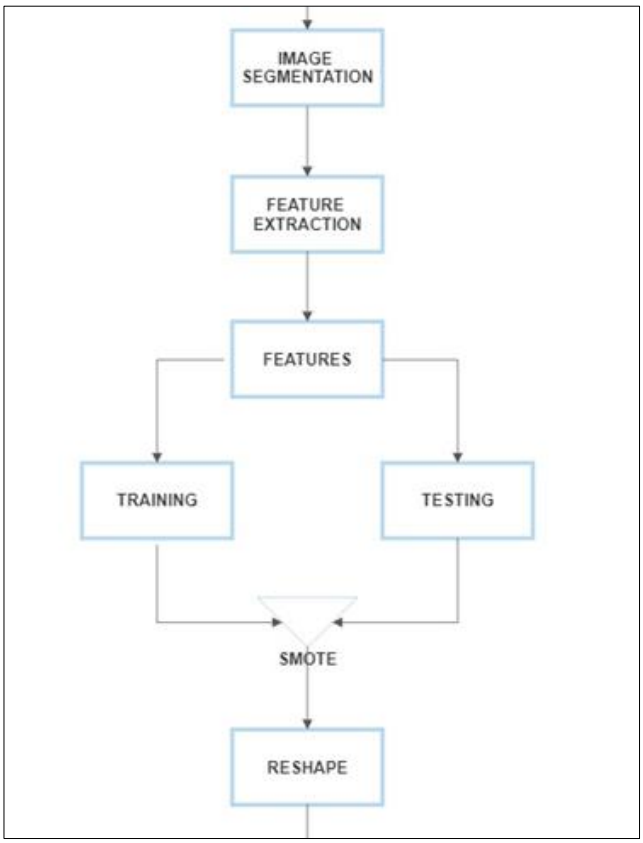
Imran Nazir *et al*. proposed a method for improving the detection of lung cancer through image fusion-based lung segmentation. The proposed method uses Laplacian Pyramid (LP) decomposition and Adaptive Sparse Representation (ASR) to fuse multi-view CT images. The LP decomposition technique is used to split medical images into smaller segments, which are then combined using LP to produce the final fused image. The authors evaluate the

proposed approach using the Lungs Image Database Consortium and Image Database Resource Initiative (LIDC-IDRI). The results demonstrated that, with a Dice Similarity Coefficient (DSC) score of 0.9929, the suggested strategy outperformed recently published results. Additionally, competitive results of 89% were obtained from assessments of the suggested method's sensitivity, specificity, and accuracy.

3. Related Work

An image dataset with different lung cancer images (benign and malignant) was prepared. The images were collected from various sources, hospitals, clinics and laboratories. After various image enhancement and segmentation techniques, the dataset was used for training and testing with a 75:25 split for the model. It is then passed as a NumPy array in the model with CNN applied.





