Medical data analysis in edge computing environment

•Mayukh Roy •Shubhadeep Maity •Hrithik Rudra

Abstract

Pulmonary Embolism (PE) is a critical condition where a blood clot obstructs a pulmonary artery, leading to impaired blood flow to the lungs. Computed Tomography Pulmonary Angiography (CTPA) is the gold standard for diagnosing PE and has proven to reduce mortality rates. However, accurately identifying emboli within CTPA images remains challenging. This study proposes an innovative approach employing VGG16 and XGBOOST models to categorize CTPA images as either 'PE' (images with emboli) or 'No PE' (images without emboli). We utilize a Kaggle dataset containing over 1000 images categorized accordingly. To address the need for efficient and timely classification, we integrate edge computing into our methodology. Edge computing involves processing data in proximity to its origin, minimizing latency and enabling real-time decision-making. By deploying our VGG16 and XGBOOST models on edge devices such as local servers or within healthcare facilities, we enhance the speed and reliability of CTPA image classification. This advancement facilitates prompt diagnosis by doctors, enabling timely medical interventions and potentially saving lives.

1. Introduction

EDGE COMPUTING

To implement this in edge computing, you would need to first optimize the model and reduce its size to be able to run on edge devices with limited processing power and memory. This can be done through techniques like model quantization, pruning, and compression. Once the model is optimized, it can be deployed on the edge device using a framework like TensorFlow Lite or ONNX Runtime. These frameworks allow you to convert the model into a format that can be executed on the edge device's hardware and provide APIs for loading and running the model. You would also need to optimize the input processing pipeline to run efficiently on the edge device. This could involve using optimized image processing libraries and techniques like batching and prefetching to minimize the processing time and memory usage. Finally, you would need to integrate the model into your edge application and design a user interface to display the results. This could involve developing a custom application or using an existing platform like OpenCV or TensorFlow.js to build the user interface.

VISUAL GEOMETRIC GROUP-16 (VGG16)

Visual Geometric Group-16 (VGG16) is a convolutional neural network (CNN) architecture that was developed by the Visual Geometry Group (VGG) at the University of Oxford. It is a deep neural network that consists of 16 layers and was introduced in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" by Karen Simonyan and Andrew Zisserman in 2014. VGG16 achieved excellent results on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014 and has since become a popular model for image classification tasks. Here is a breakdown of the VGG16 architecture:

1. Input layer: The input layer takes in the input image of size 224x224x3 (RGB image). Convolutional layers: VGG16 has 13 convolutional layers, where each layer uses a 3x3 filter with a stride of 1 and a padding of 1 to maintain the spatial resolution of the input. The number of filters in each layer increases from 64 in the first layer to 512 in the last layer.
2. Max pooling layers: After each set of convolutional layers, there is a max pooling layer that reduces the spatial dimensions of the feature maps by a factor of 2.
3. Fully connected layers: The output of the last max pooling layer is flattened and passed through three fully connected layers with 4096, 4096, and 1000 neurons respectively. The first two fully connected layers have a dropout rate of 0.5 to prevent overfitting.
4. Softmax layer: The output of the last fully connected layer is passed through a softmax layer to get the class probabilities for each of the 1000 categories in the ImageNet dataset. The total number of trainable parameters in VGG16 is around 138 million, making it a very large and computationally expensive model. However,

it has shown excellent performance on image classification tasks and is widely used as a benchmark for other CNN architectures.features are then flattened and fed into the XGBOOST for classification.

