3D Convolutional Neural Network and K-Nearest Neighbour with Radiomics Features for Covid Classification

K. Kirubananthavalli[1*] and Dr. P. Sundareswaran[2]

[1]Assistant Professor, Department of Computer Science and Engineering, P. S. R. Engineering College, Sivakasi, Tamil Nadu, India*
[2]Assistant Professor, Department of Computer Science and Engineering, Manonmaniam Sundaranar University, Tirunelveli, Tamil Nadu, India
[1]kkirubananthavalli06@gmail.com, [2]psundareswaran@msuniv.ac.in*

KEYWORDS

2D CT, 3D CT, Pyradiomics, GLCM, 3D-CNN, KNN.

ABSTRACT

For the purpose of identifying COVID-19 lung affection When it comes to finding tumors, computed tomography plays a crucial role. However, the intricacy of the CT makes categorization a formidable challenge for the doctor. From 2D CT, just a handful of characteristics may be retrieved. Compared to 2D CT, 3D CT offers additional features and equivalent diagnostic performance. Learn how to spot and categorize a COVID-19 lung ailment with the use of 3D computed tomography (CT) in this article. To acquire a plethora of details from CT scans, radiologists utilize data-characterization algorithms. Discovering illness traits is possible with the use of these radiomics properties. To extract GLCM features, we utilize pyradiomics, an open-source library for Python. The 3D CT data is processed using a combinations model that incorporates GLCM, CNN, and KNN features. To get a stronger three-dimensional volumetric representation, 3D-CNN is used. It is the job of the final 3D-CNN layer to learn an accurate image representation. From that layer, we extract the characteristics and feed them into the KNN classifier so it can make more predictions. This approach has been shown to enhance accuracy by as much as 98.9%.

I. INTRODUCTION

Massive or aberrant cell proliferation in a CT area characterizes the COVID-19 lung illnesses. This has the potential to infect people of all ages. It is possible to distinguish between malignant and benign forms of COVID lung disease. Primary tumors are those that originate in the CT and grow there; secondary or metastatic tumors are those that originate in other regions of the body and spread to the CT. The lives of patients depend on accurate categorization. When it comes to complicated structures like the CT, CNN is one of the numerous algorithms utilized for tumor segmentation and classification. In order to provide an accurate diagnosis, better segmentation and categorization of COVID lung diseases are required. The aberrant tissues may be identified by the 3D convolutional neural network model due to the differences in pixel values linked to the biological structural information.

The K-Nearest-Neighbour (KNN) method is the simplest machine learning technique for categorization. Finding the K-nearest matches in the training data and making predictions based on their labels is how the classification is accomplished. During training, the method just keeps the parameter's labels and feature vectors that are required for image training. The unlabeled query point in the classification job is given the k closest neighbors. The image's K-nearest neighbors' labels form the basis of its categorization. When k=1, the image is tagged as the closest class. We use the Euclidean distance computation technique to find the distance between each row of training data and the test data.

This work presents a multimodal approach to classification accuracy that integrates GLCM Features, 3D CNN, and KNN classifier. The benefits of both approaches are combined in the 3D-CNN-KNN model. Convolutional neural networks (CNNs) use filters to extract data from input images. Activation maps will be the final format for the retrieved features. To extract GLCM features, we utilize pyradiomics, an open-source library for Python. In order to make additional predictions, the GLCM features and the features generated from the 3DCNN's final layer (the Feature Layer) are fed into the KNN classifier. Segmenting and accurately predicting COVID lung affection is now possible using 3D-CNN and the KNN model.

Here is the breakdown of the remaining tasks: The relevant literature is examined in Section II. Section III lays forth the framework and methods. Section IV delves into the findings of the experiments. Section-V presents the conclusion.

II. RELATED WORKS

In the context of the COVID-19 pandemic, a significant body of research has focused on vaccine uptake, hesitancy, and the effectiveness of booster doses against emerging variants. Viswanath et al. explore the individual and social determinants of COVID-19 vaccine uptake, highlighting the complexity of factors influencing vaccination decisions [1]. Similarly, Biswas et al. address vaccination hesitancy among healthcare workers, emphasizing the need for targeted interventions to increase vaccine acceptance within this crucial population [2]. The nature and extent of this hesitancy are further examined in studies by Holzmann-Littig et al. and Gilboa et al., both of which focus specifically on healthcare workers in Germany and the broader healthcare community, respectively [3-4].

As the Omicron variant emerged, research pivoted to evaluate the efficacy of existing vaccines against this new challenge. Külper-Schiek et al. provided a living systematic review analyzing the protection vaccines offer against both mild and severe COVID-19, including against the Omicron variant [5]. Chenchula et al. and Le et. al. further contributed to this discussion by systematically reviewing the efficacy of booster doses against Omicron, demonstrating a nuanced understanding of vaccine performance amidst evolving viral strains [6-7]. Ai et al. compared the neutralizing sensitivity of the Omicron variant to immune sera elicited by vaccines, offering critical insights into the variant's potential impact on vaccine effectiveness [8].

The broader implications of these findings are significant, as evidenced by the work of Harder et al., who presented interim results of a living systematic review on the efficacy and effectiveness of COVID-19 vaccines against SARS-CoV-2 infection [9]. The ongoing debate around the need for COVID-19 vaccine boosters, as discussed by Khan et al., highlights the dynamic nature of the pandemic and the corresponding public health responses [10]. This is further illustrated by Magen et al., who examined the impact of administering a fourth dose of the BNT162b2 mRNA COVID-19 vaccine in a nationwide setting [11].

