DEPT. OF CEN

AMRITA SCHOOL OF ENGINEERING

III - B.TECH

CSE-AI

SEMISTER-5

21AIE303

Signal and Image Processing

END SEMISTER PROJECT

DATE: 25-01-2023

Machine learning And Deep Learning techniques for pulmonary nodule computer-aided diagnosis using CT images: A systematic review

Submitted by:

TEAM : 9

DIVITH PHOGAT (CB.EN.U4AIE20013)

KARNATI SAI PRASHANTH (CB.EN.U4AIE20027)

SAI RISHITH REDDY (CB.EN.U4AIE20036)

S.V.S.DHANUSH (CB.EN.U4AIE20068)

T.S.GANESH KUMAR (CB.EN.U4AIE20073)

Under the supervision of

(Dr.Sowmya V)

ABSTRACT

Keywords - Pulmonary nodules,computer-aided diagnosis (CAD)

The early detection of pulmonary nodules is crucial for the management and treatment of lung cancer. With the advancement in computer technology and intelligent algorithms, the effectiveness of computer-aided diagnosis (CAD) for pulmonary nodules has improved. This study evaluates the performance of various machine learning algorithms using a common dataset in the diagnosis of pulmonary nodules.we have used LIDC-IDRI dataset and its subset LUNA16 in this project.

Contents

Chapter 1

INTRODUCTION

In the field of Computer-Aided Diagnoses (CAD), such as medical image analysis [5], medical picture classification [6], and automatic medical report production [1], deep learning has made impressive strides [3]. These CAD systems with deep learning might aid doctors by providing second opinions and highlighting problematic areas in the health-care data [4].

Recent developments in CAD systems based on deep learning are also beneficial for lung cancer diagnosis. Currently, lung cancer is the leading cause of mortality world-wide [2]. However, it takes a long time for doctors to diagnose because of the enormous amount of computed tomography (CT) scans. The development of automated pulmonary nodules detection [7] and classification [8] algorithms has received a lot of attention. The diagnosis of early-stage lung cancer is considerably aided by such algorithms.

Chapter 2

LITERATURE

We have analyzed the review paper ”Machine learning techniques for pulmonary nodule computer-aided diagnosis using CT images: A systematic review” which published in 2019 by Haizhe Jin , Cheng Yu, , Zibo Gong , Renjie Zheng , Yinan Zhao , Quanwei Fu and we have found that Deep-learning-based CAD was found to be superior to conventional machine-learning-based CAD in terms of the number of published studies and algorithm performance. The best performances were as follows: feedforward neural network (FNN) and convolutional neural network (CNN) for detecting pulmonary nodules; region-based CNN (R-CNN) for the segmentation of pulmonary nodules; residual neural network (ResNet) for the classification of nodules and non-nodules; and deep neural network (DNN) for the classification of benign and malignancy.

Chapter 3

OBJECTIVE

There are different types of techniques and machine learning, deep learning models for pulmonary nodule detection like CNN, FNN, ELM, DBN, GAN and machine learning models like SVM, KNN and many more.

In our project we are going to compare three deep learning models namely 3D CNN and 2D CNN, LSTM, FNN and machine learning models like SVM and KNN and find out the best sutiable algorithm for pulmonary nodule detection.

Chapter 4

SVM

SVM stands for Support Vector Machine. It's a Supervised Learning algorithm. It can be used for both Classification and Regression tasks. It aims to create a decision boundary, known as a hyperplane, that separates data points into different classes. The algorithm chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors.

4.1 Working of SVM

4.1.1 Support Vectors

In order to classify the data between two classes there can be many boundaries to segregate them in n-dimensional space, but we need to find the optimal point that helps to classify the data points more accurately and this optimal line is known as a hyperplane. The dimension of this plane depends on the no of features the data have i.e if there the data have 2 features the hyperplane is a straight line. The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector.

The margin is the distance between the decision boundary (the line that separates the classes) and the closest data points from each class, called the support vectors. A larger margin indicates a better separation between the classes, while a smaller margin indicates a weaker separation.

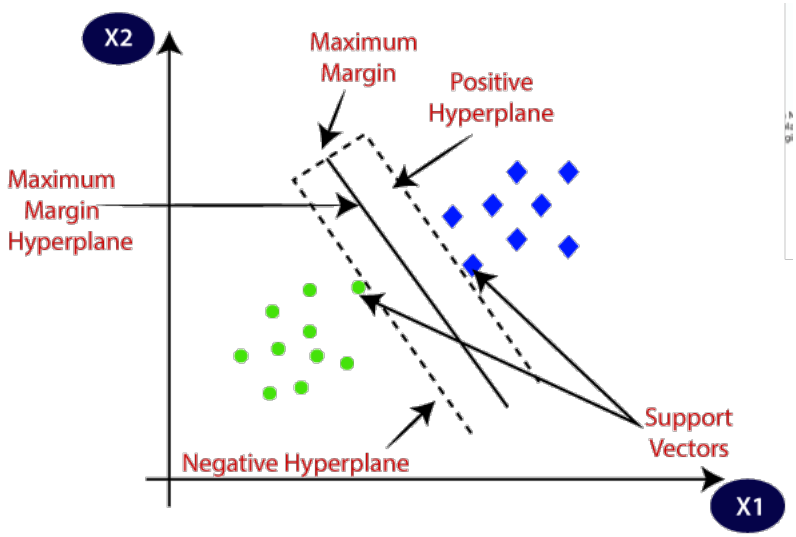


Figure 4.1: SVM

4.1.2 Classification of the data

The goal is to choose a hyperplane that maximizes the distance between the closest data points from each class, known as support vectors, in the provided dataset. This is known as the maximum margin hyperplane. SVM searches for the maximum marginal hyperplane in the following steps:

1. Construct a hyperplanes that seperate the classes in the best way. The figure displays three different hyperplanes (black, blue and orange) that are used to separate the classes. The black hyperplane has the best segregation of the classes, with the lowest classification error. On the other hand, the blue and orange hyperplanes have a higher classification error, indicating that they are not as effective in separating the classe

2. Pick the hyperplane that provides the greatest separation between the closest data points from each class, referred to as the support vectors, in the given dataset as shown in the right side figure.