XGBOOST (EXTREME GRADIENT BOOSTING)

XGBoost (Extreme Gradient Boosting) is a popular machine learning algorithm for supervised learning tasks, particularly for classification and regression problems. It is a gradient boosting framework that uses a set of decision tree-based models for ensemble learning. Here are some key features of XGBoost:

1. Gradient boosting: XGBoost is based on the gradient boosting framework, which means that it builds an ensemble of decision trees incrementally by optimizing a loss function using gradient descent. Each decision tree in the ensemble is built to minimize the residual error of the previous tree.
2. Regularization: XGBoost uses two forms of regularization to prevent overfitting: L1 regularization (lasso regression) and L2 regularization (ridge regression). Regularization is controlled by hyperparameters such as lambda (L2 regularization) and alpha (L1 regularization).
3. Handling missing values: XGBoost has built-in support for handling missing values in the input data. It learns the best direction to move in order to impute missing values during training.
4. Feature importance: XGBoost provides a feature importance score that can be used to understand which features are the most important in the model. This is based on the number of times a feature is used to split the data across all trees in the ensemble.
5. Parallel processing: XGBoost is designed to take advantage of parallel processing to speed up training on large datasets. It uses multiple cores on a single machine and can also be run in distributed mode across multiple machines.
6. Early stopping: XGBoost allows for early stopping, which means that training can be stopped once the validation error stops improving. This helps to prevent overfitting and speeds up training.

XGBoost has become a popular algorithm in machine learning competitions and is widely used in industry for a variety of tasks, including predictive modeling, natural language processing, and image processing. It has a number of hyperparameters that can be tuned to optimize performance, but it generally requires less tuning than other models such as neural networks.

PULMONARY EMBOLISM AND ITS DETECTION IN CTPA IMAGES

Pulmonary embolism is a life-threatening condition where a blood clot blocks a pulmonary artery in the lungs, impairing blood flow and oxygenation. It can cause symptoms like sudden shortness of breath, chest pain, and rapid heartbeat. Prompt medical attention is crucial for diagnosis and treatment.

Pulmonary embolism (PE) detection in CT pulmonary angiography (CTPA) images is an important application of medical imaging for diagnosing potentially life-threatening conditions. Here are some common methods used for detecting pulmonary embolism in CTPA images:

1. Visual examination by radiologists to identify filling defects or areas of decreased enhancement in the pulmonary arteries.
2. Slice-based evaluation, reviewing multiple image slices to identify abrupt vessel cutoffs or segmental perfusion defects.
3. Computer-aided detection (CAD) systems using algorithms and machine learning to automatically analyze images and highlight potential emboli.
4. Volumetric assessment using 3D volume-rendered images to comprehensively evaluate the pulmonary arteries and detect emboli in different orientations.

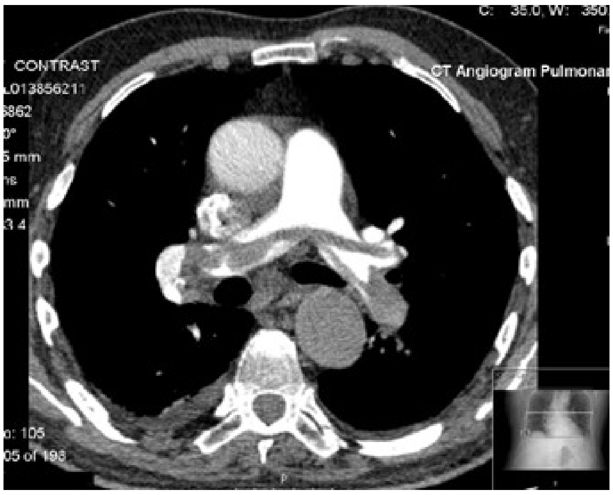


Fig 1.1 : Example of a CTPA, demonstrating a saddle embolus

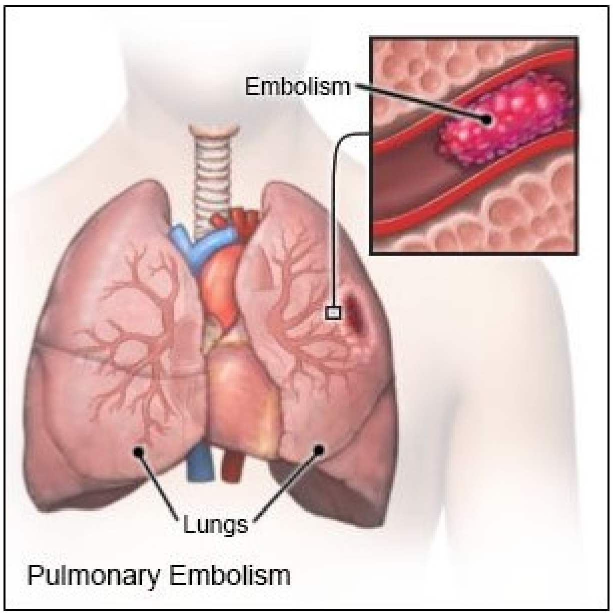


Fig 1.2 : Pulmonary Embolism diagram

2. Background Study/Related Work

A large number of studies have been conducted on the topic of Medical image classification, which are as follows with their results -