Beyond the clinical and epidemiological aspects, the pandemic has also shed light on the social and psychological dimensions of health, such as stigma and mental health outcomes. Kane et al. and Dar et al. explored health-related stigma outcomes for high-burden diseases in low and middle-income countries, and the stigma experienced by COVID-19 survivors in Kashmir, India, respectively [13-14]. Additionally, Shahnawaz et al. and Lin and Wang (2023) investigated the relationship between intolerance of uncertainty, symptom severity in COVID-19 patients, and the lessons learned from the stigma of COVID-19 survivors, offering critical insights into the pandemic's psychological impact [15-16].

These studies collectively underscore the multifaceted challenges posed by the COVID-19 pandemic, from vaccine uptake and hesitancy to the effectiveness of booster doses against emerging variants and the psychological impact on individuals and communities. The ongoing research in these areas continues to inform public health strategies and interventions aimed at mitigating the impact of the pandemic.

III. PROPOSED METHODOLOGY

The suggested approach employs a combinational algorithm for categorization. Noise is removed from the 3D image and the bias fields is adjusted using N4ITK as part of the pre-processing steps. The region expanding approach is used for the segmentation. It uses 3D CNN and GLCM techniques to extract the features. After feature extraction, the KNN classifier is fed the data it needs to make predictions shown in Figure 1.

Python is the language of choice for the suggested method's implementation of CT Covid lung disease detection and categorization. The following are some of the reasons why Python has been selected to implement the suggested model:

1. Python code is easier to understand and smaller.
2. Python's data structure is intuitive and top-notch.
3. Python is open-source and offers additional data sets and visual packages.

So, instead of development, the work that is being suggested uses python. Python modules such as NumPy, scipy, scikit-learn, TensorFlow, and Keras are used to execute the suggested model. Below you can see the design of the model that has been suggested:

a) Datasets

The NIFTI format is used in our model for 3D images. Each of the 300 images in the collection has dimensions of 155*240*240. About half of the images are benign, while the other half are cancerous, sample dataset CT images shown in Figure 2. There are two halves to the dataset: one is used for training, and the other is for testing. To segment the covid affection from the three-dimensional (3D) CT input images, which have been labeled using the train, est and validation modality of the dataset, this dataset is used for training and testing purposes. We have developed a system that uses NII images for 3D COVID lung affection categorization.

b) Model Architecture

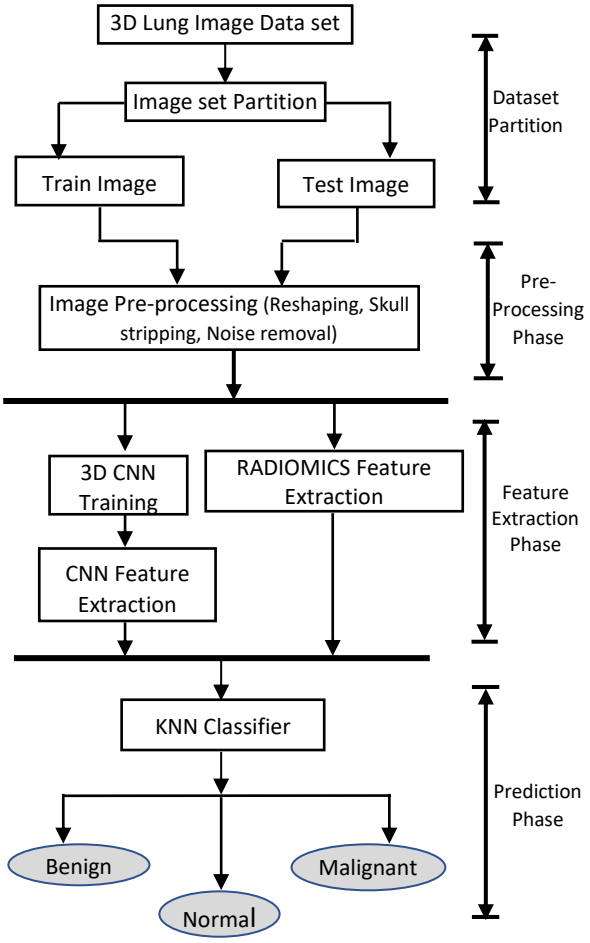

Fig 1: Proposed Architecture

c) Pre-Processing

First, the acquired images undergo a pre-processing step. For a flawless input image, the first things to do in the pre-processing stage are to resize the image and remove noise using Median filters.

Normalization

This process adjusts the range of pixel intensity values in equation (1).

$$I' = \frac{I - \min(I)}{\max(I) - \min(I)} \quad (1)$$

Where I is the original pixel intensity, $\min(I)$ and $\max(I)$ are the minimum and maximum intensities in the image, and I' is the normalized intensity.

Histogram Equalization

This technique improves the contrast of the image in equation (2).

$$I' = cdf(I) \times (L - 1) \quad (2)$$

Where $cdf(I)$ is the cumulative distribution function of the pixel intensities, and L is the number of possible intensity levels.

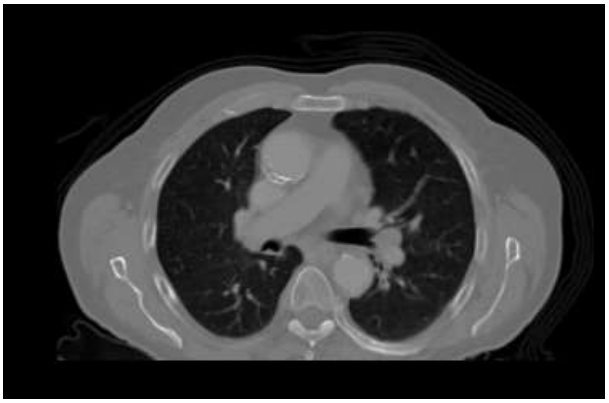

Fig 2: Input 3d CT Image

Reshaping Images

Figure 3 illustrates, by "image resizing," we mean the process of enlarging images. Reducing the number of pixels in a image by reshaping has various benefits, such as shortening the training period of a neural network's learning process and making the model simpler.