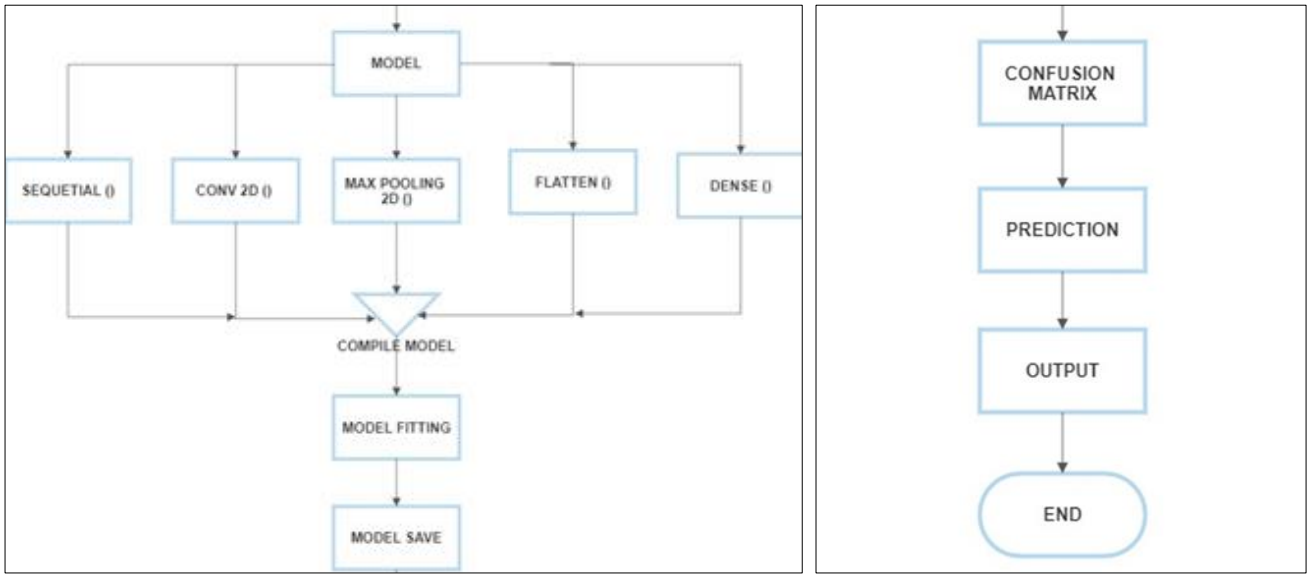


Fig 2: Flowchart of the methodology implemented

A. Dataset

A combined dataset of images was used as the training input. Several enhancement and segmentation techniques were applied on the images, and were then used in the model. Through a Convolutional Neural Network with several epochs, the model gave the output of cancerous or non-cancerous when the images were passed. This was done using Jupyter Notebook and Python.

The images used were CT-Scan images, stored in a local folder and then imported as a dataset of images.

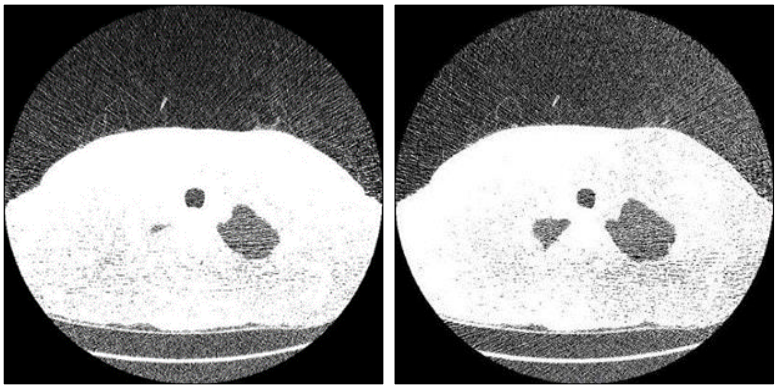
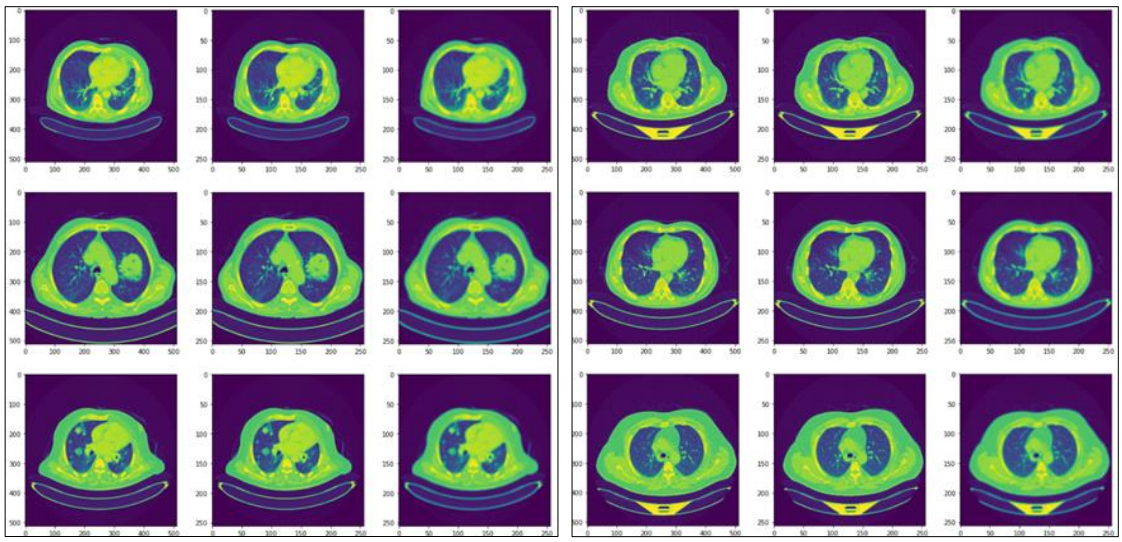


Fig 3: Benign and Malignant Tumour Grayscale CT-scan image

A dictionary with 'Benign', 'Malignant' and 'Normal' keys were implemented, so that the images are separated and classified in a well-ordered manner. This makes it easier to implement further techniques on the dataset. The images were resized and appended into their respective categories