1. "A Two-Stage Convolutional Neural Network for Pulmonary Embolism Detection From CTPA Images"[5]

Methodology: The researchers used a two-stage approach, employing a ResNet-18 architecture, a popular type of convolutional neural network, to detect pulmonary embolism from computed tomography pulmonary angiography (CTPA) images.

Accuracy: The model achieved an accuracy of 75.4%, indicating its ability to correctly identify cases of pulmonary embolism.

2. "On edge deep learning implementation: approach to achieve 5G"[8]

Methodology: The researchers were discussing the state of art within mobile edge computing with deep learning to server low-latency, real time application by providing application specific resource allocation.

Results: The experimental results have indicated significant amount of improvement in respond time while executing in low latency.

3. "U-net: Convolutional networks for biomedical image segmentation"[3]

Methodology: The U-net architecture, which is a type of convolutional neural network, was employed for biomedical image segmentation. This network architecture is specifically designed for accurate segmentation of images.

Accuracy: The model achieved an accuracy of 77.5%, highlighting its capability to accurately segment biomedical images for various applications.

4. "Artificial Intelligence Algorithm with SVM Classification using Dermascopic Images for Melanoma Diagnosis"[6]

Methodology: The study utilized an artificial intelligence algorithm in combination with Support Vector Machine (SVM) classification for melanoma diagnosis. SVM is a popular machine learning algorithm used for classification tasks.

Accuracy: The model demonstrated an accuracy of 70% in diagnosing melanoma cases using dermascopic images.

5. "Accurate Pulmonary Nodule Detection in Computed Tomography Images Using Deep Convolutional Neural Networks"[1]

Methodology: A computer-aided detection (CAD) system employing deep convolutional neural networks (CNNs) was used for accurate detection of pulmonary nodules in computed tomography (CT) images.

Accuracy: The model achieved an accuracy of 89%, indicating its effectiveness in accurately identifying pulmonary nodules in CT images.

6. "A novel method for pulmonary embolism detection in CTA images"[2]

Methodology: The researchers developed a novel computer-aided detection (CAD) system for identifying pulmonary embolism in computed tomography angiography (CTA) images.

Accuracy: The model demonstrated an impressive accuracy of 95.1%, indicating its high level of accuracy in detecting pulmonary embolism cases.

7. "Deep convolutional neural network based medical image classification for disease diagnosis"[4]

Methodology: The researchers employed CapsNet, a type of neural network architecture known for its ability to capture hierarchical image features, for medical image classification tasks related to disease diagnosis.

Accuracy: The model achieved an accuracy of 74%.

3. Methodology

3.1. PROPOSED METHODOLOGY

The methodology proposed aims to classify the given CTPA scans from the Dataset correctly into "PE" and "No PE" classes with minimal misclassifications.

Fig 3.1 illustrates the procedure step by step.

