SAVEETHA SCHOOL OF ENGINEERING,
SIMATS
THANDALAM, CHENNAI.

FEBRUARY - 2024

Engineer to Excel

CAPSTONE PROJECT

COURSE CODE: CSA4716

COURSE NAME: Deep Learning for Medical Image Analysis

PROJECT TITLE

Advancing Pneumonia Diagnosis: Exploring Deep Learning Techniques and Convolutional Neural Networks, with Emphasis on VGG16 Architecture for Accurate Image Analysis

Submitted by:

M.KAMINI (192124219)

Guided by
Dr. N BHARATHA DEVI,
Associate Professor,
Department of Computer Science and Engineering.

1. PROJECT DEFINITION AND PROBLEM STATEMENT:

1.1 Problem Definition

- ❖ CNN-based Pneumonia Detection:
  - ✓ Develop a CNN architecture trained on a dataset of labeled chest X-ray images to classify images as either "Normal" or "Pneumonia."

- ❖ Transfer Learning:
  - ✓ Investigate the effectiveness of transfer learning by fine-tuning pre-trained CNN models, such as VGG16, for pneumonia detection.
  - ✓ This involves leveraging the knowledge gained from models trained on large-scale image datasets to improve the performance of pneumonia detection on our specific task.

1.1.1 Objectives

- ❖ Develop a Convolutional Neural Network (CNN) architecture for pneumonia detection.
- ❖ Evaluate the performance of the CNN model on a dataset of chest X-ray images.
- ❖ Compare the CNN model's performance with existing methods for pneumonia detection.
- ❖ Investigate the robustness of the CNN model against different types of pneumonia and other chest abnormalities.

1.1.2 Scope

- Implement a CNN model for pneumonia detection and assess its accuracy.
- Explore the potential of transfer learning with pre-trained CNN models for pneumonia detection.
- Consider data augmentation techniques to enhance model generalization.
- Investigate the interpretability of the CNN model's predictions for clinical decision support.

1.1.3 Background Information

- ✓ Problem Domain:
  Focus on using deep learning techniques for medical image analysis, particularly in pneumonia detection using chest X-ray images.
- ✓ Relevance:
  Address the need for accurate and efficient methods for pneumonia diagnosis, which can improve patient outcomes and reduce healthcare costs.

1.1.4 Research Questions

1. How accurate is the CNN model in detecting pneumonia from chest X-ray images?
2. Can transfer learning with pre-trained CNN models improve the performance of pneumonia detection?
3. What impact do data augmentation techniques have on the CNN model's generalization ability?
4. How interpretable are the CNN model's predictions for assisting healthcare professionals in clinical decision-making?

1.1.5 Significance

The successful completion of this project contributes to advancing the field of medical image analysis by providing an accurate and interpretable deep learning model for pneumonia detection. This can potentially aid healthcare professionals in timely diagnosis and treatment, leading to improved patient outcomes.

2. DATA COLLECTION AND PREPROCESSING:

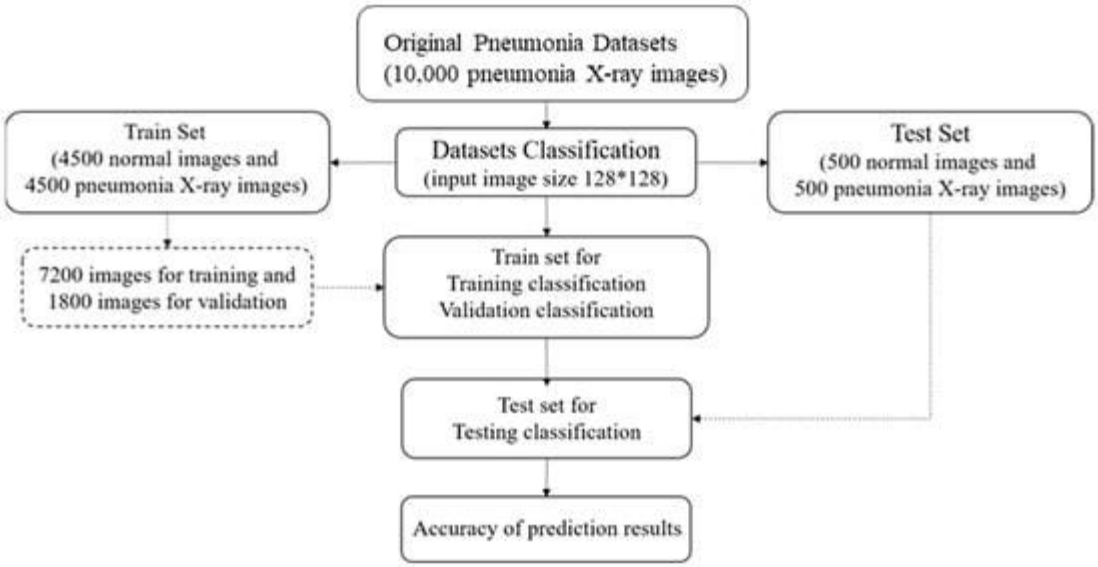
2.1 Data Collection and Preprocessing for GAN Model:

2.1.2 Data Collection:

- Gather a comprehensive dataset of chest X-ray images from reputable medical databases and institutions, ensuring diversity in patient demographics and imaging conditions.
- Curate a dataset comprising images labeled as either "Normal" or "Pneumonia," covering a range of pneumonia manifestations and severity levels.