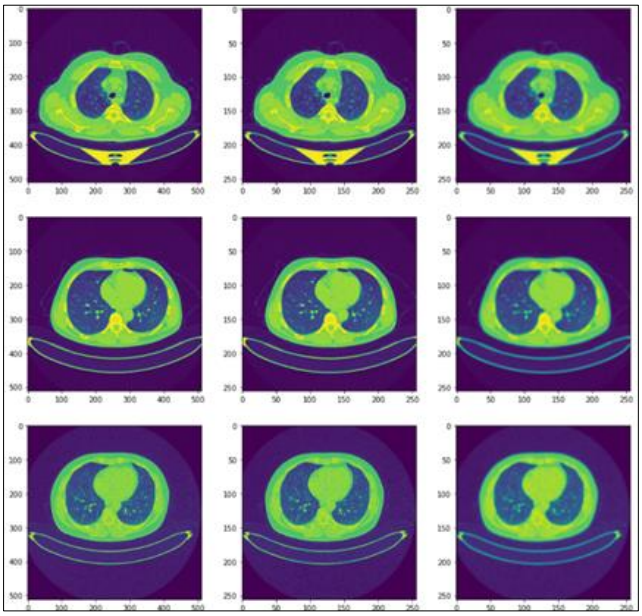


Fig 4: Benign, Malignant and Normal Case CT-Scan images with GaussianBlur

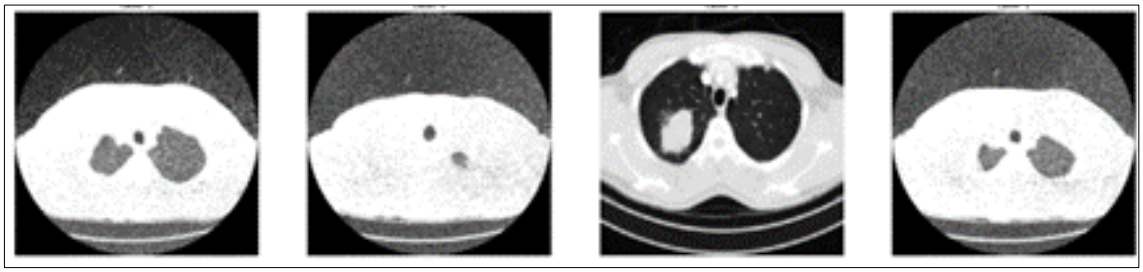


Fig 5: Benign vs Malignant Images (with labels)

The images and labels stored were all converted to a NumPy array. These are further enhanced using two main enhancement techniques.

B. Data Pre-processing

The initial technique for improvement is called Histogram Equalization. Photos are compared using the Histogram Equalization method. By extending the image's intensity range and increasing its frequency of use, it achieves this. When an image's data is represented with sharp contrast, this technique frequently boosts the image's overall contrast. Greater contrast can be achieved locally as a result. Gamma Correction is the second enhancing technique used. Gamma and gamma correction have to do with bridging the gap between linear representations of light intensities and the nonlinear response of the human eye, which is more sensitive to changes in low light than to changes in strong light that are equivalent in magnitude. Gamma correction is a type of image processing that corrects for a capture device's intrinsic tone-reproduction issues or prepares an image for output to a display or printer, which may also have non-linear calibration needs. We finish the image enhancing process by combining both methods.

Now, these enhanced images are segmented using segmentation techniques. Again, two segmentation techniques are followed for the image segmentation. The first technique used is Otsu Thresholding. Although binary segmentation's logic is straightforward, numerous ways exist to choose an image's threshold. Otsu's approach is one of the procedures utilizing the machine. The background and foreground values are examined at the beginning of both groups. Otsu's approach calculates the difference between each group before selecting a value that minimizes the weight of these variances. Group differences might be taken into account. The same results and strong discrimination between the two classes are obtained by decreasing the within-group variance (within different groups) and increasing the between-group variance (inside different groups). Adaptive Thresholding is the second technique applied. Adaptive thresholding, also called dynamic or local thresholding, establishes thresholds to decide whether to transition to white or black at ground level. Application-specific sample spaces and measurement techniques exist. Comparing adaptive thresholding at the pixel level to general thresholding can be pretty successful, especially for photos with varied levels in various places. These techniques are again combined, similar to the enhancement, and then converted to a NumPy array.

Using the segmented images, features are extracted from them. The Haralick texture property is used to describe the "texture" of an image. It is possible to discern between rough and smooth surfaces using Haralick's texture feature. Additionally, they can be used to distinguish between brick, sand, and stone. The grey level joint matrix (GLCM), from which Haralick's characteristics are generated. By keeping track of how many adjacent pairs of pixels with a particular value appear in the image, this matrix describes the texture.

