LUNG CANCER DETECTION AND CLASSIFICATION USING IMAGE PROCESSING AND CNN-DEEP LEARNING ARCHITECTURE

Dhruv K.C, Nitin Kumar P, Rajath G, Rhitesh Kumar Singh
Project Guide: Prof. Vanamala H R

ABSTRACT

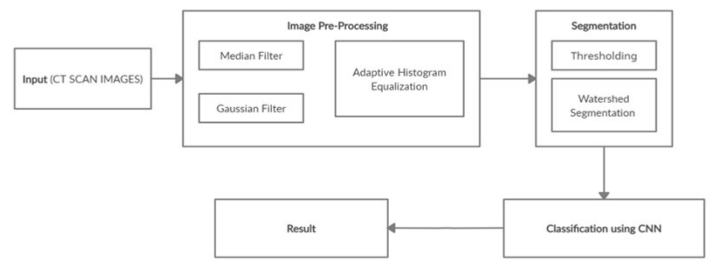
- A large number of cancer deaths in the world is due to lung cancer, which is caused due to unbalanced cell growth. Through our work, a CNN-DEEP learning model is proposed with the help of image pre-processing techniques for detecting and classifying lung cancer in a dataset we aim to help in the diagnosis of the patient's cases: benign, or malignant.
- Here we use the Kaggle Lung CT Challenge dataset and employ different image pre-processing techniques like Median filter to remove salt and pepper noise, Gaussian filter to remove high frequency noises and Adaptive Histogram Equalization to increase the contrast of the image.
- We utilize our 2D CNN model for lung cancer detection and classification and acquire a superior evaluation of the model. We divide our pre-processed dataset into 80%, 10% and 10% for training, validation and testing respectively, and obtain testing accuracy of 99.28% and precision of 1, recall of 0.99, and F1-score of 1.

MOTIVATION

- Currently, there are many techniques to detect and diagnose lung cancer. Most of the methods used by doctors are a long procedural task and might take some time to diagnose lung cancer.
- Early detection of lung cancer is necessary and beneficial to the patient.
- Regular chest x-rays have been studied for lung cancer screening, but they did not help most people live longer.
- In recent years, a test known as a low-dose CAT scan or CT scan (LDCT) has been studied in people at a higher risk of getting lung cancer. LDCT scans can help find abnormal areas in the lungs that may be cancer.
- CT scanning also seems to be less harmful than regular chest x-ray scans.
- A neural network model can be used for the detection of cancer nodules in these CT scan images.
- A detection model with good accuracy can be used for the early detection of lung cancer as it is the key to increasing survival rates for patients with lung cancer.
- Catching cancer early often allows for a higher likelihood of successful treatment. Some early cancers may have signs and symptoms that can be noticed, but that is not always the case.
- After a cancer diagnosis, staging provides important information about the extent of cancer in the body and anticipated response to treatment.

METHODOLOGY

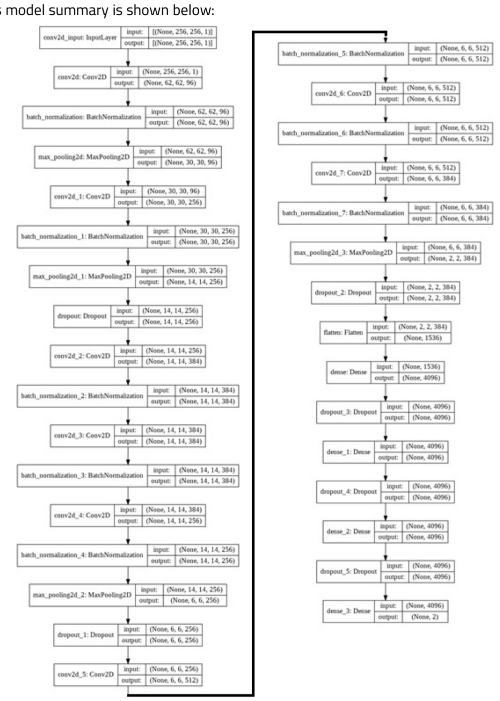
- We are using image processing techniques and convolutional neural networks (CNN) to detect lung cancer nodules.
- We use image processing to remove the noise that is present in the CT-scan images so that we can identify if there are any abnormal masses present in the lungs.
- Image is pre-processed using the median, Gaussian filters, and Adaptive Histogram Equalization to reduce noise and enhance the contrast. Cancer nodules are then segmented out of the image using Thresholding & Watershed segmentation.
- Lastly, CNN-Deep Learning is used for feature extraction to extract various parameters and the spatial properties of the image and to classify the detected nodules as malignant or benign.
- The block diagram of our proposed methodology is shown:



Block Diagram

CNN

- After image pre-processing, we developed our proposed model.
- This was done after implementing different architectures first and then analysing the key advantages from each model.
- Our proposed model consists of 12 layers of which 8 are convolution layers and 4 are fully connected layers.
- Our model's model summary is shown below:



```
Model: "sequential"
Layer (type)                 Output Shape              Param #
================================================================
conv2d (Conv2D)              (None, 62, 62, 96)        11712
batch_normalization (BatchNo (None, 62, 62, 96)        384
max_pooling2d (MaxPooling2D) (None, 30, 30, 96)        0
conv2d_1 (Conv2D)            (None, 30, 30, 256)       614656
batch_normalization_1 (Batch (None, 30, 30, 256)       1024
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 256)       0
dropout (Dropout)            (None, 14, 14, 256)       0
conv2d_2 (Conv2D)            (None, 14, 14, 384)       885120
batch_normalization_2 (Batch (None, 14, 14, 384)       1536
conv2d_3 (Conv2D)            (None, 14, 14, 384)       1327488
batch_normalization_3 (Batch (None, 14, 14, 384)       1536
conv2d_4 (Conv2D)            (None, 14, 14, 256)       98560
batch_normalization_4 (Batch (None, 14, 14, 256)       1024
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 256)         0
dropout_1 (Dropout)          (None, 6, 6, 256)         0
conv2d_5 (Conv2D)            (None, 6, 6, 512)         3277312
batch_normalization_5 (Batch (None, 6, 6, 512)         2048
conv2d_6 (Conv2D)            (None, 6, 6, 512)         6554112
batch_normalization_6 (Batch (None, 6, 6, 512)         2048
conv2d_7 (Conv2D)            (None, 6, 6, 384)         196992
batch_normalization_7 (Batch (None, 6, 6, 384)         1536
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 384)         0
dropout_2 (Dropout)          (None, 2, 2, 384)         0
flatten (Flatten)            (None, 1536)              0
dense (Dense)                (None, 4096)              6295552
dropout_3 (Dropout)          (None, 4096)              0
dense_1 (Dense)              (None, 4096)              16781312
dropout_4 (Dropout)          (None, 4096)              0
dense_2 (Dense)              (None, 4096)              16781312
dropout_5 (Dropout)          (None, 4096)              0
dense_3 (Dense)              (None, 2)                 8194
================================================================
Total params: 52,843,458
Trainable params: 52,837,890
Non-trainable params: 5,568
```

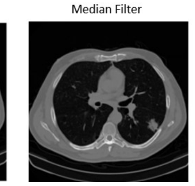
Model Summary

RESULTS

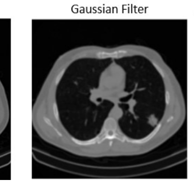
Image Pre-processing outputs:

Original Image

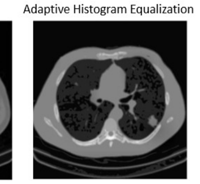
Median Filter



Gaussian Filter



Adaptive Histogram Equalization

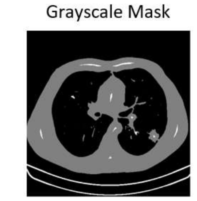


Segmentation Outputs:

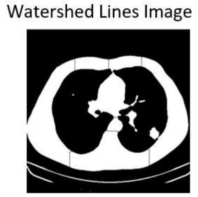
Binary Image

Complement of Binary Image

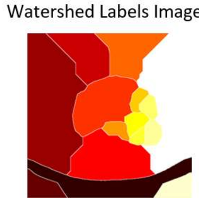
Grayscale Mask



Watershed Lines Image

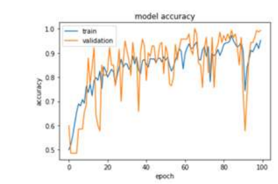


Watershed Labels Image

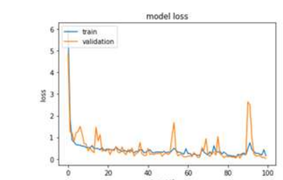


Segmented Output

CNN Results:



Model Accuracy plot



Model Loss plot

	precision	recall	f1-score	support
0	0.98	1.00	0.99	65
1	1.00	0.99	0.99	75
accuracy			0.99	140
macro avg	0.99	0.99	0.99	140
weighted avg	0.99	0.99	0.99	140

Classification Report

Model	Accuracy	Precision		Recall		F1 Score		Train loss	Validation loss
		0	1	0	1	0	1		
LeNet	88.57	0.84	0.91	0.91	0.85	0.87	0.88	2.4172e-05	0.9127
AlexNet	89.99	0.88	0.92	0.91	0.89	0.89	0.91	0.2752	0.9657
VGG16	51.43	0.46	0.00	1.00	0.00	0.63	0.00	0.6931	0.6930
Proposed Model	99.28	0.98	1.00	1.00	0.99	0.99	0.99	0.1812	0.0235

Comparison of evaluation metrics of various architectures

CONCLUSION

After comparing the output of K-means clustering and Watershed segmentation, we observe that the watershed segmentation algorithm can detect the objects(nodules) and plot a contour around the objects which represent the edge/boundaries of that object, whereas, in the k-means clustering algorithm, it only assigns a particular colour to each cluster but cannot differentiate between the objects. So, the watershed segmentation algorithm is found to be more advantageous than the K-means clustering algorithm.

Our proposed model was able to achieve higher accuracy (99.28) and performance when compared to other architectures and models we came across during literature survey.

When tested on a new dataset (test dataset), our model was able to classify 138 images out of 140 correctly with 2 wrong predictions.