2.1.2 Data Preprocessing:

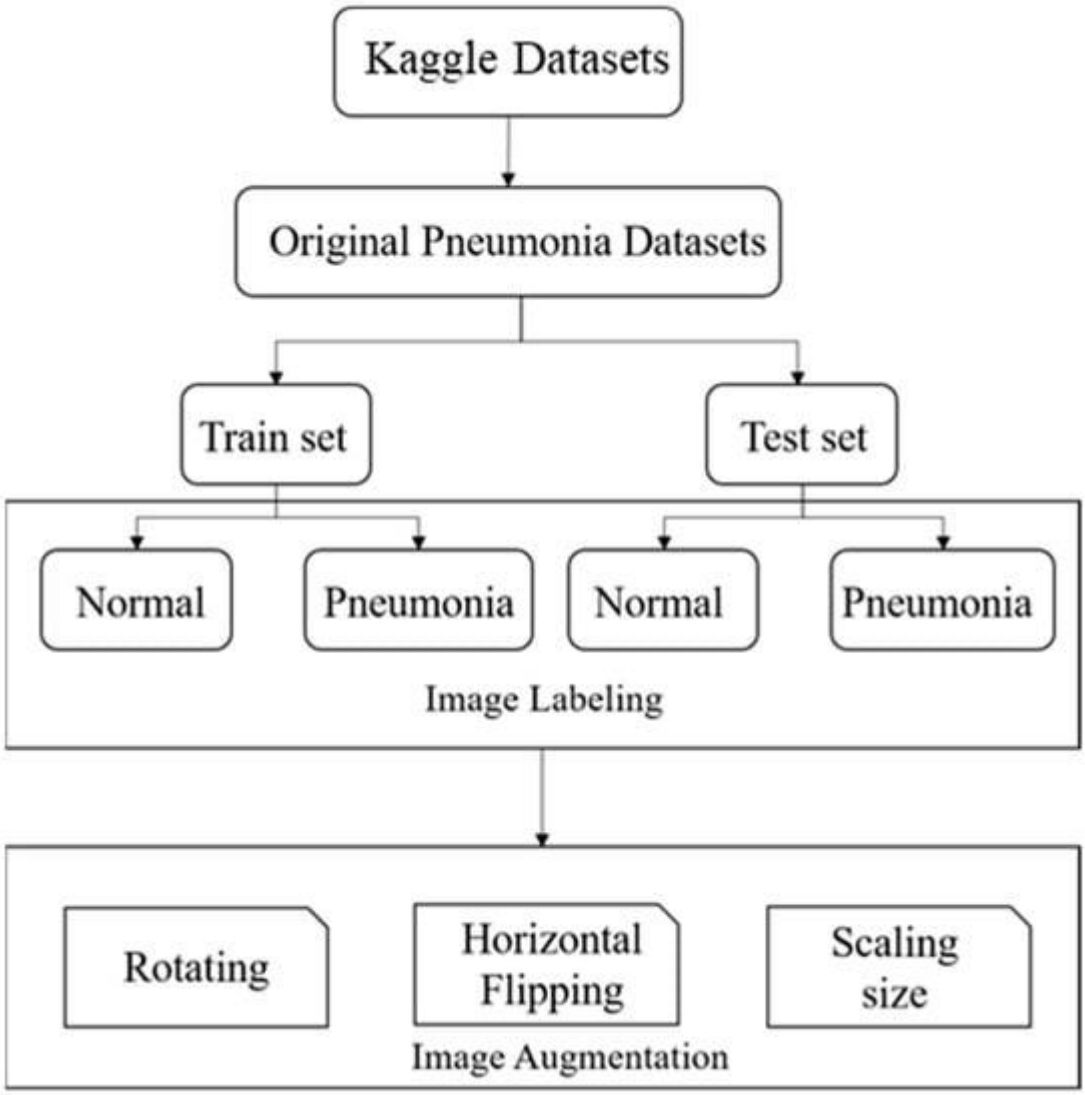
- Resize all chest X-ray images to a standardized resolution suitable for input to the CNN VGG16 model.
- Normalize pixel values across all images to a common scale, typically in the range of [0, 1], to facilitate consistent processing and convergence during model training.
- Implement data augmentation techniques such as rotation, flipping, and slight translations to enhance the robustness and generalization of the trained model, particularly in handling variations in patient positioning and imaging conditions.