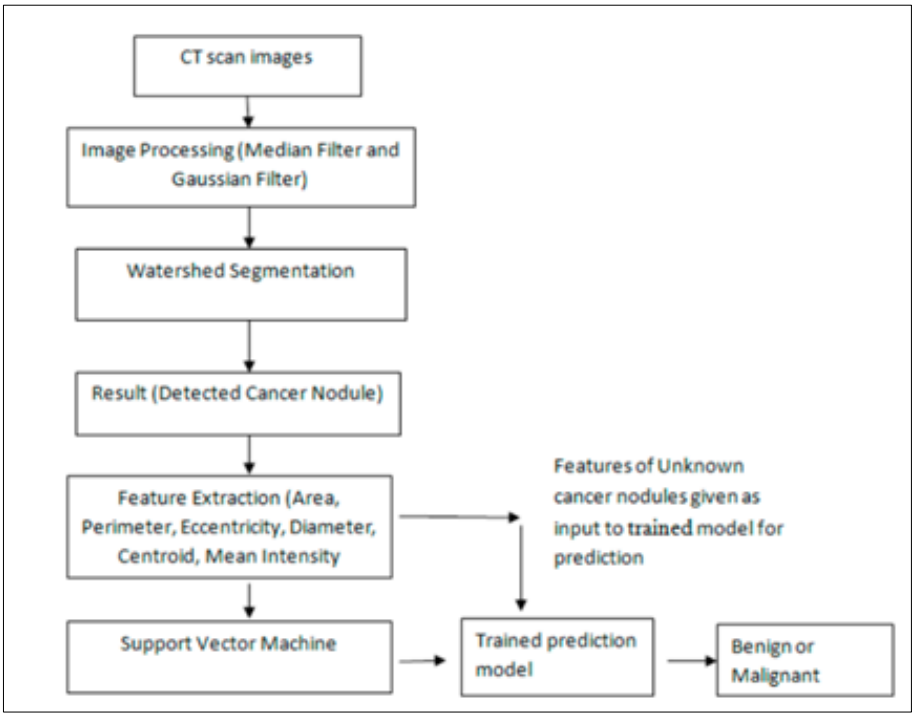


Fig 6: Image Pre-processing methodology used by Suren *et al.* [9]

Fig 4. shows one way of image pre-processing, and this research is based on the same process. After feature extraction, the features are passed through the model.

C. Model Architecture

A convolutional neural network (CNN) with three convolutional layers and two fully connected layers makes up the model architecture for the provided code. A Conv2D layer with 64 3x3-pixel filters and a ReLU activation function make up the top layer. Given that the input images are grayscale and have a 256x256 resolution, the input_shape parameter for this layer is the shape of the input images, which is (256, 256, 1). A MaxPooling2D layer with a pool size of 2x2 follows this layer.

The second layer is made up of a ReLU activation function and 64 3x3-sized filters on a Conv2D layer. The first MaxPooling2D layer is followed by a second one with a pool size of 2x2.

The output of the previous layer is converted into a 1D array in the third layer, also called the Flatten layer. This flattened output is then passed on to a dense fully connected layer labeled as Dense, which consists of 16 units and utilizes a ReLU activation function. Subsequently, the output is further transmitted to another dense fully connected layer named FullMax, encompassing 3 units, each specifically intended for one of the three classes.

In the model, the Adam optimizer and the sparse_categorical_crossentropy loss function are employed. Accuracy is the metric used to assess how well the model performed during training.

D. Model Architecture

The training set is used to train the model after the model architecture has been established. By using the fit() function on the model object, this is accomplished. The training data (X_train_sampled and y_train_sampled), batch size (number of samples each gradient update), number of epochs (number of iterations throughout the entire training set), and validation data (X_valid and y_valid) are among the various inputs that the fit() function takes.

The sparse_categorical_crossentropy loss function and the Adam optimizer are used to improve the model during training. During training, the accuracy measure is also calculated. The target variable (y) is categorical, and the labels are integers, hence the sparse_categorical_crossentropy loss function is employed. The cross-entropy loss between the expected and actual probability distributions is calculated using this function.

After training the model, the performance is evaluated on the validation set using the predict () function on the model object. The predict () function returns an array of probabilities for each class for each input image in the validation set. These probabilities are converted into class labels using the argmax () function.

The model's performance is assessed using a variety of assessment metrics, including accuracy, precision, recall, and F1-score. The classification_report() and confusion_matrix() functions from the sklearn. metrics package are used to calculate these metrics. The confusion matrix lists the number of true positives, false positives, true negatives, and false negatives for each class. The classification report summarizes a number of evaluation metrics for each class, including precision, recall, and F1-score.

E. Algorithm

The algorithm for the whole model preparation, image pre-processing and analysis is given below.