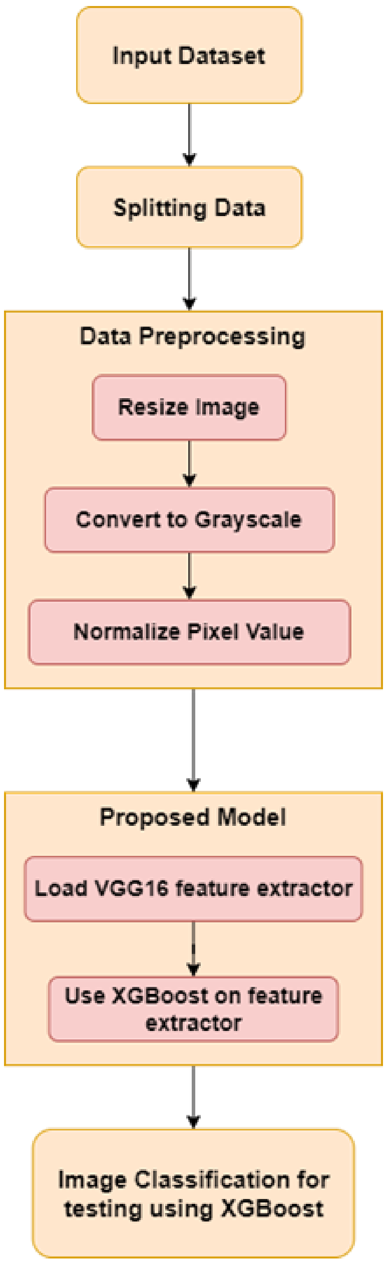


Fig 3.1 : Flowchart for the entire methodology

3.2. DATASET USED

Dataset used for our paper is The RSNA STR Pulmonary Embolism Dataset. The Society of Thoracic Radiology (STR) has teamed up with the Radiological Society of North America (RSNA) to use machine learning algorithm at its most in the diagnosis of PE.

We also have evaluated our method on the RSNA STR PE Detection Dataset that is available on Kaggle for the academic research and education.[7]

3.2.1. Files Description

The Dataset that we have used contains two subdirectories for negative and positive PE cases.
One subdirectory is the "Train" data which contains the metadata for all the images.
Another one is the "Test" data.

3.2.2. Data Used

Study Instance UID – the unique ID for each study in the dataset.
Series Instance UID – the unique ID for each series within the study.
SOP (Standard Operating Procedure) Instance UID – the unique ID for each image within the study.
"pe_present_on_image" shows the image-level and it also notes whether any form of PE is present on the image or not.
The images were downloaded from Kaggle in .jpg format and the train.csv file was used to segregate the images into two classes, namely 'PE' (The images that contain embolism) and 'No PE' (The images that did not contain any embolous) based on the "pe_present_on_image" column.

3.2.3. Creating Train and Test sets

After the division of images into two different classes we split the data into train and test data so that there are 832 images in the train set and 105 images in the test set. This is done so that our model will have enough data to learn from and provide better reliability and performance.

3.3. MODEL BUILDING

Image interpretation by computer programmes has become an important topic for Machine Learning when it comes to Research and Application. The digital image collecting, and storage technology has also shown rapid growth. Medical image classification is a challenging task in Deep Learning which tries to classify medical images into distinct categories to aid doctors and radiologists in diagnosing disease or doing additional study.

There are two steps for classifying medical images. The first step is taking a photograph and then extracting information from it, which is then used to classify the images and generate models in the second stage. Doctors/radiologists formerly used their professional experience to extract features from medical images to classify them into distinct classes, which was a tedious, monotonous, and time-consuming task. Based on previous research, studies of Medical Image Classification have a lot of merit. However, we are still unable to accomplish our goal, i.e., perfect mechanism for image classification. If the classification process is performed well, the data will aid medical doctors in diagnosing disorders that require further investigation.

3.4. MODEL EVALUATION METRICS

The classification report includes various metrics used for evaluation of our model, which are **Precision, Recall, F1 score, Accuracy and Specificity**.
The formulas for the above metrics are given as follows –

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

$$f1 - score = \frac{2 * Precision * Recall}{Precision + Recall}$$

$$Accuracy = \frac{TruePositives + TrueNegatives}{TruePositives + FalseNegatives + TrueNegatives + FalsePositives}$$

$$Specificity = \frac{TrueNegatives}{TrueNegatives + FalsePositives}$$

4. Results and Discussion

4.1. CODE USED

So for our project we implemented the whole coding part in JUPYTER NOTEBOOK, using the language PYTHON. The following show the coding implementation of some of the major functionalties of our proposed methodology -