To aid with image scaling, OpenCV-Python offers many interpolation techniques. That may use the scaling routines provided by scipy and nd image, which work with n-dimensional NumPy arrays. The reshaped matrix measuring is 120*120*77.

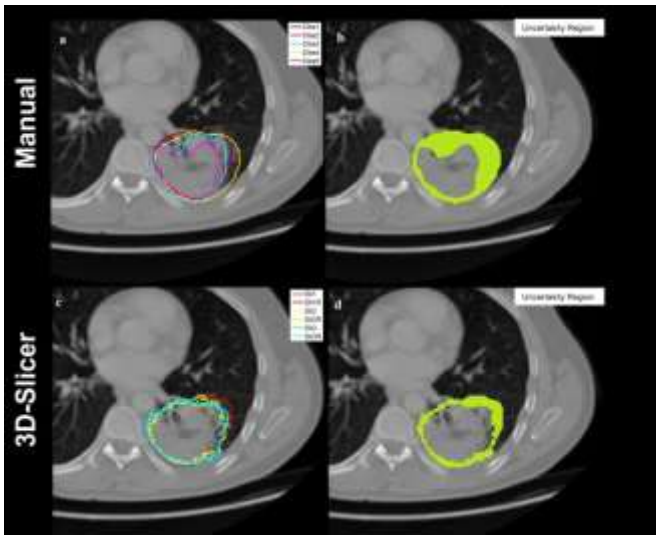

Fig 3: Reshaped Images in 3D Slicer

Noise Removal

Figure 4 and 5 illustrates, any interference with the image signal, whether natural or artificial, is considered noise. Here, we compare the input image and show the outcomes of median filtering approaches. Most of the noise in an image, or the consistent amount of noise in the shadows, is Gaussian noise. The characteristic feature of Gaussian noise is the alteration to the original value of each image pixel. A typical distribution of noise is shown by a histogram in Figure 6, which plots the degree of deformation of a pixel value versus the amount with which it occurs.

Median Filtering

This reduces noise while preserving edges in equation (3).

$$I'(x, y) = median\{I(x + k, y + l) | (k, l) \in W\} \quad (3)$$

Where $I(x, y)$ is the original image, $I'(x, y)$ is the filtered image, and W is the window centered around (x, y).

Edge Detection

This detects edges in the image in equation (4).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (4)$$

Where G_x and G_y are the gradients in the x and y directions.

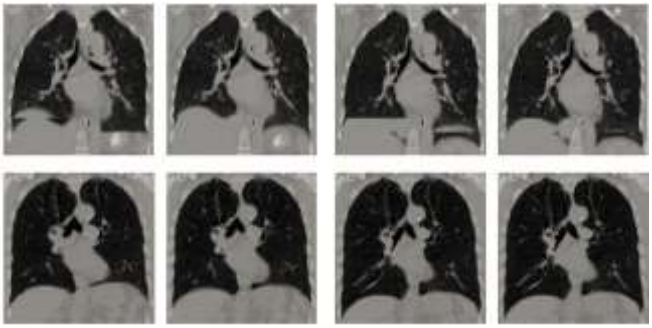

Fig 4: Noise Image

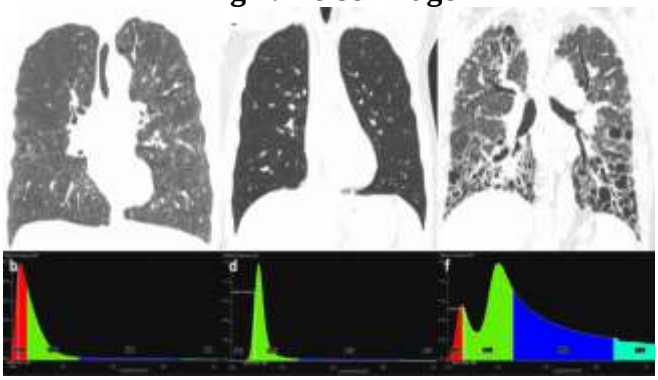

Fig 5: Noise Removal Image

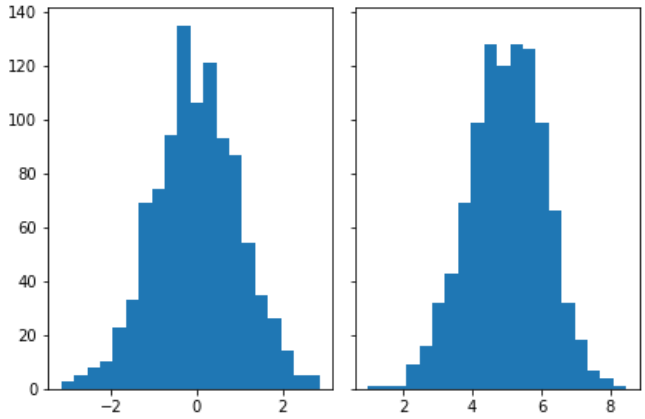

Fig 6: Histogram of Original and Noise image

d) Feature Extraction Using 3D-CNN

3D-CNN Architecture used

Feature extraction is carried out via the convolution + pooling layers of the 3D-CNN model. One output layer with four output nodes for classifying each of the four tasks, two fully-connected layers, and three 3D convolution (Conv) layers make up our 3D-CNN. The first convolution layer has eight filters with a 7×7×7 kernel size, the second layer has sixteen filters with a 5×5×5 kernel size, and the third layer has th irty-two filters with a 3×3×3 kernel size.