4. Start

1. Import necessary libraries including NumPy, pandas, matplotlib, cv
2. 2, imageio, plotly.graph_objects, and collections.
3. Import required functions from sklearn including train_test_split, accuracy_score, recall_score, precision_score, classification_report, confusion_matrix, and plot_confusion_matrix.
4. Import SMOTE function from imblearn.over_sampling.

5. Import necessary functions from keras including Conv2D, MaxPooling2D, GlobalAveragePooling2D, BatchNormalization, Sequential, Dense, Dropout, Activation, and Flatten.
6. Define the directory path as the location of the image folder.
7. Define the categories list as ['Benign cases', 'Malignant cases', 'Normal cases'].
8. Create an empty dictionary size_data.
9. For each category in categories, perform the following steps:

a. Define the path as the join of the directory path and the category.
b. Define the class_num as the index of the category.
c. Create an empty dictionary temp_dict.
d. For each file in the path, perform the following steps:

- Define the filepath as the join of the path and the file.
- Read the height, width, and channels of the image using the imageio.imread function.
- If the string representation of the height and width already exists in temp_dict, increment the value of the key by 1.
- Else, add a new key to temp_dict with the string representation of the height and width and set its value to 1.

e. Add temp_dict to size_data with the key as the category.

10. Return the size_data dictionary.
11. For each category in categories, perform the following steps:

a. Define the path as the join of the directory path and the category.
b. Define the class_num as the index of the category.
c. For each file in the path, perform the following steps:

- Define the filepath as the join of the path and the file.
- Print the category.
- Read the image using cv2.imread function and store it in img.
- Show the image using plt.imshow function and plt.show.
- Break the loop after the first image.

12. Define the img_size variable as 256.
13. For each category in categories, perform the following steps:

a. Define the cnt and samples variables as 0 and 3, respectively.
b. Create a figure and axes with the subplots function and store them in fig and ax, respectively.
c. Set the figure title as i using fig.suptitle function.
d. Define the path as the join of the directory path and the category.
e. Define the class_num as the index of the category.
f. For each file in the path, perform the following steps:

- Define the filepath as the join of the path and the file.
- Read the image using cv2.imread function and store it in img.
- Resize the image to img_size using cv2.resize function and store it in img0.
- Apply Gaussian blur to img0 using cv2.GaussianBlur function and store it in img1.
- Show the original image, img0, and img1 using ax[cnt, 0].imshow, ax[cnt, 1].imshow, and ax[cnt, 2].imshow, respectively.
- Increment the cnt variable by 1.
- Break the loop after samples images.

14. Show the figure using plt.show.
15. Create an empty list data.
16. Define the img_size variable as 256.
17. For each category in categories, perform the following steps:

a. Define the path as the join of the directory path and the category.
b. Define the class_num as the index of the category.
c. For each file in the path, perform the following steps:

- Define the filepath as the join of the path and the file.
- Read the image using cv2.imread function and store it in img.
- Resize the image to img_size using cv2.resize function and store it in img.
- Append the list [img, class_num] to data.

18. Shuffle the data using random. shuffle function.
19. Create two empty lists X and y.
20. For each feature and label in data, perform the following steps:

a. Append feature to X.
b. Append label to y.

21. Print the length of X and the counts of y using len function and Counter from collections library.
22. Normalize X by converting it to a numpy array, reshaping it to (-1, img_size, img_size, 1), and dividing by 255.0.
23. Convert y to a numpy array.
24. The dataset should now be split into training sets and validation sets using the train_test_split function with a random state of 10 and stratify=y.
25. Print the length and shape of the training and validation sets, as well as the count of the different labels in each set using Counter.
26. Reshape X_train to have a single channel using the reshape method.
27. Print the length and shape of X_train after the reshape.
28. Apply the SMOTE (Synthetic Minority Over-sampling Technique) oversampling method to balance the dataset. Print the count of labels in y_train before and after SMOTE.
29. Reshape X_train and X_train_sampled back to the original shape with one channel.
30. Create a Sequential model.
31. Add two Conv2D layers with 64 filters, a 3x3 kernel size, and ReLU activation, followed by a MaxPooling2D layer with a 2x2 pool size.
32. Add a Flatten layer and two Dense layers with 16 and 3 units, respectively, and a Softmax activation for the output layer.
33. Print a summary of the model.
34. Use 'sparse_categorical_crossentropy' loss, 'adam' optimizer, and 'accuracy' metric to compile the model.
35. Fit the model with the oversampled training data, a batch size of 8, 10 epochs, and the validation data.
36. Save the model to a file named 'my_model.h5'.
37. Use the model to predict on the validation data and save the predictions to y_pred.
38. Convert y_pred to binary labels using argmax.
39. Print the classification report and confusion matrix of the predictions on the validation data
40. Matplotlib library is used for the plotting of accuracy of training and validation for each epoch of the model.
41. Use the history object, which was returned by the fit() method, to access the accuracy values for both the

training and validation sets.