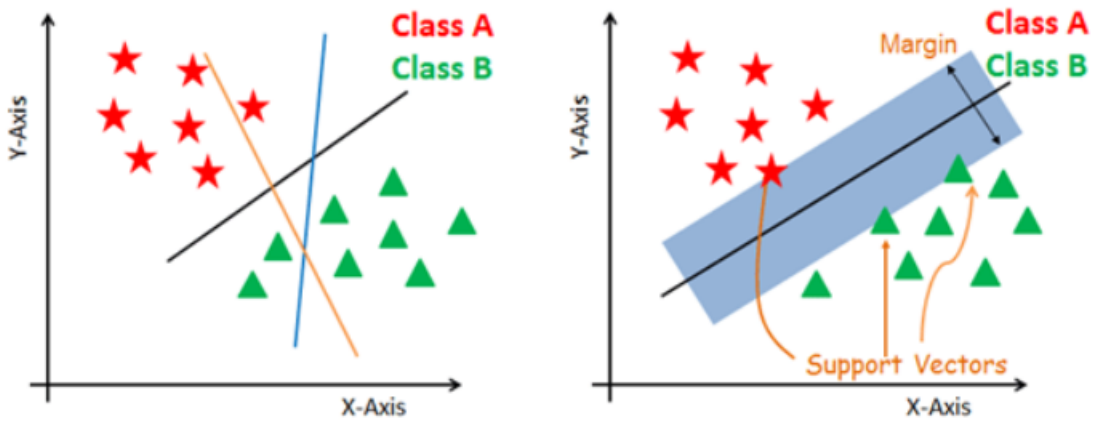


Figure 4.2

4.1.3 Linear vs Non-Linear Dataset

SVM can be classified into two types, Linear and Non - Linear depending on the dataset used. Linear SVM is used for linearly separable data, which means if the dataset can be classified into two classes by using a single straight line if the data cannot be separated by a straight line we will go for Non - Linear SVM.

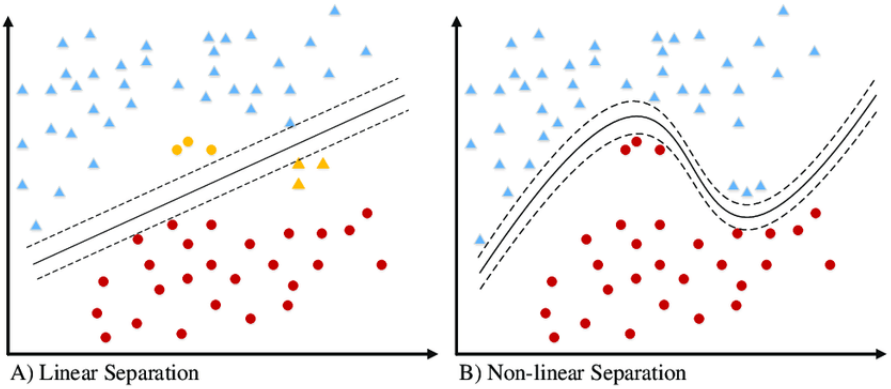


Figure 4.3: Linear vs Non-Linear Dataset

4.1.4 Parameters for SVM

- Kernel: The primary role of a kernel is to convert the input data of a dataset into the desired format.
- Regularization Parameter: The parameter C in SVM optimization controls the trade-off between maximizing the margin and minimizing the misclassification of training

examples. A higher C value results in a smaller margin and lower misclassification rate, while a lower C value leads to a larger margin and higher misclassification rate. If C is less than 1, it is considered a soft-margin SVM, and if it is greater than 1, it is considered a hard-margin SVM. We will be using C = 5 for our classification as it gives better accuracy compared to other values.

- Gamma Value: The gamma parameter in an SVM defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The default value for gamma is 1/(number of features * variance of the features), A high gamma value results in a decision boundary that is closer to the training examples, while a low gamma value results in a decision boundary that is farther away from the training examples.

4.1.5 Types of kernels

The SVM algorithm can handle non-linear classification problems by using a kernel function to transform the input data into a higher dimensional space, where a linear decision boundary can be applied. Different types of kernel functions are available such as polynomial kernel, radial basis function (RBF) kernel, and linear kernel. These kernel functions can be chosen based on the characteristics of the dataset, whether it is linear or non-linear. In this way, the SVM algorithm can create a non-linear decision boundary to separate the classes in the best possible way.

- Linear Kernel: A linear kernel in SVM measures the similarity between two observations by calculating the dot product of their feature vectors. This dot product is obtained by adding up the products of each pair of feature values from the two observations, providing a measure of how similar the observations are based on their individual feature values and it is used in classifying linearly separable data.

- Radial basis function (RBF) kernel: The RBF kernel is a popular technique used in SVM classification, known for its ability to map non-linear decision boundaries by transforming the input space into a higher-dimensional space. Its versatility and ability to handle complex non-linear data make it a widely used option in many classification tasks.

We will be using Radial Basis Function (RBF) kernel in our project.

4.2 Results and Analysis

By using the above-mentioned parameter we have obtained an 81.47 accuracy. The accuracy may varies depending on the dataset and also the train, test split ratio Below is the confusion matrix for our model.