4.1.1. VGG16 Implementation

In[15]: Vmodel = VGG16(weights='imagenet', include_top=False, input_shape=(SIZE, SIZE, 3))

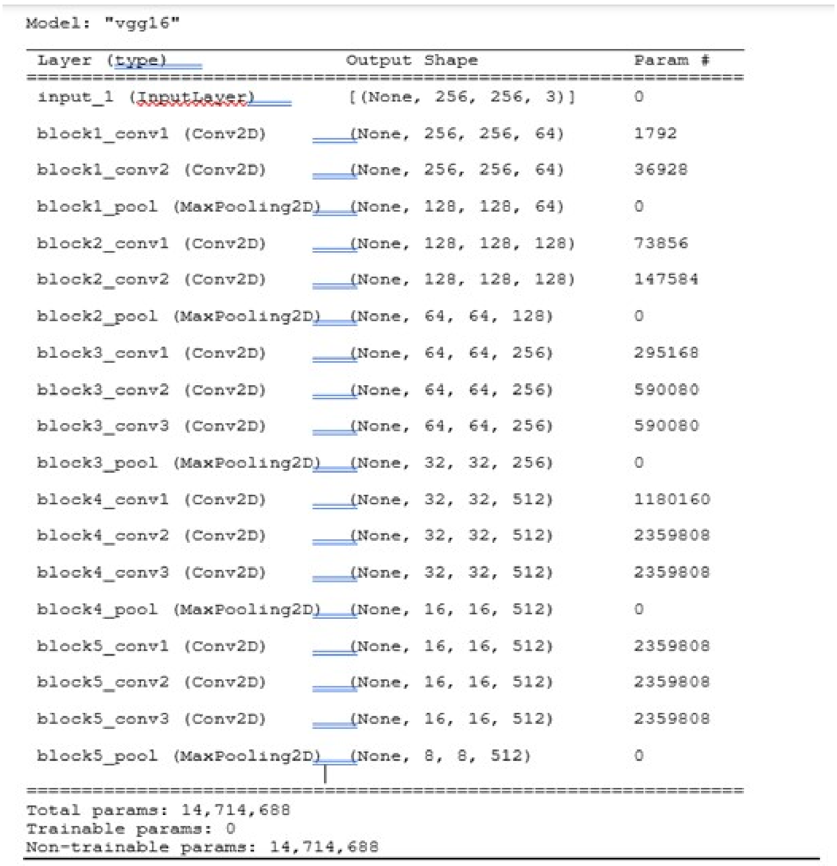
In [16] : # Make loaded layers as non-trainable. This is important as we want to work with pre-trained weights

for layer in Vmodel.layers:

    layer.trainable = False

In [17] : Vmodel.summary() #Trainable parameters will be 0

Out [17] :



```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

 block2_conv1 (Conv2D)       (None, 128, 128, 128)     73856

 block2_conv2 (Conv2D)       (None, 128, 128, 128)     147584

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0

 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160

 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0

 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
```

4.1.2. Feature extraction using VGG16

In [18] : #send train data for feature extraction

feature_extractor=Vmodel.predict(x_train)

Out [18] :

52/52 [==============================] - 792s 14s/step

In [19] : features = feature_extractor.reshape(feature_extractor.shape[0], -1)

In [20] : X_for_training = features

In [21] :

len(X_for_training[1])

X_for_training.shape

Out [21] :

(1664, 32768)

In [23] : # Send test data through same feature extractor process

X_test_feature = Vmodel.predict(x_test)

X_test_features = X_test_feature.reshape(X_test_feature.shape[0], -1)

Out [23] : 7/7 [==============================] - 90s 13s/step

4.1.3. XGBOOST implementation

In [25] :

import xgboost as xgb

model = xgb.XGBClassifier()

model.fit(X_for_training, y_train)

Out [25] :

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

4.2. RESULTS

Fig 4.1 represents the confusion matrix which describes the performance of our model on test data. The proposed methodology has performed well with very few misclassifications - 5 out of 105 unseen test data points.