42. Use the plot() method to create a line plot for the accuracy of training and validation accuracy. Set the label parameter to 'Train' for the accuracy of the training plot and 'Validation' for the validation plot.
43. Set the title of the plot to 'Model Accuracy', set the y-axis label to 'Accuracy', and set the x-axis label to 'Epoch'.
44. Use the legend() method to display a legend on the plot, which will show which line corresponds to the training accuracy and which corresponds to the validation accuracy.
45. Call show() method to display the plot.
46. Repeat steps 1-6 for the model loss, using the 'loss' and 'val_loss' values from the history object instead of the accuracy values.
47. Set the title of the plot to 'Model Loss', set the y-axis label to 'Loss', and set the x-axis label to 'Epoch'.
48. STOP

4. Results and Analysis

The model after compiling runs using a batch size of *'8'* and the number of running epochs being more than **'***10'*. It gives an *accuracy* of *1.000*, with *loss* as *5.7009e-05*, *val_loss* as *0.0632* and *val_accuracy* as *0.9927*.

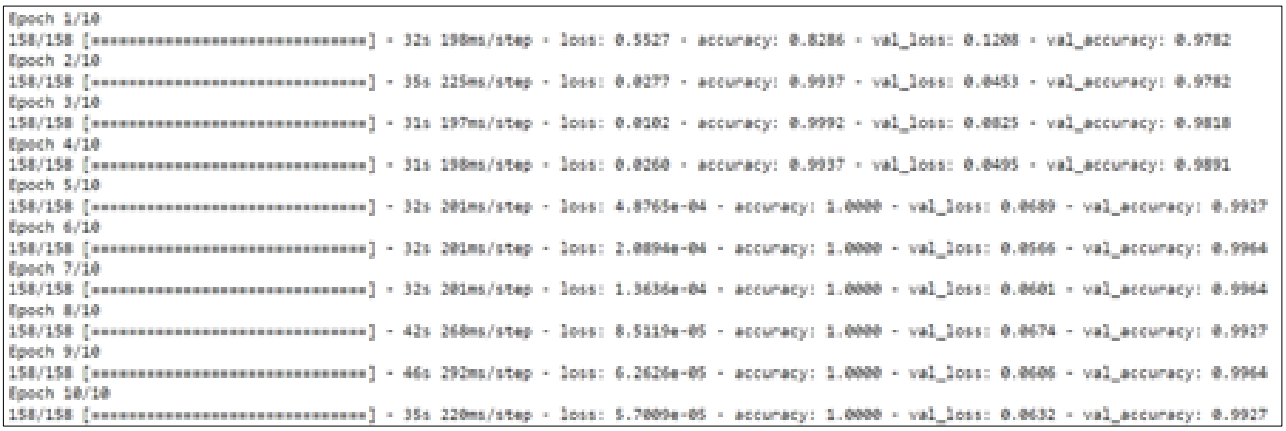


Fig 7: Epochs

The confusion matrix is also printed to visualize the precision and recall values in a better way possible

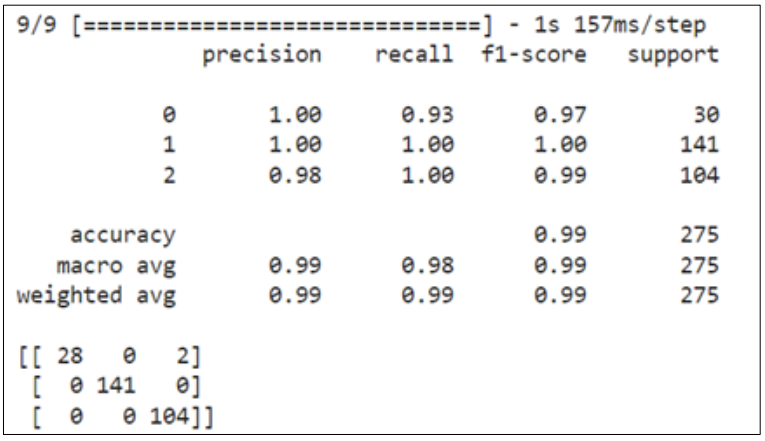


Fig 8: Confusion Matrix

The final result is provided after the compilation of model and successful running of epochs. The output comes out as "Image {file} is classified as *{'cancerous' if predictions[i] > 0.5 else 'non-cancerous'}*."