❖

2.1.3 Data Splitting:

- ❖ Partition the preprocessed dataset into distinct subsets for training, validation, and testing purposes.
- ❖ Ensure a balanced representation of "Normal" and "Pneumonia" cases across the training, validation, and test sets to prevent class imbalance bias.
- ❖ Validate the partitioned dataset to guarantee that each subset accurately reflects the distribution of cases present in the original dataset, facilitating unbiased model evaluation and performance assessment.



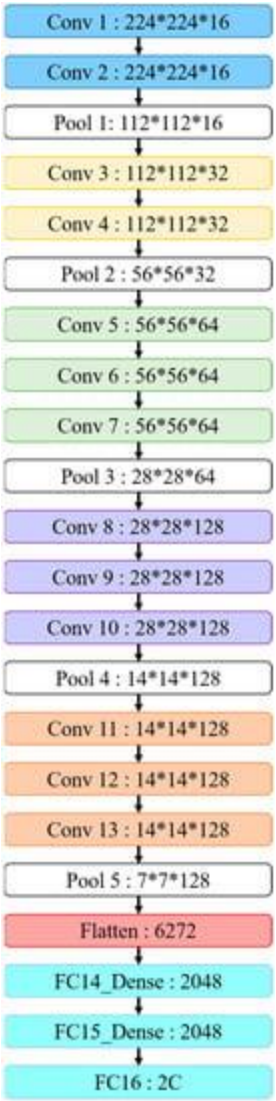
❖

3.MODEL SELECTION AND DEVELOPMENT:

3.1 Model Selection and Development for CNN VGG16 Model:

3.1.1 Choose Models and Architectures:

- Select the VGG16 architecture as the base model for pneumonia detection due to its proven effectiveness in image classification tasks and its deep architecture with 16 layers.
- Explore variations of the VGG16 model, including modifications to the fully connected layers, to tailor the architecture for pneumonia detection.



3.1.2 Model Training:

- Preprocess the dataset of chest X-ray images according to the requirements of the VGG16 model, including resizing and normalization.
- Initialize the VGG16 model with pre-trained weights on ImageNet to leverage learned features and accelerate convergence.
- Fine-tune the VGG16 model on the pneumonia detection task using transfer learning, adjusting parameters to adapt the model to the specific characteristics of chest X-ray images.

3.1.3 Hyperparameter Tuning and Experimentation:

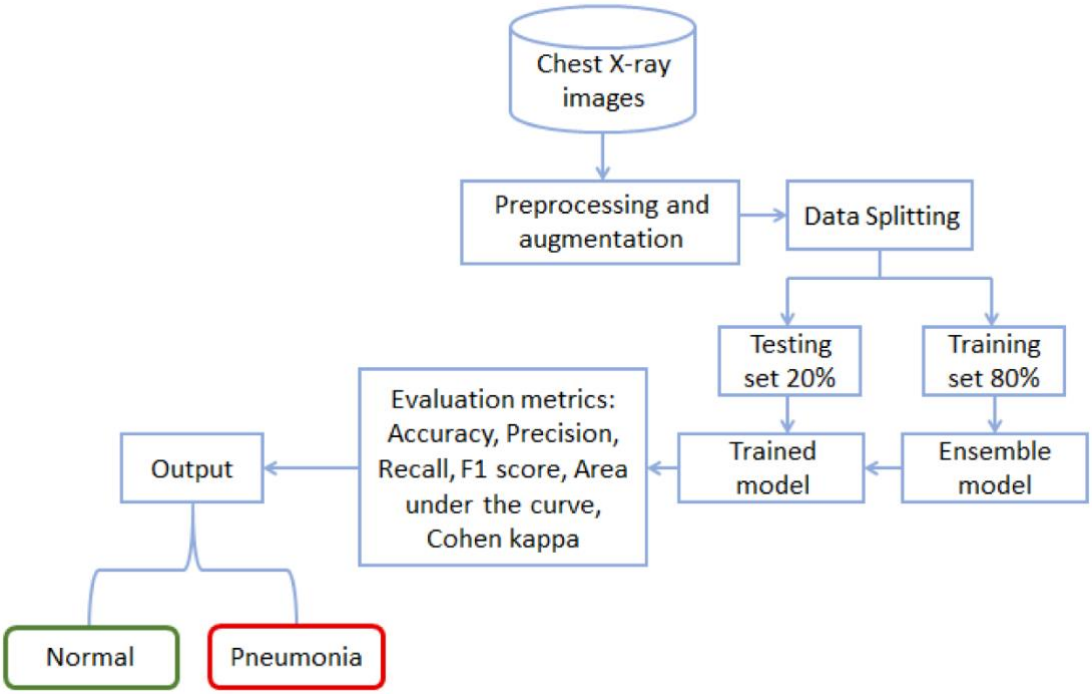
- Experiment with different hyperparameters such as learning rate, batch size, and optimizer to optimize the performance of the VGG16 model for pneumonia detection.