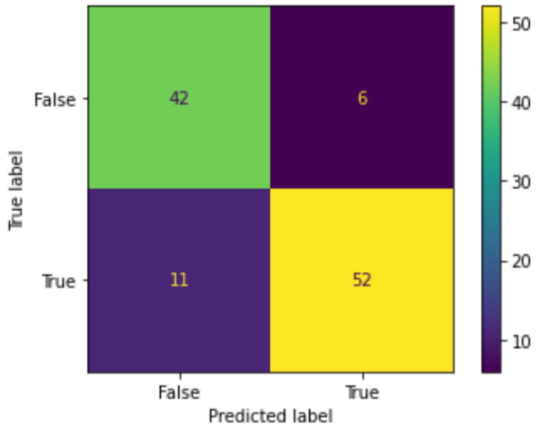


Figure 4.4: Confusion Matrix

The Precision, Recall, and F1 Scores are as follows,

- Precision: 0.918
- Recall: 0.889
- F1 Score: 0.903

Chapter 5

KNN

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for both classification and regression tasks. The basic idea behind KNN is that an object (i.e. a data point) is classified based on the majority class of its k-nearest neighbors, where k is a user-specified parameter. The algorithm typically uses some type of distance metric (e.g. Euclidean distance) to determine the ”nearest” neighbors, and the choice of k will affect the final classification of the object in question.

5.1 WORKING OF KNN

In the context of detecting pulmonary nodules (small, abnormal growths in the lungs that can indicate lung cancer), KNN would likely be used as a classification algorithm to determine whether a given nodule is benign (non-cancerous) or malignant (cancerous) based on the characteristics of other nodules that have been previously identified and labeled as such.

The basic idea behind KNN is that an object (in this case, a pulmonary nodule) is classified based on the majority class of its k-nearest neighbors, where k is a user-specified parameter. The algorithm typically uses some type of distance metric (e.g. Euclidean distance) to determine the ”nearest” neighbors, and the choice of k will affect the final classification of the object in question.

In order to use KNN for pulmonary nodule detection, a dataset of previously identified and labeled nodules would be required. This dataset would be used to train the algorithm and would include various characteristics of the nodules, such as size, shape, and texture.

Once the algorithm has been trained, it can then be used to classify new, unseen nodules based on their characteristics.

5.1.1 CLASSIFICATION OF DATA

The classification of data using the K-Nearest Neighbors (KNN) algorithm is done by comparing the characteristics of a new, unseen nodule to the characteristics of previously identified and labeled nodules in the training dataset. The algorithm uses a distance metric (such as Euclidean distance) to determine the k-nearest neighbors of the new nodule, where k is a user-specified parameter.

Once the k-nearest neighbors have been identified, the algorithm looks at the majority class of these neighbors, and assigns the new nodule to that class. So, if the majority of the k-nearest neighbors are benign nodules, the new nodule will be classified as benign. Similarly, if the majority of the k-nearest neighbors are malignant nodules, the new nodule will be classified as malignant.

The choice of k, distance metric and weighting scheme will affect the final classification of the new nodule. A higher value of k will result in a more conservative classification, while a lower value of k will result in a more liberal classification. The appropriate distance metric and weighting scheme will depend on the specific characteristics of the data.

5.1.2 PARAMETERS FOR KNN

The main parameter is the number of nearest neighbors, k, that the algorithm uses to classify new nodules. This parameter is often chosen by the user, and a higher value for k will result in a more conservative (i.e., less sensitive) classification, while a lower value for k will result in a more liberal (i.e., more sensitive) classification.

- Another important parameter is the distance metric used to determine the ”nearest” neighbors. Common distance metrics include Euclidean distance, Manhattan distance and Minkowski distance. The appropriate distance metric will depend on the specific characteristics of the data.

- A third parameter is the weighting scheme used, which can be uniform or distance

based. This parameter tells the algorithm how to weight the influence of the different nearest neighbours on the final decision.

5.2 RESULT AND ANALYSIS

By using above mentioned parameter we got different accuracies for different values of k

- For k=5 We got 85.2
- For k=10 we got 84.68

We got the Best accuracy for k=5

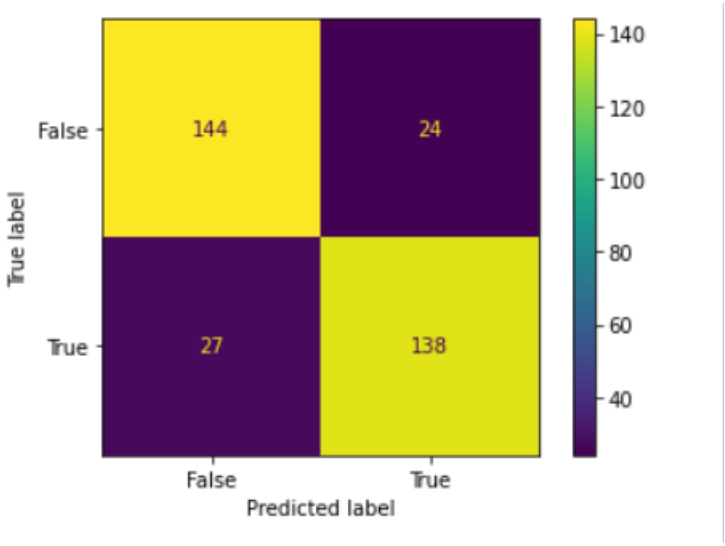


Figure 5.1: Confusion Matrix

Chapter 6

FEEDFORWARD NEURAL NETWORK

Feed-forward neural network or simply fnn is type of artificial neural network.In FNN the connection between neuons doesn't form any loop and all information flows in forward direction only. the input data enter through hidden layers and finally exits output layer. A Feedforward neural network consists of four components

- **Input layer:** It contains neurons that received from the user.It is passed on to the next layer which is called hidden layer. the size of input layers will be same as the number of variables present in the dataset
- **Hidden layer:** It is the intermediate layer. The numbers of neurons for hidden layer is user-defined and numbers of hidden layers is also user-defined. The neurons will transformations to the input before passing them to the next layer. As the model is trained the weights are also updated to be more accurate.
- **Output layer:** It is the predicated features and size of the output layer depends on the number of classes present in our dataset
- **Neuron Weights**: The neuron weights are small values ranging from 0 to 1 it tells connection strength between two neurons