To illustrate, taking into account the 3D volume inputs of 240*240*155, the output pattern dimensions a t the first Conv layer were 38*64*6*160, with eight of these dimensions distributed across five filters (i. e., channels); at the second Conv layer, the dimensions were 17*30*2*320, with sixteen of these dimens ions distributed across sixteen channels, caused by a stride of two following the Conv operation. Then, t

he fully-connected layer was fed a one-dimensional vector with 81, 67, 402 elements, which had been tr ansformed from the Conv2 layer's output shown in Figure 7 and 8.

The convolution operation in a 3D-CNN for a given input volume V and a 3D kernel K is given by equation (5):

$$O(x,y,z) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\sum_{k=0}^{L-1} V(x+i,y+j,z+k) \times K(i,j,k) \quad (5)$$

where:
$O(x,y,z)$ is the output feature map at position (x,y,z)
$V(x+i,y+j,z+k)$ is the input volume value at position $(x+i,y+j,z+k)$
$K(i,j,k)$ is the kernel value at position (i,j,k)
M,N,L are the dimensions of the kernel (depth, height, width)

Feature Extraction

Convolution: Apply the 3D convolution operation across the input volume to produce feature maps in equation (6).

$$O^l = \sigma\left(\sum_{m=1}^{M} V_m^{l-1} * K_m^l + b^l\right) \quad (6)$$

O^l is the output feature map of layer l
V_m^{l-1} is the m-th input volume to the l-th layer
K_m^l is the m-th kernel for the l-th layer
$*$ denotes the 3D convolution operation.
b^l is the bias term for the l-th layer
σ is ReLU the activation function
Activation: Apply a non-linear activation function ReLU to introduce non-linearity into the model in equation (7).

$$\sigma(x) = \max(0,x) \quad (7)$$

Pooling: Apply 3D max pooling to reduce the spatial dimensions of the feature maps while retaining important features in equation (8).

$$P(x,y,z) = max_{i=0}^{p-1} max_{j=0}^{q-1} max_{k=0}^{r-1} O(x+i,y+j,z+k) \quad (8)$$

Where p,q,r are the dimensions of the pooling window.

Training of 3D CNN Model:
Our 3D-CNN was trained using the following parameters: an activation function for rectified linear units (ReLU) for every node in the Conv and Fully Connected layers, a loss function at the output layer, a stochastic gradient descent algorithm with an initial rate of learning of 10−3 (without momentum) and annealing shortly after 20 epochs with an initial minimum learning rate of 10−6, a mini-batch size of 32, and a dropout probability of 0.5 in the third 3D Conv layer to prevent overfitting. The 3DCNN is further activated using the RELU function shown in model history for Figure 9. To assess the efficacy, the leave-one-subject-out cross-validation method was used.

Fig 7: Labelled Images using conv2

For every set of 12 individuals, we utilized 11 to train the 3D-CNN and 1 to test it. After each set of 12 subjects, we repeated the procedure. The input feature maps learnt by each convolutional layer were the outputs of those layers shown in Table 1.

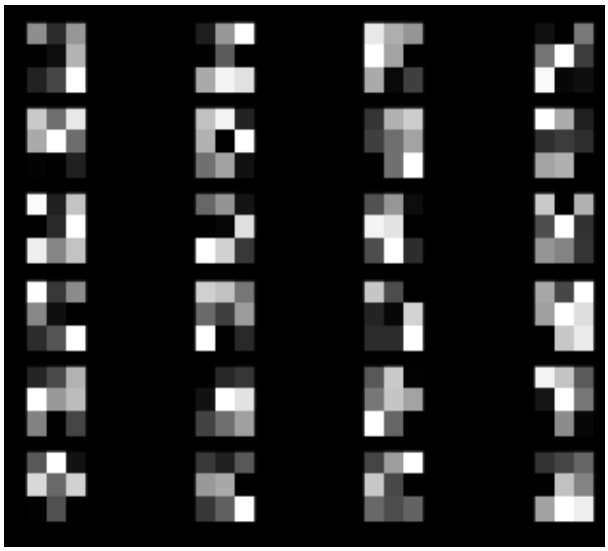

Fig 8: Labelled Images using conv3

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv3d_1 (Conv3D)	(None, 38, 64, 6, 160)	3040
batch_normalization_1 (Batch	(None, 38, 64, 6, 160)	640
max_pooling3d_1 (MaxPooling3	(None, 19, 32, 3, 160)	0
conv3d_2 (Conv3D)	(None, 17, 30, 2, 320)	921920
batch_normalization_2 (Batch	(None, 17, 30, 2, 320)	1280
max_pooling3d_2 (MaxPooling3	(None, 8, 15, 2, 320)	0
flatten_1 (Flatten)	(None, 76800)	0
dense_1 (Dense)	(None, 1024)	78644224
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 2048)	2099200
dropout_2 (Dropout)	(None, 2048)	0
dense_3 (Dense)	(None, 2)	4098

Total params: 81,674,402
Trainable params: 81,673,442
Non-trainable params: 960

Figure 9: Model History

Table 1: EPOCH and Accuracy Parts

Epoch 1/20
307/307 [==============================] - 200s 652ms/step - loss: 1.8425 - accuracy: 0.4984
Epoch 2/20
307/307 [==============================] - 196s 638ms/step - loss: 1.6016 - accuracy: 0.5179
Epoch 3/20
307/307 [==============================] - 201s 655ms/step - loss: 1.0286 - accuracy: 0.5798
Epoch 4/20
307/307 [==============================] - 202s 659ms/step - loss: 1.2259 - accuracy: 0.5244
Epoch 5/20
307/307 [==============================] - 197s 643ms/step - loss: 0.9684 - accuracy: 0.6026
Epoch 6/20
307/307 [==============================] - 199s 649ms/step - loss: 0.8191 - accuracy: 0.6580
Epoch 7/20
307/307 [==============================] - 202s 658ms/step - loss: 0.8346 - accuracy: 0.6547
Epoch 8/20
307/307 [==============================] - 197s 642ms/step - loss: 0.8130 - accuracy: 0.6743