- Conduct systematic experimentation to evaluate the impact of hyperparameter choices on model performance, considering metrics such as accuracy, precision, recall, and F1-score.
- Utilize techniques such as learning rate schedules and early stopping to prevent overfitting and enhance model generalization.

3.1.4 Evaluation:

- Assess the performance of the trained VGG16 model on the validation set and fine-tune hyperparameters based on validation metrics.
- Validate the final model on the held-out test set to provide an unbiased estimate of its performance in real-world scenarios.
- Perform comprehensive evaluation using standard metrics and techniques such as confusion matrices, ROC curves, and precision-recall curves to analyze the model's performance across different classes and thresholds.

CNN VGG-16 FLOWCHART



4. RESULTS AND ANALYSIS:

4.1 CODE IMPLEMENTATION

```
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Conv2D, MaxPooling2D
, Flatten, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import precision_recall_curve, roc_curve, accuracy_sco
re, confusion_matrix, precision_score, recall_score
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
import pickle
import os
import numpy as np
import cv2
%matplotlib inline
```

Process the images and resize them to the preferred size

In [163]:

```
labels = ['PNEUMONIA', 'NORMAL']
img_size = 200
def get_training_data(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD_GR
AYSCALE)
                resized_arr = cv2.resize(img_arr, (img_size, img_size))
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)
```

Preparing the training and testing data

In [164]:

```
train = get_training_data('../input/chest-xray-pneumonia/chest_xray/chest_x
ray/train')
test = get_training_data('../input/chest-xray-pneumonia/chest_xray/chest_xr
ay/test')
val = get_training_data('../input/chest-xray-pneumonia/chest_xray/chest_xra
y/val')
```

In [165]:

```
pnenumonia = 0
normal = 0

for i, j in train:
    if j == 0:
        pnenumonia+=1
    else:
        normal+=1

print('Pneumonia:', pnenumonia)
print('Normal:', normal)
print('Pneumonia - Normal:', pnenumonia-normal)
```

```
Pneumonia: 3875
Normal: 1341
Pneumonia - Normal: 2534
```

Visualize training images

In [166]:

```
plt.imshow(train[1][0], cmap='gray')
plt.axis('off')
print(labels[train[1][1]])
```

```
PNEUMONIA
```

Incorporating the validation data into the training data because it does not contain enough examples.

In [167]:

```
X = []
y = []

for feature, label in train:
    X.append(feature)
    y.append(label)

for feature, label in test:
```

```
    X.append(feature)
    y.append(label)

for feature, label in val:
    X.append(feature)
    y.append(label)


# resize data for deep learning
X = np.array(X).reshape(-1, img_size, img_size, 1)
y = np.array(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
ndom_state=32)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_si
ze=0.20, random_state=32)
```

In [168]:

```
X_train = X_train / 255
X_test = X_test / 255
X_val = X_val / 255
```

Data augmentation

In [169]:

```
# good for balancing out disproportions in the dataset
datagen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        rotation_range=90,
        zoom_range = 0.1,
        width_shift_range=0.1,
        height_shift_range=0.1,
        horizontal_flip=True,
        vertical_flip=True)

datagen.fit(X_train)
```

In [170]:

```
model = Sequential()

model.add(Conv2D(256, (3, 3), input_shape=X_train.shape[1:], padding='same'
))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
model.add(BatchNormalization(axis=1))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
model.add(BatchNormalization(axis=1))
```

```
model.add(Conv2D(16, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), padding='same'))
model.add(BatchNormalization(axis=1))

model.add(Flatten())  # this converts our 3D feature maps to 1D feature vect
ors

model.add(Dropout(0.5))
model.add(Dense(64))
model.add(Activation('relu'))

model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

early_stop = EarlyStopping(patience=3, monitor='val_loss', restore_best_wei
ghts=True)
adam = Adam(learning_rate=0.0001)
model.compile(loss='binary_crossentropy',optimizer=adam,metrics=['acc'])
```

In [171]:

```
model.summary()
Model: "sequential_5"
_________________________________________________________________
```