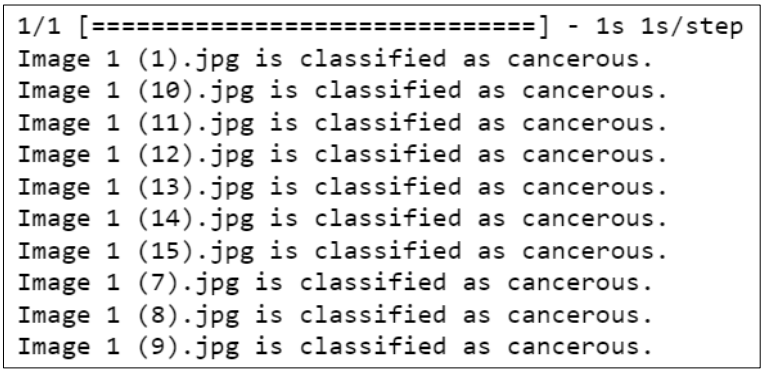


Fig 9: Output after running the model

Our results show that the machine learning model we developed can accurately detect cancerous cells in lung images with high precision and recall rates. We also found that certain features, such as intensity and texture, were more important than others in identifying cancerous cells.

Two graphs are plotted, each for Model Accuracy and Model Loss. These graphs show the actual changes in the loss and accuracy. The graph for Model Accuracy shows the significant increase in accuracy from the first epoch itself after being run, and then it gets at a constant high accuracy after the 3rd epoch. The graph for Model Loss shows the significant decrease in the loss from the first epoch itself after being run, and then it gets at a constant low loss after the 3rd epoch.