6.1 WORKING OF FNN

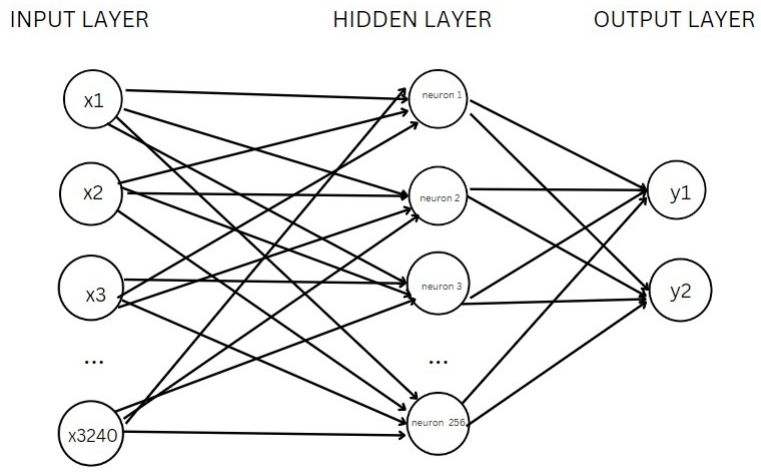


Figure 6.1: neural network architecture

for every neuron there is coresponding weight w and it multipled input of the corresponding of the weight and then will add bias b the corresponding equation looks this below equation is the first neuron in hidden layer

$$Z = [w_1x_1 + w_2x_2 + w_3x_3 + ... + w_mx_m] + b \tag{6.1}$$

We can generalize the above equation for all in neurons present in the hidden layer

$$Z = W.X + B \tag{6.2}$$

Where W is the weight matrix where rows will be the neurons of the hidden layer and columns will be the inputs and X contains and it is a column vector and b contains bias and it is a column vector the variable Z goes to activation layer and after the activation layer the output will be input to next hidden layer if FNN has multiplule hidden layers otherwise it goes to hidden layer otherwise it goes to output layer.

6.2 PARAMETERS FOR FNN

the input size for FNN Model is 3240 and we selected the number of hidden layers as 1 and the number of neurons for the respective hidden layer is 256.So the size of the W matrix will be $[256 \times 3240]$ and the input matrix X is column vector $[3240 \times 1]$ and bias matrix B will be $[256 \times 1]$.We have activation function as Relu.Since our model is a binary classification and 2 neurons for the output layer

6.3 Results For FNN

```
Epoch: 1 Validation accuracy: 98.94736842105263 Epoch Time: 0m 1s
Epoch: 2 Validation accuracy: 98.94736842105263 Epoch Time: 0m 1s
Epoch: 3 Validation accuracy: 98.94736842105263 Epoch Time: 0m 1s
Epoch: 4 Validation accuracy: 98.94736842105263 Epoch Time: 0m 1s
Epoch: 5 Validation accuracy: 98.94736842105263 Epoch Time: 0m 1s
Epoch: 6 Validation accuracy: 98.94736842105263 Epoch Time: 0m 1s
```

Figure 6.2: Accuracy

	precision	recall	f1-score	support
0	0.99	1.00	0.99	94
1	0.00	0.00	0.00	1
accuracy			0.99	95
macro avg	0.49	0.50	0.50	95
weighted avg	0.98	0.99	0.98	95

Figure 6.3: Classification report

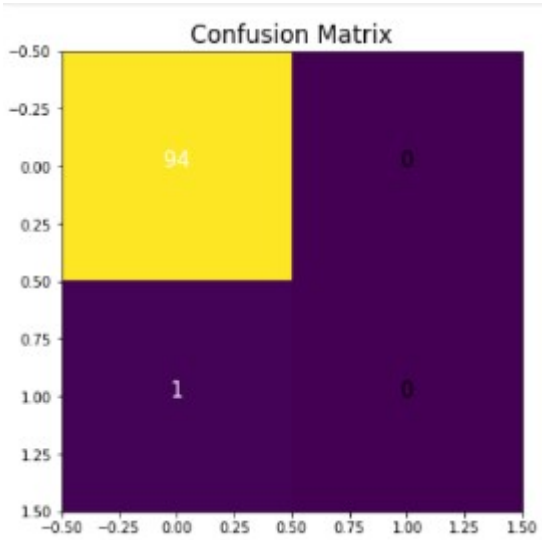


Figure 6.4: Confusion matrix

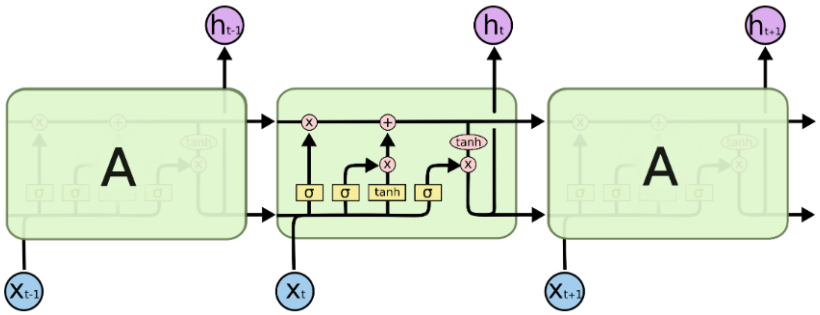
6.4 Analysis For FNN

We got 98.94% accuracy and our 1st epoch takes around 20 mins to run and we will cache that epoch so that remaining epoch will be faster.Our model can correctly classify the negative class but it fails to classify the positive class.This can be attributed to the dataset imbalance we have around 5.4 lakhs of benign and 1600 malignant data.

Chapter 7

LSTM

LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network (RNN) architecture that is able to capture long-term dependencies in sequential data, making it well-suited for tasks such as natural language processing and speech recognition. LSTMs achieve this by introducing a memory cell, gates that control the flow of information into and out of the cell, and a hidden state that propagates information throughout the network. These features enable LSTMs to selectively remember or forget information from the input sequence, allowing them to learn and make predictions based on long-term context.