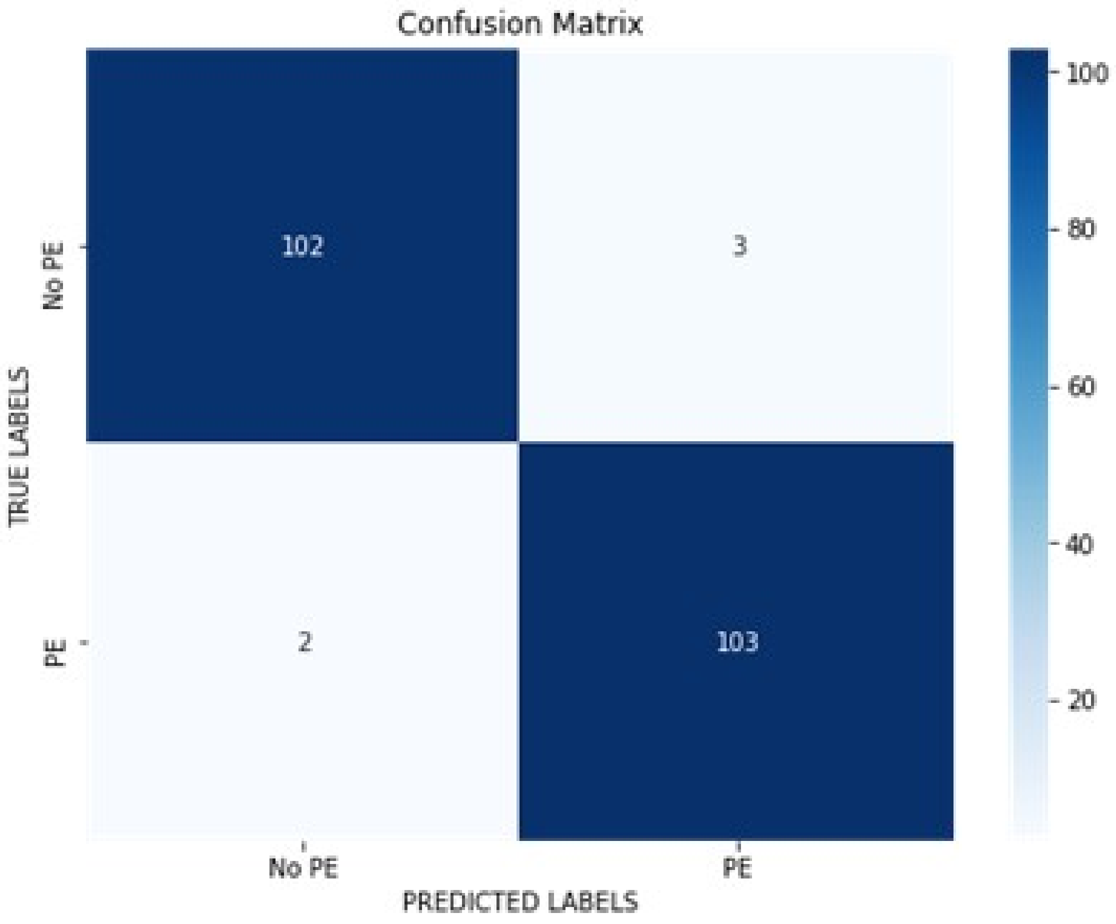
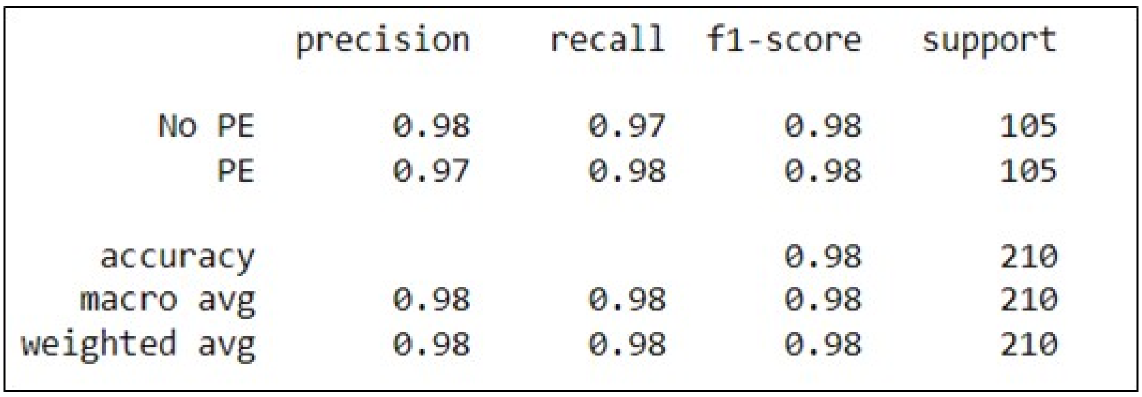