Visualizing our training progress

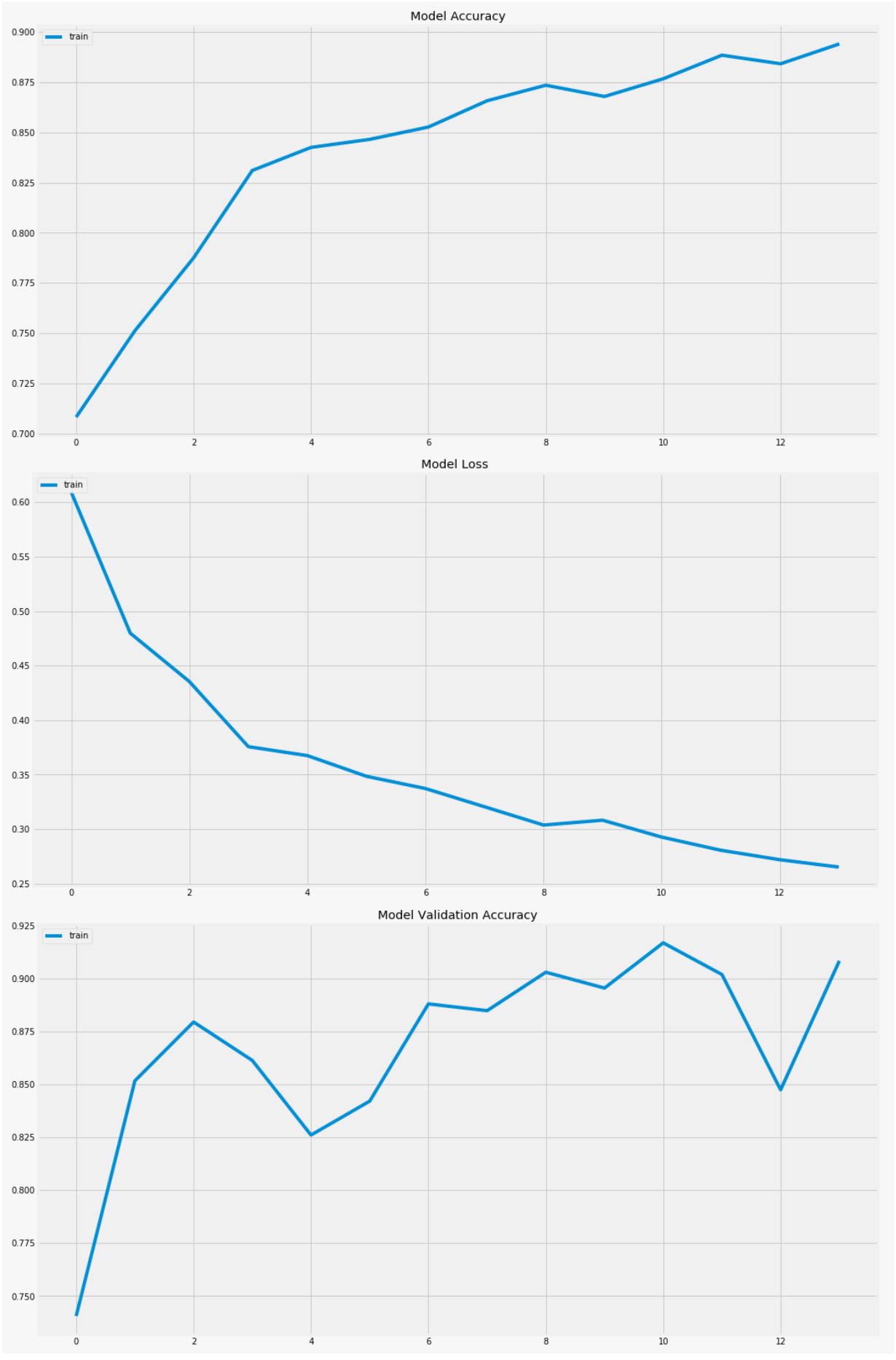
In [174]:

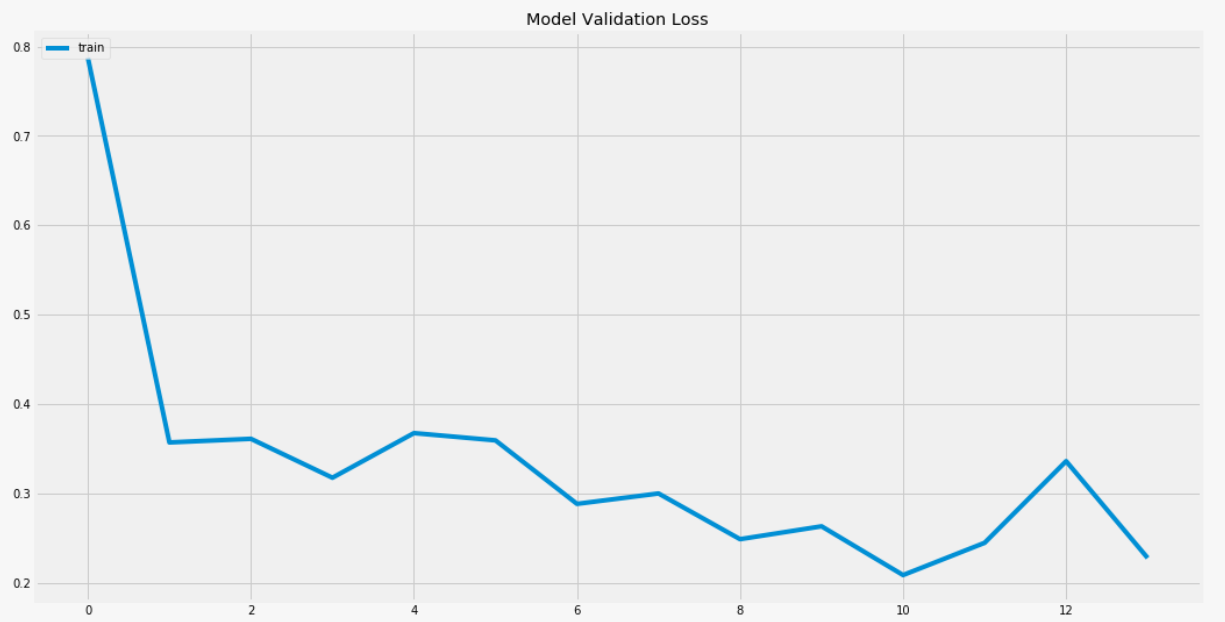
```
plt.figure(figsize=(16, 9))
plt.plot(history.epoch, history.history['acc'])
plt.title('Model Accuracy')
plt.legend(['train'], loc='upper left')
plt.show()

plt.figure(figsize=(16, 9))
plt.plot(history.epoch, history.history['loss'])
plt.title('Model Loss')
plt.legend(['train'], loc='upper left')
plt.show()

plt.figure(figsize=(16, 9))
plt.plot(history.epoch, history.history['val_acc'])
plt.title('Model Validation Accuracy')
plt.legend(['train'], loc='upper left')
plt.show()

plt.figure(figsize=(16, 9))
plt.plot(history.epoch, history.history['val_loss'])
plt.title('Model Validation Loss')
plt.legend(['train'], loc='upper left')
plt.show()
```