Recurrent neural networks (RNNs) are extended by LSTM networks, which were primarily developed to address RNN failure scenarios. It is a network that processes the current input while taking into account the previous output (feedback) and temporarily keeping the prior input in its memory. Out of all of its uses, voice processing, non-Markovian control, and musical creation make up the majority of its uses. RNNs do, however, have shortcomings. In the beginning, it is unable to keep data for a longer length of time. Sometimes, in order to forecast the present output, a reference to specific data that was saved a long time ago is needed. However, because there is no way to de-

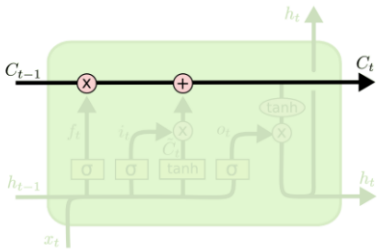
cide which context has to be remembered and which needs to be carried forward, RNNs are unable to manage such long-term dependencies. Exploding gradients and disappearing gradients, which happen while a network is being trained via backtracking, are further problems with RNNs. Long Short-Term Memory (LSTM) was introduced as a result. The training model is kept unmodified, and the vanishing gradient problem has been nearly entirely eliminated.

7.1 Working of LSTM

7.1.1 Memory Cell

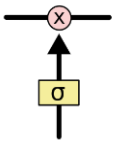
The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state acts like a conveyor belt. It runs straight down the entire chain, with only someminor linearinteractions.



The basic function of the meomory unit is forgetting and Remembering.

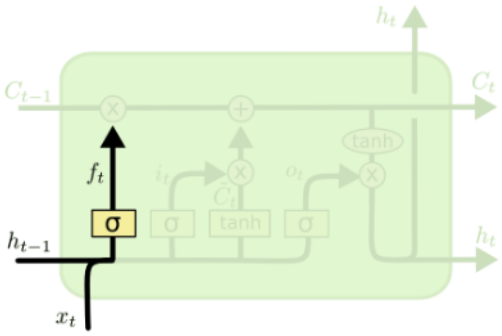
Gates are a way to optionally let information through. They are composed out of a sigmoid function is applied which give the answer between 0 -1 and a pointwise multiplication operation which multiplies the vectors pointwise.



The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. For value =0 ,it will not allow anything to go through it and if the valus is 1 means it will allow everything to pass through.

7.1.2 Forget Gate

A forget gate in a Long Short-Term Memory (LSTM) network is used to decide what information should be discarded or ”forgotten” from the previous state of the LSTM cell. The forget gate is a sigmoid layer that takes as input the current input and the previous hidden state and produces a value between 0 and 1 for each element in the previous cell state. Values close to 0 indicate that the corresponding elements in the previous cell state should be forgotten, while values close to 1 indicate that the corresponding elements should be kept. The output of the forget gate is then multiplied element-wise with the previous cell state to update it.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

7.1.3 Input Gate

An input gate in a Long Short-Term Memory (LSTM) network is used to decide what new information should be added to the current state of the LSTM cell. The input gate is composed of two parts:

1. A sigmoid layer that takes as input the current input and the previous hidden state and produces a value between 0 and 1 for each element in the cell state. These values are used as a gate or switch to decide which new information should be added to the cell state.

2. A tanh layer that also takes as input the current input and the previous hidden state and produces a new ”candidate” cell state.

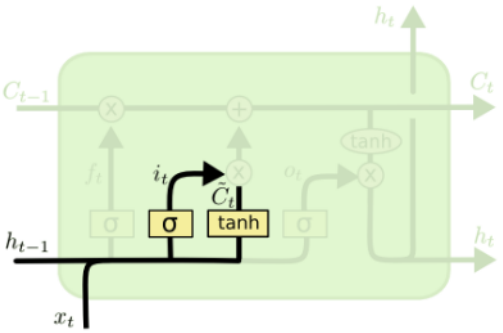
The input gate formula can be represented as :

i_t = (W_ix_t + U_ih_t-1 + b_i) (sigmoid)

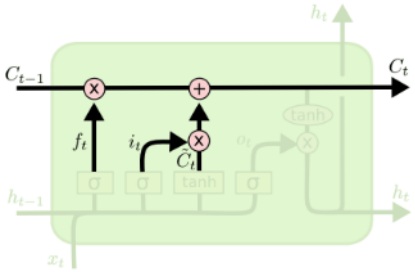
c_candidate_t = tanh(W_cx_t + U_ch_t-1 + b_c) (tanh)

where, i_t is the input gate activation at time t c_candidate_t is the candidate cell state at time t x_t is the input at time t h_t-1 is the previous hidden state W_i, U_i, b_i are the weights and bias for the input gate W_c, U_c, b_c are the weights and biases for the candidate cell state

The input gate then multiplies the i_t and the c_candidate_t to update the current cell state by adding the new information.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

7.1.4 Output Gate

An output gate in a Long Short-Term Memory (LSTM) network is used to decide what information from the current state of the LSTM cell should be outputted. The output gate is a sigmoid layer that takes as input the current input, the previous hidden state, and the current cell state and produces a value between 0 and 1 for each element in the cell state. These values are used as a gate or switch to decide which information should be outputted. The output gate formula can be represented as:

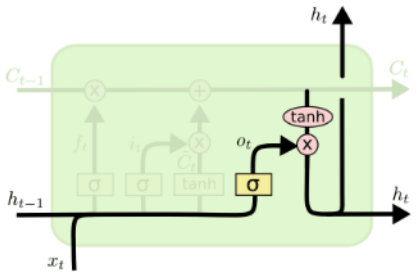
o_t = (W_ox_t + U_oh_t-1 + V_oc_t + b_o) (sigmoid)

where, o_t is the output gate activation at time t x_t is the input at time t h_t-1 is the previous hidden state c_t is the current cell state W_o, U_o, V_o, b_o are the weights and bias for the output gate

The output of the output gate is then multiplied element-wise with the current cell state and passed through a tanh layer to produce the final output, h_t.

h_t = o_t * tanh(c_t)

This final output is the output of the LSTM cell at time t, and it is used to predict the next word in a sentence, next frame in a video, and so on.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

7.2 LSTM Model

In an LSTM deep learning model, the LSTM layers are stacked one on top of the other, creating a deep network. This allows the model to learn hierarchical representations of the data, and to make predictions based on both the current input and the previous layers' hidden states. The depth of the network can be increased or decreased depending on the complexity of the task and the amount of data available for training.