Epoch 9/20
307/307 [==============================] - 203s 660ms/step - loss: 0.6213 - accuracy: 0.7134
Epoch 10/20
307/307 [==============================] - 199s 649ms/step - loss: 0.7606 - accuracy: 0.6906
Epoch 11/20
307/307 [==============================] - 214s 697ms/step - loss: 0.6486 - accuracy: 0.6710
Epoch 12/20
307/307 [==============================] - 200s 652ms/step - loss: 0.6098 - accuracy: 0.7101
Epoch 13/20
307/307 [==============================] - 206s 672ms/step - loss: 0.5133 - accuracy: 0.7720
Epoch 14/20
307/307 [==============================] - 198s 646ms/step - loss: 0.4423 - accuracy: 0.7590
Epoch 15/20
307/307 [==============================] - 209s 679ms/step - loss: 0.3630 - accuracy: 0.8339
Epoch 16/20
307/307 [==============================] - 193s 630ms/step - loss: 0.4013 - accuracy: 0.8306
Epoch 17/20
307/307 [==============================] - 193s 628ms/step - loss: 0.3970 - accuracy: 0.8371
Epoch 18/20
307/307 [==============================] - 196s 637ms/step - loss: 0.3184 - accuracy: 0.8599
Epoch 19/20
307/307 [==============================] - 194s 632ms/step - loss: 0.3000 - accuracy: 0.8762
Epoch 20/20
307/307 [==============================] - 204s 664ms/step - loss: 0.2926 - accuracy: 0.8893

e) GLCM Feature extraction using PyRadiomics

The field of radiomics has the potential to revolutionize a wide range of imaging modalities, from radiography and ultrasound to CT scans. It may be used to improve the accuracy of diagnosis, prognosis evaluation, and therapeutic response prediction, especially when combined with genetic, biochemical, and clinical data. Researchers in the subject of radiomics are now concentrating on finding the best selective feature extraction techniques, which differ depending on the imaging modality and the disease being examined.

Volume, form, surface, density, intensity, texture, position, and relationships to neighboring tissues are all features.

When describing areas of interest in radiology, semantic characteristics represent the ones that are most often employed. Those characteristics that try to quantify mathematical descriptors of lesion heterogeneity are called agnostic features.

Contrast

Measures the intensity contrast between a pixel and its neighbor over the whole image in equation (9).

$$\boldsymbol{Contrast} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(i-j)^2\,P(i,j) \qquad (9)$$

Where $P(i,j)$ is the normalized GLCM value at position (i,j) and N is the number of gray levels.

Correlation

Measures how correlated a pixel is to its neighbor in equation (10).

$$\boldsymbol{Correlation} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\frac{(i-\mu_i)\big(j-\mu_j\big)P(i,j)}{\sigma_i\,\sigma_j} \qquad (10)$$

Where μ_i and μ_j are the means, and σ_i and σ_j are the standard deviations of the marginal distributions of $P(i,j)$

Energy

Angular Second Moment measures the sum of squared elements in the GLCM in equation (11).

$$Energy = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} P(i,j)^2 \quad (11)$$

Homogeneity

Inverse Difference Moment measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal in equation (12).

$$\boldsymbol{Homogeneity} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \frac{P(i,j)}{1+|i-j|} \quad (12)$$

The study made use of the suggested default settings and radiomic characteristics provided in accordance with the PyRadiomics Python package, version 3.8.0. Eleven first-order features, seventeen form features, and fifty-six texture features were extracted. A grand total of 287 image features were computed, including first-order and texture features, using eight wavelet decompositions. Four decompositions were used for two-dimensional features shown in Table 2.

Table 2: GLCM Features

Features	Values
Energy	2918821481.0
Entropy	4.920992838328338
Kurtosis	2.1807729393860265
Mean	825.235436306502
Skewness	0.27565085908587594
Contrast	74.04325876559685
Correlation	0.39322090748573196
Strength	0.9828367173152485
Volume	2
Shape	0.5621171627174117
Surface	6438.821603779402
Density	54.27945170740796
Intensity	0.27565085908587594
Texture	17.33
Correlation	0.39322090748573196

F) CNN AND KNN CLASSIFIER

Figure 10 illustrates, a multivariate feature space contains the training examples, which are vectors labeled with a class. Simply storing the training samples' class labels and feature vectors is the training phase of the algorithm. Classification involves taking an unlabeled vector and labeling it with the most common label among the k training examples closest to it, The Euclidean distance is a popular distance measure for continuous variables. When working with the expression of genes microarray data, for instance, it has been used as a metric with Pearson and Spearman correlation coefficients. Learning the distance metric using specialized methods, such Large Margin Nearest Neighbor or Neighbor Components Analysis, may sometimes greatly enhance the classification accuracy of k-NN. Importing KNN Classifier from sklearn imports train_test_split and preprocessing, as well as GLCM and CNN inputs via KNN Classifier, is utilized in my python code for the suggested task. Convolutional (conv3d_1, conv3d_2), subsampling (maxpool1, maxpool2), and output (dense_5) layers make up the GLCM-CNN architecture.

The retrieval area of the three-dimensional feature maps. The problem of over-or under-segmentation can arise, however, since the pooling mechanism will fail to capture all feature information as the network becomes deeper. To fix the 3D feature maps' missing spatial information, we revamp the 3D-CNN model by adding a layer cohesion architecture. The up-sampling technique is carried done via bilinear interpolation. To interpolate bivariate functions, bilinear interpolation extends a two-dimensional rectangular grid. To address the issue of misclassified voxels, post-processing is used. The voxels that have a tiny linked area are considered to be tumors, and a threshold approach will remove them (i.e., the

false tumor region) if their connected area is smaller than a certain threshold. In an adaptive setting, the specified threshold is initially set at 10% of the maximal connected area.