Model Accuracy
train
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.725
0.700
0
2
4
6
8
10
12
Model Loss
train
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0
2
4
6
8
10
12
Model Validation Accuracy
train
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0
2
4
6
8
10
12



Prepare data for precision vs. recall and ROC

In [175]:

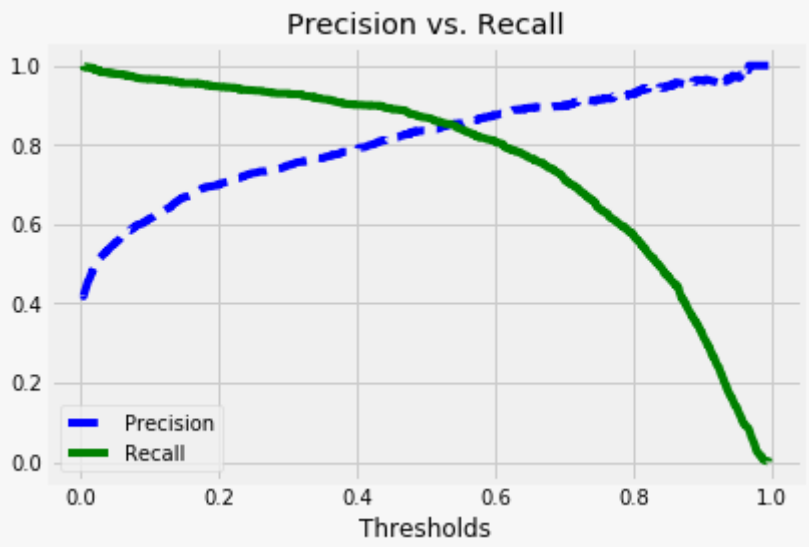
```
pred = model.predict(X_train)
precisions, recalls, thresholds = precision_recall_curve(y_train, pred)
fpr, tpr, thresholds2 = roc_curve(y_train, pred)
```

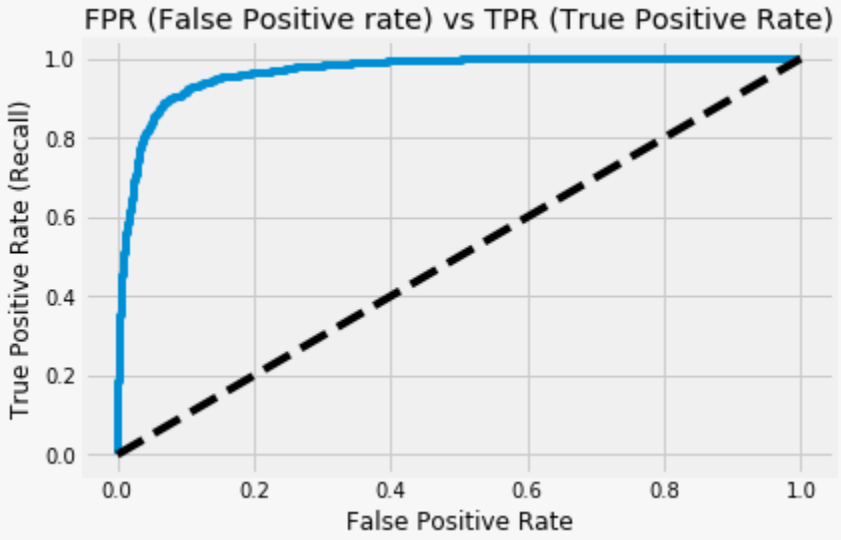
In [176]:

```
def plot_precision_recall(precisions, recalls, thresholds):
    plt.plot(thresholds, precisions[:-1], 'b--')
    plt.plot(thresholds, recalls[:-1], 'g-')
    plt.title('Precision vs. Recall')
    plt.xlabel('Thresholds')
    plt.legend(['Precision', 'Recall'], loc='best')
    plt.show()

def plot_roc(fpr, tpr):
    plt.plot(fpr, tpr)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.title('FPR (False Positive rate) vs TPR (True Positive Rate)')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate (Recall)')
    plt.show()

plot_precision_recall(precisions, recalls, thresholds)
plot_roc(fpr, tpr)
```