These models have the ability to process and analyze long-term dependencies in the data, which makes them well-suited for tasks that involve understanding sequential information.

7.2.1 Dataset

we have used segmented images of hand-picked Discom files. DICOM is an acronym for Digital Imaging and Communications in Medicine. Files in this format are most likely saved with either a DCM or DCM30 (DICOM 3.0) file extension, but some may not have an extension at all.

DICOM is both a communications protocol and a file format, which means it can store medical information, such as ultrasound and MRI images, along with a patient's information, all in one file. The format ensures that all the data stays together, as well as provides the ability to transfer said information between devices that support the DICOM format.

This is an sample image of the DICOM file.

In our dataset we have used 700 Dicom files for the training and 200 Dicom files for validating. we have used only few files for the faster computation of the program. Taking a lot of dataset increases the runtime of the algorithm.
The Dataset is available in my kaggle repository.
Before training my model we have pre-processed our dataset, we have used transformation function with one key-value pair. The key is ”train” and the value is a compose object created using the transforms.Compose() function from the PyTorch transforms library. The compose object includes three transforms applied in order:

1. transforms.Resize((32, 32)): resizing the image to 32x32 pixels.
2. transforms.Grayscale(): converting the image to grayscale.
3. transforms.ToTensor(): converting the image to a tensor.

This composed transformation will be applied to images in the training set. It resizes the image to 32x32 pixels, converts it to grayscale and converts it to a tensor.

7.2.2 Model

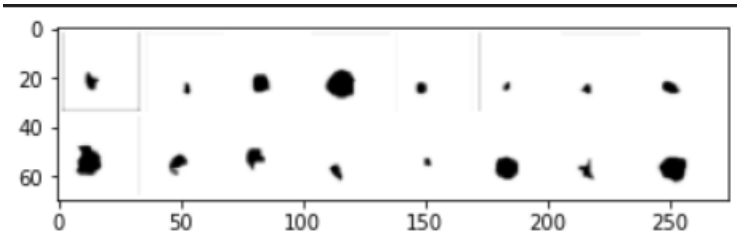
we have used a custom PyTorch module called ”LSTM” which is a subclass of nn.Module. The __ini__ method of the LSTM class takes 4 arguments:

1. input_dim: The number of input features.
2. input_dim: The number of input features.

3. input_dim: The number of input features.

4. n_layers: The number of layers in the LSTM.

In the __init__ method, it initializes the number of layers and hidden dimension and defines the LSTM cell using nn.LSTM and nn.Linear from the Pytorch library. The nn.LSTM takes the input features, hidden features and number of layers as input. The nn.Linear layers is used to transform the output of the LSTM to the final output.
The forward method is called when the model is run on inputs. It takes the input x and applies the LSTM to it and Linear layer to get the output. It also initializes the hidden state and the cell state of the LSTM with zeros and takes the last hidden state of the LSTM as output. And it also returns the output from the linear layer.
we have instantiated our input_dim as 32 , OUTPUT_DIM as 2 , HIDDEN_DIM as 128 and n_layer as 15.
This is a random training image of the dataset.

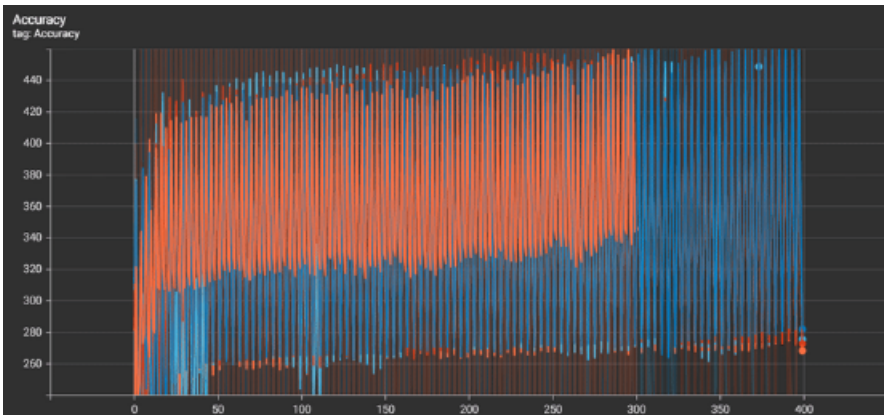


7.3 Analysis

we have predicted the accuracy of our model. Accuracy is a commonly used metric to evaluate the performance of a classification model. It is defined as the proportion of correct predictions made by the model out of all the predictions made.

The formula for accuracy is:
(Number of correct predictions) / (Total number of predictions)

For example, if a model makes 100 predictions and 80 of them are correct, the accuracy of the model is 80



we have try all the possible ways of to get the best classification accuracy for our pulmonary nodule classification. we getting an average of 80% accuracy for our model. we have also calculated the accuracy per epoch, loss and loss per epoch.

Our LSTM model got a pretty Stats.If we are getting good statistics for your LSTM model, it is likely that the model is performing well on the classification task at hand. Depending on the specific task and dataset, good statistics could include a high accuracy, high precision, high recall, or high area under the receiver operating characteristic curve (AUC-ROC).

It is important to keep in mind that accurate statistics can vary depending on the specific task and dataset. For example, in a medical diagnosis task where the negative class is much more frequent than the positive class, a model that has high accuracy but low recall (meaning it doesn't correctly identify all positive cases) might not be considered accurate.

It is also important to compare the performance of your LSTM model to other methods that have been used for the same task to see how it compares to the state-of-the-art.