CNN Working

The fundamental building block of a CNN. It involves sliding a filter (kernel) over the input to produce a feature map in equation (13).

$$O_{i,j,k} = \sum_{m}\sum_{n}\sum_{c}\left(I_{i+m,j+n,c} \times K_{m,n,c,k}\right) + b_k \quad (13)$$

where:

$\mathbf{O_{i,j,k}}$ is the output feature map at position $(\mathbf{i}, \mathbf{j})$ for the k-th filter.

$I_{i+m,j+n,c}$ is the input value at position $(i + m, j + n)$ for the c-th channel.

$K_{m,n,c,k}$ is the filter value at position $(m, \mathbf{n})$ for the c-th channel and k-th filter.

b_k is the bias term for the k-th filter.

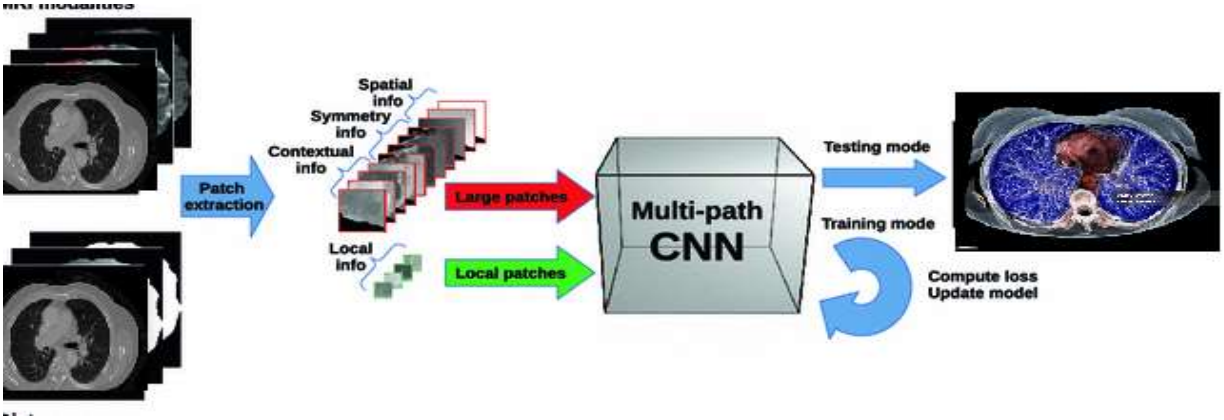

Fig 10: CNN Architecture

Activation Function: Introduces non-linearity. Commonly used activation functions include ReLU (Rectified Linear Unit) in equation (14).

$$\mathbf{ReLU}(\mathbf{x}) = \max(0, x) \quad (14)$$

Pooling Operation: Reduces the spatial dimensions of the feature maps. Common pooling operations include max pooling in equation (15).

$$\mathbf{P_{i,j,k}} = \max_{m,n}(O_{i+m,j+n,k}) \quad (15)$$

Where $\mathbf{P_{i,j,k}}$ is the pooled feature map.

Fully Connected Layer: Connects every neuron in one layer to every neuron in the next layer in equation (16).

$$\mathbf{y} = \text{softmax}(Wx + b) \quad (16)$$

where:

W is the weight matrix.

x is the input vector.

b is the bias vector.

$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$ converts the output into probabilities.

KNN Working

Distance Metric: KNN uses a distance metric to find the nearest neighbors. The Euclidean distance is commonly used in equation (17).

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n}(x_j - x_{i,j})^2} \quad (17)$$

where:

$d(x, x_i)$ is the distance between the query point x and the i-th training point x_i.

$x_j$and $x_{i,j}$ are the j-th features of the query point and the i-th training point, respectively.

Classification: Assigns the class based on the majority vote of the k-nearest neighbors in equation (18).

$$\hat{y} = \arg\max_{c} \sum_{i=1}^{k} 1(y_i = c) \quad (18)$$

where:

$\hat{y}$ is the predicted class.

y_i is the class label of the i-th nearest neighbor.

c represents the class.

To produce a forecast, this is all that is required. To determine the likelihood of a pattern being part of each output class, we can simply look at the output values. The result could be better used as a clear class prediction again. The most likely class value is chosen in this way. "Arg max" is another name for this method. This approach is implemented by a function called predict () methods. It takes the output from the network and returns the index with the highest probability. The conversion of class values to integers, beginning at 0, is assumed.

IV. IMPLEMENTATION

The suggested approach is executed in a Python environment with the help of the Keras and tensorflow library. We use Jupyter Notebook and Python script. Training In each 3D-CT images are research using the 307 samples were randomly selected to create the data. Data is made impartial by duplicating it and then conducting a rotation and flipping process. Table 3 shows that a total of 307 data samples were used for the training dataset. We trained the model over 20 epochs using 32-batch size. When working with Adam optimizer, the form of the binary cross-entropy function for loss is used. Training the model takes around ten minutes. The system's performance and settings determine this timeframe. It is possible to enhance the existing model by adjusting its parameters; it has a training loss of 0.5% and a testing loss of 1.2%.