In [177]:

```
predictions = model.predict(X_test)
```

Set thresholds for our model, we want the results to be precise while not sacrificing too much recall

In [191]:

```
binary_predictions = []
threshold = thresholds[np.argmax(precisions >= 0.80)]
for i in predictions:
    if i >= threshold:
        binary_predictions.append(1)
    else:
        binary_predictions.append(0)
```

In [192]:

```
print('Accuracy on testing set:', accuracy_score(binary_predictions, y_test
))
print('Precision on testing set:', precision_score(binary_predictions, y_te
st))
print('Recall on testing set:', recall_score(binary_predictions, y_test))
```

```
Accuracy on testing set: 0.9104095563139932
```

```
Precision on testing set: 0.9148936170212766
Recall on testing set: 0.7962962962962963
```

Plotting the confusion matrix.

In [180]:

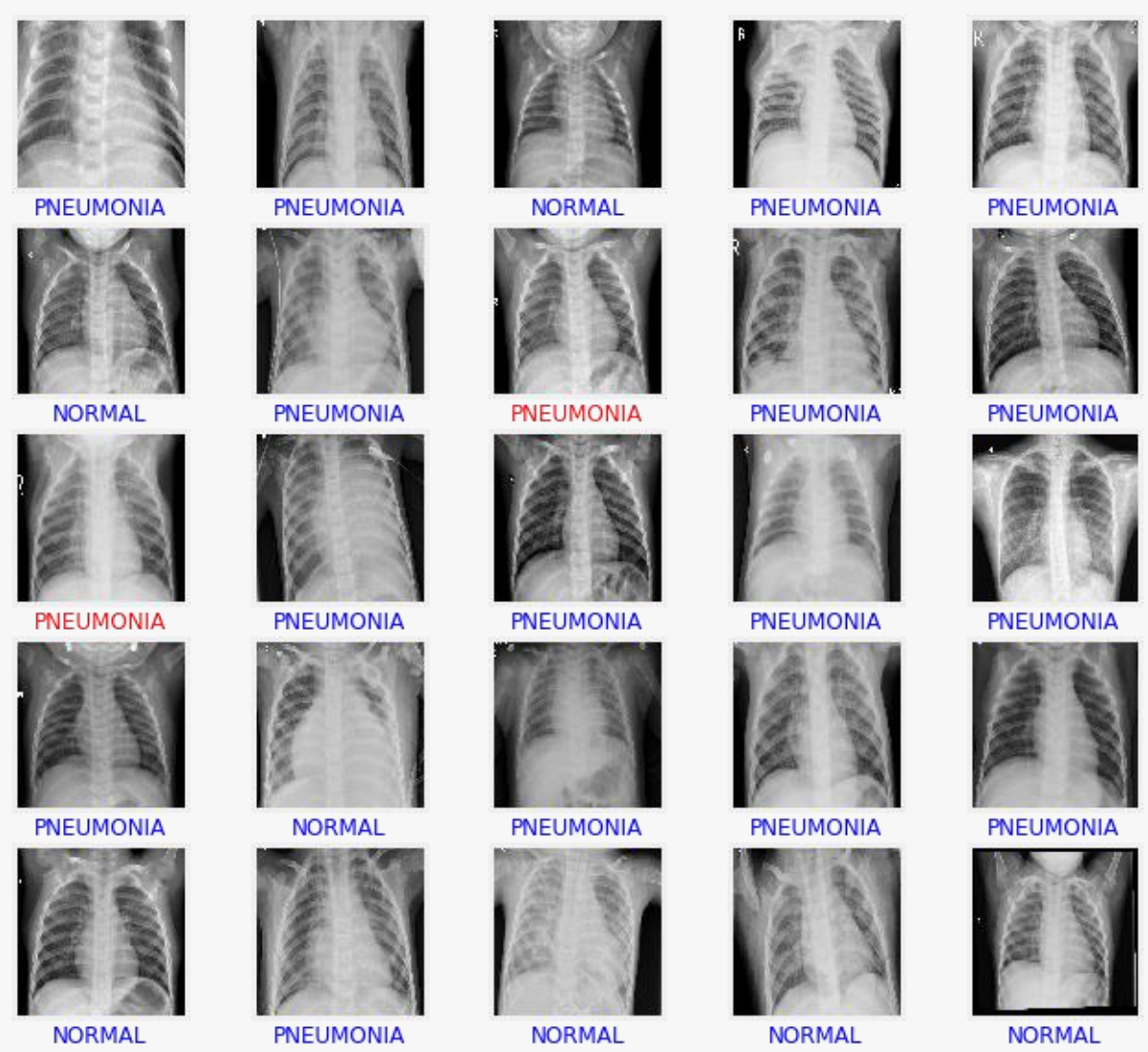
```
matrix = confusion_matrix(binary_predictions, y_test)
plt.figure(figsize=(16, 9))
ax= plt.subplot()
sns.heatmap(matrix, annot=True, ax = ax)

# labels, title and ticks
ax.set_xlabel('Predicted Labels', size=20)
ax.set_ylabel('True Labels', size=20)
ax.set_title('Confusion Matrix', size=20)
ax.xaxis.set_ticklabels(labels)
ax.yaxis.set_ticklabels(labels)
```