It would also be useful to evaluate the uncertainty of the model, how it performs on edge cases or samples that are difficult to classify, and how it generalizes to unseen data, and

to test it in a real-world scenario.

7.4 Conclusion

LSTM-based image classification can be a suitable method for computer-aided diagnosis of pulmonary nodules. LSTMs, a type of recurrent neural network, are well-suited for image classification tasks because they can effectively learn spatial relationships between pixels in an image. They are also able to retain information from previous frames or images, which can be useful in analyzing sequential data such as medical images.

Additionally, LSTMs can be trained on large amounts of data and can handle high-dimensional inputs such as images, which is important for pulmonary nodule classification, as these nodules are typically small and may be difficult to identify using traditional methods.

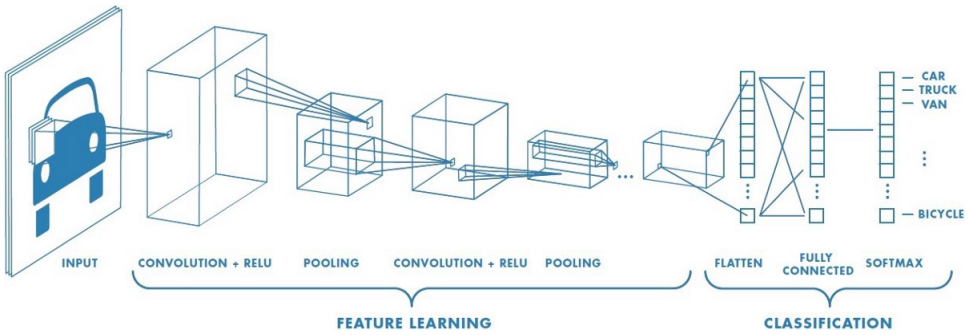
It is also important to note that LSTM-based image classification is just one of the many methods that can be used for pulmonary nodule classification and computer-aided diagnosis. Other methods such as CNNs, and other deep learning architectures have also been used with good results. The best approach will depend on the specific task, the available dataset and the resources.

It is also important to keep in mind that a computer-aided diagnosis system should be used in conjunction with radiologists' expertise and not as a replacement.

Chapter 8

CNN

A Convolutional Neural Network (CNN) is a Deep Learning algorithm that can take in an input tensor, assign importance (learnable weights and biases) to various aspects/objects in the tensor, and be able to differentiate one from the other.

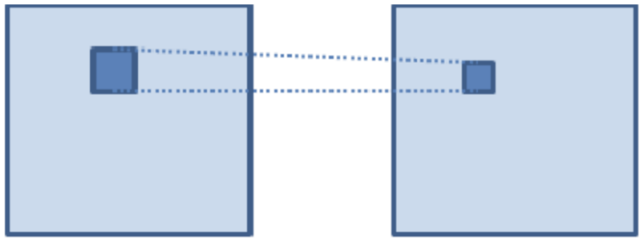


A CNN works by passing data through multiple convolutions using different filters. This 'extracts' features from the data and allows a simple neural network to process these extracted features and generate a prediction.

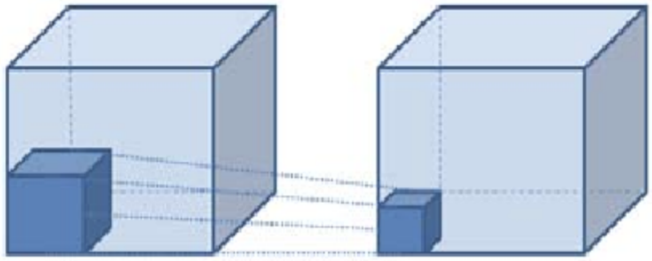
The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

8.1 2D vs 3D



(*a*) 2D convolution



(*b*) 3D convolution

3. Comparison of (a) 2D convolution and (b)

The difference between 2D and 3D CNNs is the data format:

1. 1) CNNs work on 2D tensors (images). They have a 2D kernel that passes over an image based on the given hyperparameters.

2. 3D CNNs work a little differently, i.e., on 3D data. They select a part of the 3D data, a collections voxel (similar to the concept of pixels in 2D) and perform convolutions on these with a 3D kernel.

8.2 3D CNN

The model is built using Pytorchs Conv3d and MaxPool3d classes. These perform 3D convolutions on voxel-based data.

The dataset is obtained from CT scans, which are 3D in shape. The input data is created by taking voxels around annotated points in our dataset. A fixed range is set around the point of interest and a cubical tensor is extracted. This 3D tensor can be passed into a 3D CNN.

```
Train Loss: 0.216 | Train Acc: 99.79%
Train - correct pos: 0/2 | correct neg: 968/968

Val. Loss: 1.363 |  Val. Acc: 99.07%
Val. - correct pos: 0/1 | correct neg: 107/107
```

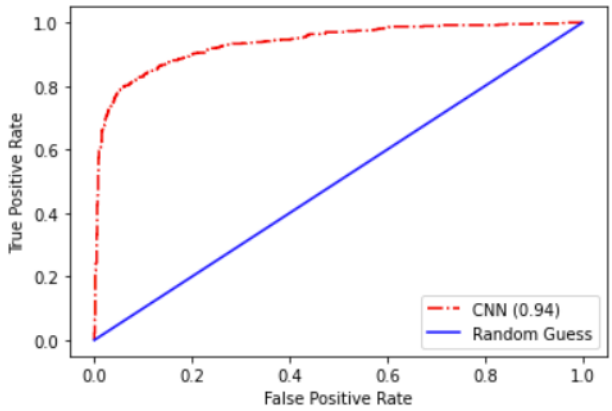
The trained model produces a 99% accuracy. While this sounds great, when we analyze the class data points, we see that non-malignant data has 1075 data points compared to just 3 malignant data points. We cannot say for sure how well the model will perform on real data.
We need to turn to 2D CNNs, with a more balanced dataset.