Table 3: Implementation Work

Image Type	Data Samples
Number of All Images	307
Number of Covid affect Image	156 images
Number of Covid mild affect Image	126 images
Number of Covid no-affect Image	25 images
Tool used	Anaconda jupyter notebook 4

V. RESULT AND DISCUSSION

The outcomes of the CNN that was developed are presented in Table 4 and Figure 12 through the use of confusion matrices. Non-white columns in confusion matrices correspond to actual classes, whereas non-white rows indicate network output classes (Figure 12). The numbers representing the percentages of accurately classified images are displayed along the diagonal shown in Figure 10. The final column is indicative of the specificity, while the final row represents the sensitivity. The conditional accuracy is displayed in the lower-right corner. The quantity of images is denoted by the upper value in the non-white frames, while the percentage of the entire class database that is utilized in the training or test set is represented by the lower value. To mitigate the issue of class imbalance in the database, we have

additionally presented the average of the mean precision, recall, and F1-score in Tables 5 and 6. It is subsequently classified utilizing KNN, following which the error-avoiding algorithm is executed error rate shown in Figure 11. This will aid in the differentiation of normal and affect. The compact kernel size facilitates the formation of a deep network, which enables the extraction of more intricate feature details. The multistage approach ensures that the geographical synchronization is maintained, allowing for testing of the model on any database. Overa all comparison performance in 2D and 3D Classification represents Table 7.

Table 4: Dataset Modeling results

Images	Actual	Predicted by proposed Model	Accuracy
Affect	126	122	0.97
Mild	156	153	0.98
No-Affect	25	25	1.00
		Avg Accuracy	0.98

Table 5: KNN results

Precision	Recall	F1-Score	Accuracy
0.94	0.98	0.96	0.95
0.97	0.89	0.93	0.95
0.95	0.94	0.94	0.95

Table 6: CNN results

Precision	Recall	F1-Score	Accuracy
0.96	0.98	0.97	0.96
0.97	0.94	0.95	0.96
0.97	0.96	0.96	0.96

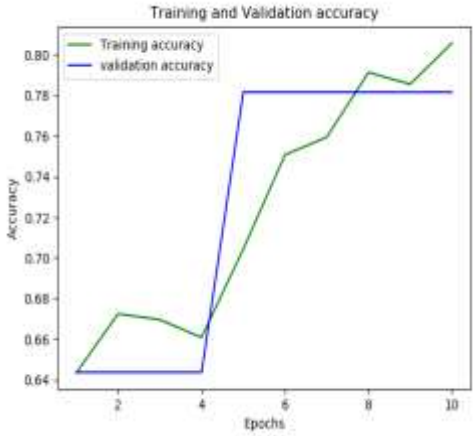

Fig 10: Train and Test Score

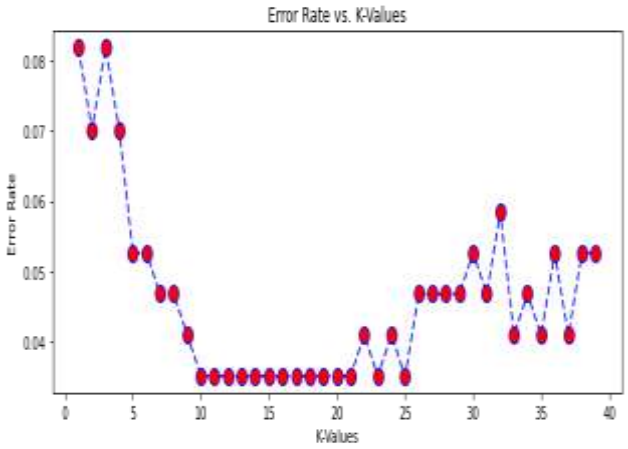

Fig 11: Error Rate

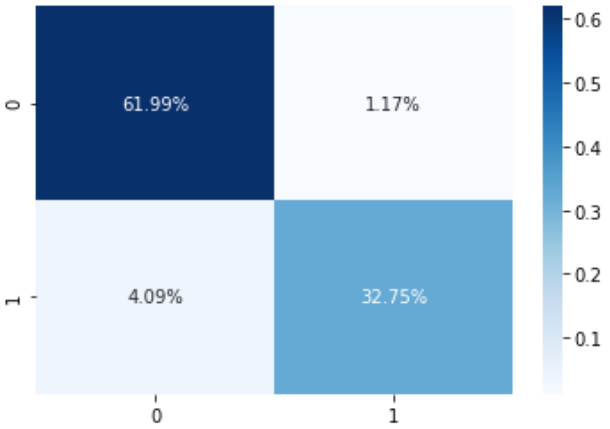

Fig 12: Percentage based Confusion Matrix

Table 7: Comparison Between 2D and 3D

Algorithm and Dimensions	SVM	Proposed Work
2D	0.90	0.96
3D	0.94	0.98

VI. CONCLUSION

Employing GLCM features, the altered original CNN implements the KNN classifier. When the input layer is modified with the GLCM image, the accuracy of cattle race recognition can be enhanced in comparison to the original CNN. The strength of the adjustment is that the statistical data in the layer that is input now contains GLCM features that are superior to those of the original image. Furthermore, the segmentation procedure is unnecessary for the classification of cattle races using this method. Three properties of GLCM features were utilized in this study: contrast, energy, and homogeneity. The accuracy of the original CNN can be enhanced by contrast and energy images, but this is the exact opposite of the homogeneity image, which has a lower accuracy. Saliency maps can reduce the time required to compute and obtain accurate GLCM feature extractions, which is speedier than if they were not utilized.

CNN and KNN each employ a unique approach to learning. In most cases, their fusion would result in increased classification rates. The potential solution to complex problems may lie in the fusion of CNN's deep learning capability to manage transformations and KNN's superior margin separation. In certain circumstances, where one classifier exhibits superiority over the other, fusion would still guarantee a satisfactory final classification outcome notwithstanding the absence of inaccuracy in one of the

classification algorithms due to the inherent characteristics of the problem.

Declarations

Conflict of Interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Competing Interests

Not Applicable.

Funding Information

Not Applicable.

Author contribution

All authors contributed equally to this work.