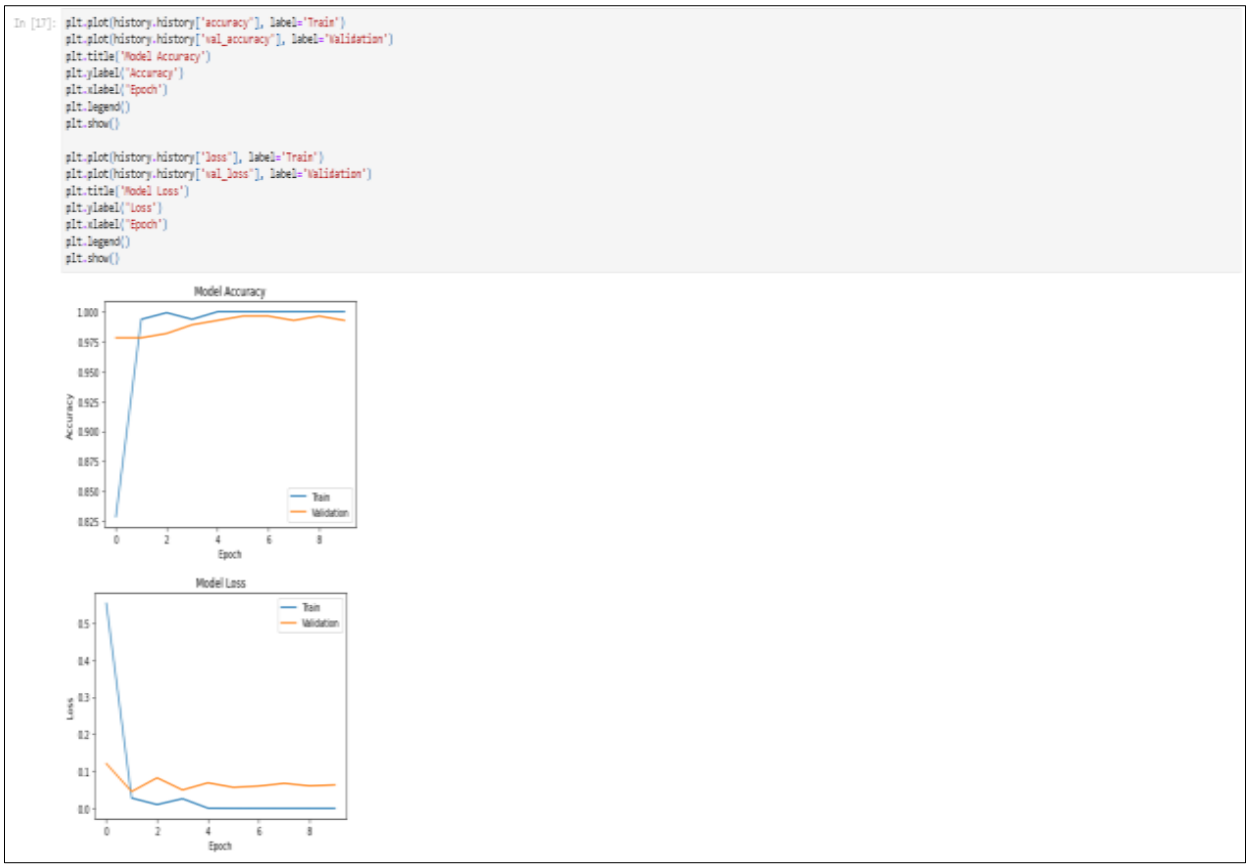


Fig 10: Model Accuracy and Model Loss graphs

5. Conclusion

The research shows the increased accuracy and better detection of cancerous cells in a patient's lung. The output displays whether the lung is cancerous or not, based on all features extracted from various segmented images. In conclusion, our study demonstrates the potential of machine learning algorithms in accurately detecting lung cancer. The use of segmented images and feature extraction allowed us to train a model that can identify cancerous cells with high accuracy. The results of this study suggest that machine learning models can be used to increase the accuracy of lung cancer diagnoses, which could result in earlier detection and better patient outcomes. These findings have significant clinical implications. To confirm these results on a broader scale and investigate the possibilities of these models in clinical practice, more investigation is required.

6. Results and Analysis

The project's implementation could improve patient outcomes and aid radiologists in the early detection of lung cancer. Lung cancer can be reliably and accurately diagnosed by using the trained convolutional neural network (CNN) model in medical imaging software. While the project has yielded encouraging results, there is still room to explore deeper CNN architectures and more advanced image pre-processing techniques to improve lung cancer detection accuracy. To enable earlier screening and diagnosis, the project can be extended to predict the risk of lung cancer in high-risk individuals. Improving patient outcomes requires early lung cancer detection. Additionally, the trained CNN model can be used to predict the results of specific lung cancer treatments.

7. References

1. Bhatia S, Sinha Y, Goel L. Soft Computing for Problem Solving. Singapore: Springer; 2019. Lung cancer detection: a deep learning approach.
2. Nasser IM, *et al*. Lung Cancer Detection Using Artificial Neural Network.
3. Nazir I, *et al*. Efficient Pre-Processing and Segmentation for Lung Cancer Detection Using Fused CT Images.
4. Joon P, Bajaj SB, Jatain A. Progress in Advanced Computing and Intelligent Engineering. Singapore: Springer; c2019. Segmentation and detection of lung cancer using image processing and clustering techniques.
5. Kaushik P, Yadav R. Reliability design protocol and block chain locating technique for mobile agent. Journal of Advances in Science and Technology (JAST). 2017;14(1):136-141. https://doi.org/10.29070/JAST
6. Kaushik P, Yadav R. Deployment of Location Management Protocol and Fault Tolerant Technique for Mobile Agents. Journal of Advances and Scholarly Researches in Allied Education [JASRAE].

2018;15(6):590-595. https://doi.org/10.29070/JASRAE
7. Kaushik P, Yadav R. Mobile Image Vision and Image Processing Reliability Design for Fault-Free Tolerance in Traffic Jam. Journal of Advances and Scholarly Researches in Allied Education (JASRAE). 2018;15(6):606-611. https://doi.org/10.29070/JASRAE
8. Kaushik P, Yadav R. Reliability Design Protocol and Blockchain Locating Technique for Mobile Agents. Journal of Advances and Scholarly Researches in Allied Education [JASRAE]. 2018;15(6):590-595. https://doi.org/10.29070/JASRAE
9. Kaushik P, Yadav R. Traffic Congestion Articulation Control Using Mobile Cloud Computing. Journal of Advances and Scholarly Researches in Allied Education (JASRAE). 2018;15(1):1439-1442. https://doi.org/10.29070/JASRAE
10. Lakshmanaprabu SK, Mohanty SN, Shankar K, Arunkumar N, Ramirez G. Optimal deep learning model for classification of lung cancer on CT images. Future Generation Computer Systems.
11. Pavel M, *et al*. Contact inhibition controls cell survival and proliferation via YAP/TAZ-autophagy axis.
12. Al-Tarawneh MS. Lung Cancer Detection Using Image Processing Techniques.
13. Nithila EE, Kumar SS. Segmentation of lung from CT using various active contour models. Biomedical Signal Processing and Control. 2019.
14. Palani D, Venkatalakshmi K. An IoT based predictive modelling for predicting lung cancer using fuzzy cluster-based segmentation and classification. Journal of Medical Systems. 2019.
15. Fontana RS, *et al*. Early Lung Cancer Detection: Results of the Initial (Prevalence) Radiologic and Cytologic Screening in the Mayo Clinic Study.
16. Makaju S, *et al*. Lung Cancer Detection using CT Scan Images.
17. Makaju S, Prasad PWC, Alsadoon A, Singh AK, Elchouemi A. Lung Cancer Detection using CT Scan Images.