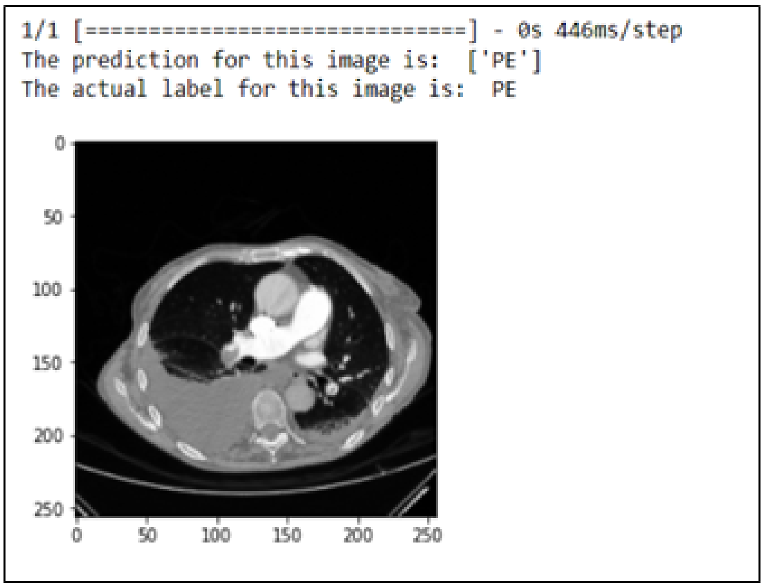
Fig 4.1 : Confusion Matrix

Table 4.1 represents the classification report that is used to assess the quality of predictions made by the algorithm.

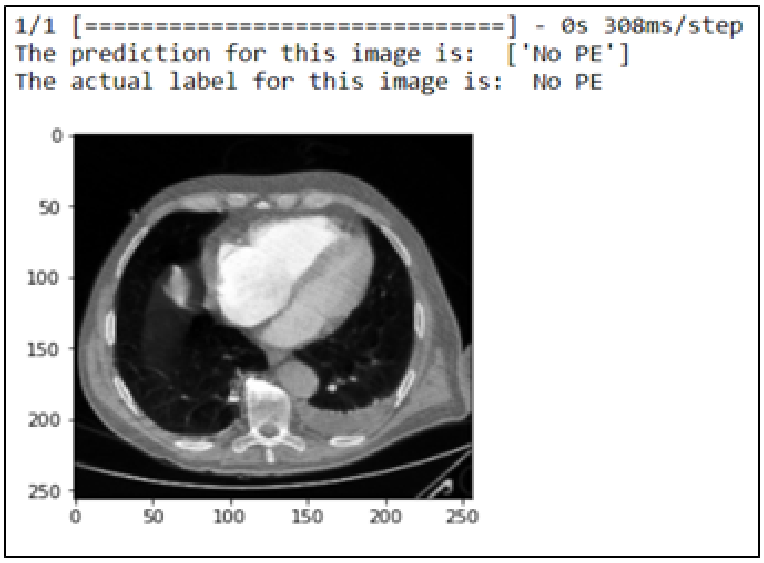


```
              precision    recall  f1-score   support

       No PE       0.98      0.97      0.98       105
          PE       0.97      0.98      0.98       105

    accuracy                           0.98       210
   macro avg       0.98      0.98      0.98       210
weighted avg       0.98      0.98      0.98       210
```