Data Availability Statement

The data used to support the findings of this study have been openly available. Dataset Link: https://www.cancerimagingarchive.net/collection/lung-pet-ct-dx/

Research Involving Human and /or Animals

Not Applicable.

Informed Consent

Not Applicable.

REFERENCES

1. Viswanath, K., M. Bekalu, D. Dhawan, R. Pinnamaneni, J. Lang, and R. McLoud. "Individual and Social Determinants of COVID-19 Vaccine Uptake." BMC Public Health 21, no. 818 (2021). https://doi.org/10.1186/s12889-021-10825-2.
2. Biswas, N., T. Mustapha, J. Khubchandani, and J.H. Price. "The Nature and Extent of COVID-19 Vaccination Hesitancy in Healthcare Workers." Journal of Community Health 46 (2021): 1244–1251. https://doi.org/10.1007/s10900-021-00984-3.
3. Holzmann-Littig, C., M.C. Braunisch, P. Kranke, M. Popp, C. Seeber, F. Fichtner, B. Littig, J. Carbajo-Lozoya, C. Allwang, T. Frank, et al. "COVID-19 Vaccination Acceptance and Hesitancy Among Healthcare Workers in Germany." Vaccines 9, no. 777 (2021). https://doi.org/10.3390/vaccines9070777.
4. Gilboa, M., I. Tal, E.G. Levin, S. Segal, A. Belkin, T. Zilberman-Daniels, A. Biber, C. Rubin, G. Rahav, and G. Regev-Yochay. "Coronavirus Disease 2019 (COVID-19) Vaccination Uptake Among Healthcare Workers." Infection Control & Hospital Epidemiology 43 (2021): 1433–1438. https://doi.org/10.1017/ice.2021.322.
5. Külper-Schiek, W., V. Piechotta, A. Pilic, M. Batke, L.S. Dreveton, B. Geurts, et al. "Facing the Omicron Variant-How Well Do Vaccines Protect Against Mild and Severe COVID-19? Third Interim Analysis of a Living Systematic Review." Frontiers in Immunology 13 (2022): 940562. https://doi.org/10.3389/fimmu.2022.940562.
6. Chenchula, S., P. Karunakaran, S. Sharma, and M. Chavan. "Current Evidence on Efficacy of COVID-19 Booster Dose Vaccination Against the Omicron Variant: A Systematic Review." Journal of Medical Virology 94, no. 7 (2022): 2969–2976. https://doi.org/10.1002/jmv.27697.
7. Le, T.T.B., T. Vasanthakumaran, H.N. Thi Hien, I.C. Hung, M.N. Luu, Z.A. Khan, et al. "SARS-CoV-2 Omicron and Its Current Known Unknowns: A Narrative Review." Reviews in Medical Virology (2022): e2398. https://doi.org/10.1002/rmv.2398.
8. Ai, J., H. Zhang, Y. Zhang, K. Lin, Y. Zhang, J. Wu, et al. "Omicron Variant Showed Lower Neutralizing Sensitivity Than Other SARS-CoV-2 Variants to Immune Sera Elicited by Vaccines After Boost." Emerging Microbes & Infections 11, no. 1 (2022): 337-343. https://doi.org/10.1080/22221751.2021.2022440.
9. Harder, T., J. Koch, S. Vygen-Bonnet, W. Külper-Schiek, A. Pilic, S. Reda, et al. "Efficacy and Effectiveness of COVID-19 Vaccines Against SARS-CoV-2 Infection: Interim Results of a Living Systematic Review, 1 January to 14 May 2021." Eurosurveillance 26, no. 28 (2021): 2100563. https://doi.org/10.2807/1560-7917.ES.2021.26.28.2100563.

10. Khan, N.A., H. Al-Thani, and A. El-Menyar. "The Emergence of New SARS-CoV-2 Variant (Omicron) and Increasing Calls for COVID-19 Vaccine Boosters-The Debate Continues." Travel Medicine and Infectious Disease 45 (2022): 102246. https://doi.org/10.1016/j.tmaid.2021.102246.
11. Magen, O., J.G. Waxman, M. Makov-Assif, R. Vered, D. Dicker, M.A. Hernán, et al. "Fourth Dose of BNT162b2 mRNA Covid-19 Vaccine in a Nationwide Setting." New England Journal of Medicine 386, no. 17 (2022): 1603-1614. https://doi.org/10.1056/NEJMoa2115624.
12. Saudi MOH. "Coronavirus Dashboard." Accessed May 29, 2023. https://covid19.moh.gov.sa/.
13. Kane, J.C., M.A. Elafros, S.M. Murray, et al. "A Scoping Review of Health-Related Stigma Outcomes for High-Burden Diseases in Low-and Middle-Income Countries." BMC Medicine 17, no. 1 (2019): 1–40. https://doi.org/10.1186/s12916-019-1250-8.
14. Dar, S.A., S.Q. Khurshid, Z.A. Wani, et al. "Stigma in Coronavirus Disease-19 Survivors in Kashmir, India: A Cross-Sectional Exploratory Study." PLoS One 15, no. 11 (2020): e0240152. https://doi.org/10.1371/journal.pone.0240152.
15. Shahnawaz, M., W. Nabi, S. Nabi, et al. "Relationship Between Intolerance of Uncertainty and Symptom Severity in Covid-19 Patients: The Mediating Role of Illness Perception and Covid-19 Fear." Current Psychology (2022): 1–8. https://doi.org/10.1007/s12144-022-03577-y.
16. Lin, J.L., and Y.K. Wang. "Lessons from the Stigma of COVID-19 Survivors: A Marxist Criticism Appraisal." Frontiers in Public Health 11 (2023): 1156240. https://doi.org/10.3389/fpubh.2023.1156240.