8.3 2D CNN

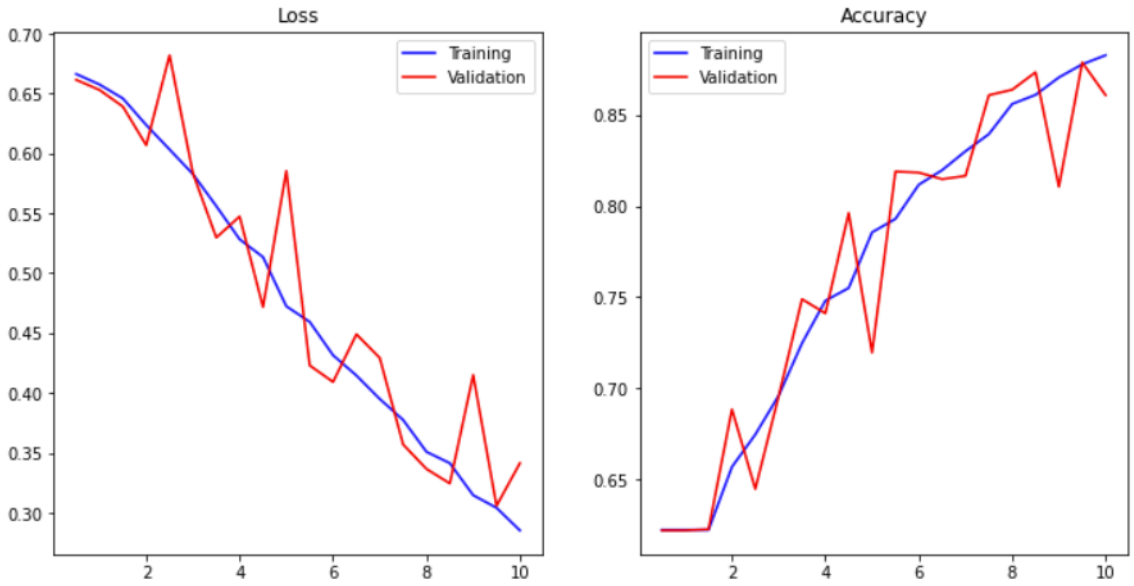
The 2D CNN model is a model inspired by SqueezeNet, built using keras. It's performance is very similar to that of AlexNet, but is approximately 510x smaller
The dataset is obtained from CT scans. While it is 3D, we can simply interpret it as an array of 2D shapes. We pass 2D images sequentially into our model.

	precision	recall	f1-score	support
0	0.91	0.86	0.88	1041
1	0.79	0.87	0.82	632
accuracy			0.86	1673
macro avg	0.85	0.86	0.85	1673
weighted avg	0.87	0.86	0.86	1673

The results are more promising with a 2D model. We get a fairly high accuracy with decent F1 scores on both classes.



The ROC curve comes out good. But the issue comes in when we look at our training metric graphs.



We see fluctuations in our loss and accuracy. This can mean a number of things.

- The model isn't regularized properly
- Model is overfitting
- Model is big and dataset is small
- Batch size is too small

Fixing these might improve the graph. Presently the model is doing great on unseen data so no more adjustments have been made.

8.4 2D or 3D

We cannot say for sure what model would be better.
While 2D CNNs can be applied to more types of data than just CT scans, 3D CNNs might be able to extract information better and get a better, more complete view of data as it exists in the real world (which is also 3D).
To make a definitive judgement on what is better, we need to improve both models and fix their datasets. Only then can a conclusion be made.
But at the same time, for now, the 2D model seems to give a good result and we can go with that for reasonably accurate predictions for both classes.

Chapter 9

Conclusion

Deep Learning has always been an effective algorithm for working with images, especially CNNs.

For FNN even though got high accuracy it fails to classify the malignant data.In medical field classification of malignant data is critical.The reason for this likely is due to data imbalance

CNNs built-in convolutional layer reduces the high dimensionality of images without losing its information. This makes CNNs extremely good at classifying images

We see this behaviour reflected in the experimental results. The 2-D CNN gets the highest accuracy at 86% compared to other models.

Bibliography

[1] Hareem Ayesha, Sajid Iqbal, Mehreen Tariq, Muhammad Abrar, Muhammad Sanaullah, Ishaq Abbas, Amjad Rehman, Muhammad Farooq Khan Niazi, and Shafiq Hussain. Automatic medical image interpretation: State of the art and future directions. *Pattern Recognition*, 114:107856, 2021.

[2] Jacques Ferlay, Isabelle Soerjomataram, Rajesh Dikshit, Sultan Eser, Colin Mathers, Marise Rebelo, Donald Maxwell Parkin, David Forman, and Freddie Bray. Cancer incidence and mortality worldwide: sources, methods and major patterns in globocan 2012. *International journal of cancer*, 136(5):E359–E386, 2015.

[3] Brenden M Lake, Wojciech Zaremba, Rob Fergus, and Todd M Gureckis. Deep neural networks predict category typicality ratings for images. In *CogSci*, 2015.

[4] Muhammad Imran Razzak, Saeeda Naz, and Ahmad Zaib. Deep learning for medical image processing: Overview, challenges and the future. *Classification in BioApps*, pages 323–350, 2018.

[5] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[6] Nguyen Xuan Thao, Mumtaz Ali, Le Thi Nhung, Hemant Kumar Gianey, and Florentin Smarandache. A new multi-criteria decision making algorithm for medical diagnosis and classification problems using divergence measure of picture fuzzy sets. *Journal of Intelligent & Fuzzy Systems*, 37(6):7785–7796, 2019.

[7] Hongtao Xie, Dongbao Yang, Nannan Sun, Zhineng Chen, and Yongdong Zhang.

Automated pulmonary nodule detection in ct images using deep convolutional neural networks. *Pattern Recognition*, 85:109–119, 2019.

[8] Wentao Zhu, Chaochun Liu, Wei Fan, and Xiaohui Xie. Deeplung: Deep 3d dual path nets for automated pulmonary nodule detection and classification. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 673–681. IEEE, 2018.

Thank You