Our model achieves an accuracy of 97.61% (approx. 98.00%) and an overall sensitivity of 97.00%, i,e; the model can successfully predict if a given image does not belong to a particular class 97 times out of 100. For the “No PE” class, the sensitivity is seen to be 98.00%, while for the “PE” class, it is observed to be 97.00As for the Precision and Recall values, our model achieves a Precision of 98.00% for “No PE” class, and 97.00% for “PE” class, and we have a Recall of 97.00% for “No PE” and 98.00% for “PE”. Our model achieves a F1-score of 98.00% for both “PE” and “No PE” classes. At the end of the coding process, we checked results on few images from the dataset, to predict whether they can detect PE or not as shown in Fig 4.2.



```
1/1 [==============================] - 0s 446ms/step
The prediction for this image is:  ['PE']
The actual label for this image is:  PE
```



```
1/1 [==============================] - 0s 308ms/step
The prediction for this image is:  ['No PE']
The actual label for this image is:  No PE
```

Fig 4.2 : Few images from dataset

5. Conclusion and Future Directions

So, the VGG16 model combined with XGBOOST has successfully classified the CTPA images into two classes, and through this classification, we get to know which image has detected Pulmonary Embolism.

The future scope of this project can be quite promising, as medical imaging and healthcare continue to evolve with advancements in technology. Here are some potential areas of future development and application:

1. Improved accuracy: As deep learning algorithms and models continue to advance, there is scope for enhancing the accuracy of medical image classification. This can lead to more reliable and precise diagnoses, aiding healthcare professionals in making informed decisions.
2. Integration with healthcare systems: The project can be extended to integrate with existing healthcare systems and electronic health records (EHRs). This integration can enable seamless access to medical image classification results and facilitate better patient management and treatment planning.
3. Expansion to other medical conditions: The project can be expanded to classify images related to various other medical conditions. This can include cancer detection, cardiovascular disease diagnosis, neurological disorder identification, and more. This would contribute to a broader range of applications in healthcare.
4. Real-time image analysis: The project can be optimized to perform image classification in real-time, allowing for immediate analysis and decision-making during medical procedures or emergencies. This can be especially valuable in time-sensitive situations.
5. Collaboration with medical experts: Collaborating with medical professionals and researchers can provide valuable insights and feedback for further improvement of the project. This interdisciplinary approach can help bridge the gap between technology and healthcare, leading to more effective solutions.
6. Deployment in edge computing: As computing capabilities improve, deploying the project on edge devices can enable on-device image classification, reducing the need for extensive data transfer and improving privacy and security. This can be particularly beneficial in resource-constrained environments or remote areas with limited connectivity.

Bibliography

[1] Philip A Araoz et al. "Panel discussion: pulmonary embolism imaging and outcomes." In: *American Journal of Roentgenology* 198.6 (2012), pp. 1313–1319.

[2] Haydar Özkan et al. "A novel method for pulmonary embolism detection in CTA images." In: *Computer methods and programs in biomedicine* 113.3 (2014), pp. 757–766.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[4] Samir S Yadav and Shivajirao M Jadhav. "Deep convolutional neural network based medical image classification for disease diagnosis." In: *Journal of Big data* 6.1 (2019), pp. 1–18.

[5] Xin Yang et al. "A two-stage convolutional neural network for pulmonary embolism detection from CTPA images." In: *IEEE Access* 7 (2019), pp. 84849–84857.

[6] Vivekanadam Balasubramaniam. "Artificial intelligence algorithm with SVM classification using dermascopic images for melanoma diagnosis." In: *Journal of Artificial Intelligence and Capsule Networks* 3.1 (2021), pp. 34–42.

[7] Errol Colak et al. "The RSNA pulmonary embolism CT dataset." In: *Radiology: Artificial Intelligence* 3.2 (2021), e200254.

[8] Dhritiman Mukherje and Aman Anand. "On edge deep learning implementation: approach to achieve 5G." In: *Multimedia Tools and Applications* 82.8 (2023), pp. 12229–12243.