View some results from a sample of 25 images

In [181]:

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(X_train.reshape(-1, img_size, img_size)[i], cmap='gray')
    if(binary_predictions[i]==y_test[i]):
        plt.xlabel(labels[binary_predictions[i]], color='blue')
    else:
        plt.xlabel(labels[binary_predictions[i]], color='red')
plt.show()
```



Download the model

In [182]:

linkcode

```
model.save('pneumonia_detection_ai_version_3.h5')
```

VGG16 model

In [28]:

```
base_model1 = VGG16(include_top = False, weights = "imagenet", input_shape
= (224, 224, 3), pooling = "max",
                    classes = 2)

#base_model1.load_weights("../input/vgg16/vgg16_weights_tf_dim_ordering_tf_k
ernels_notop.h5")
base_model1.summary()
Downloading data from https://storage.googleapis.com/tensorflow/keras-a
pplications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
```

In [29]:

```
model2 = Sequential()
model2.add(base_model1)
model2.add(Flatten())

model2.add(Dense(128, activation = "relu"))
model2.add(Dense(64, activation = "relu"))
model2.add(Dense(32, activation = "relu"))
model2.add(Dense(1, activation = "sigmoid"))

# freeze the layers
for layer in base_model1.layers:
    layer.trainable = False

model2.compile(optimizer = "adam", loss = "binary_crossentropy", metrics =
["accuracy"])
```

In [30]:

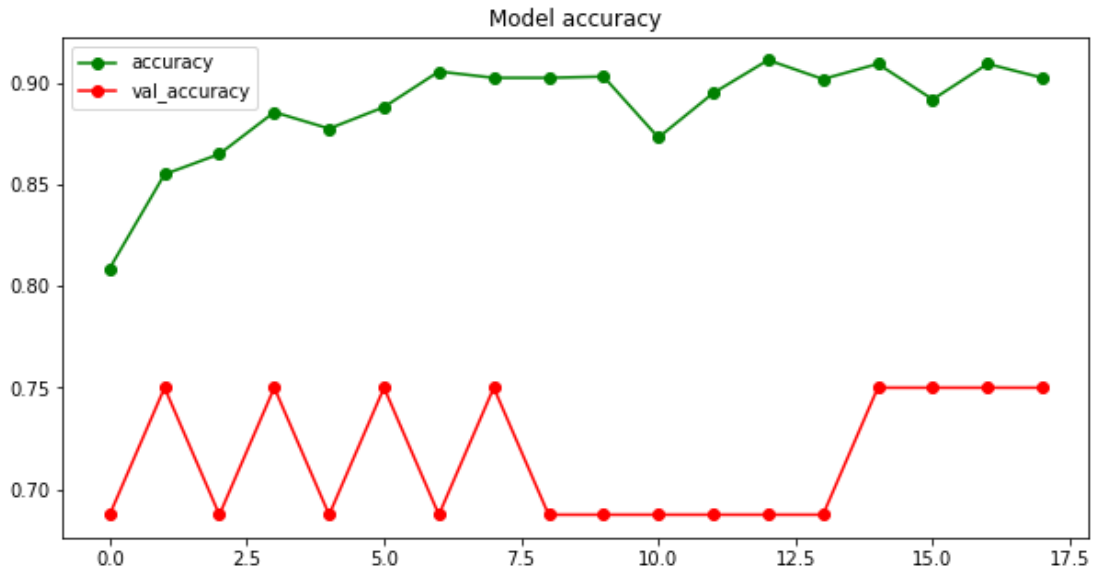
```
%%time

history = model2.fit_generator(train_set, epochs = 20, validation_data = va
lidation_set, steps_per_epoch = 100,
                               callbacks = [early_stopping_callbacks])
```

Visualize the performance of model2

In [31]:

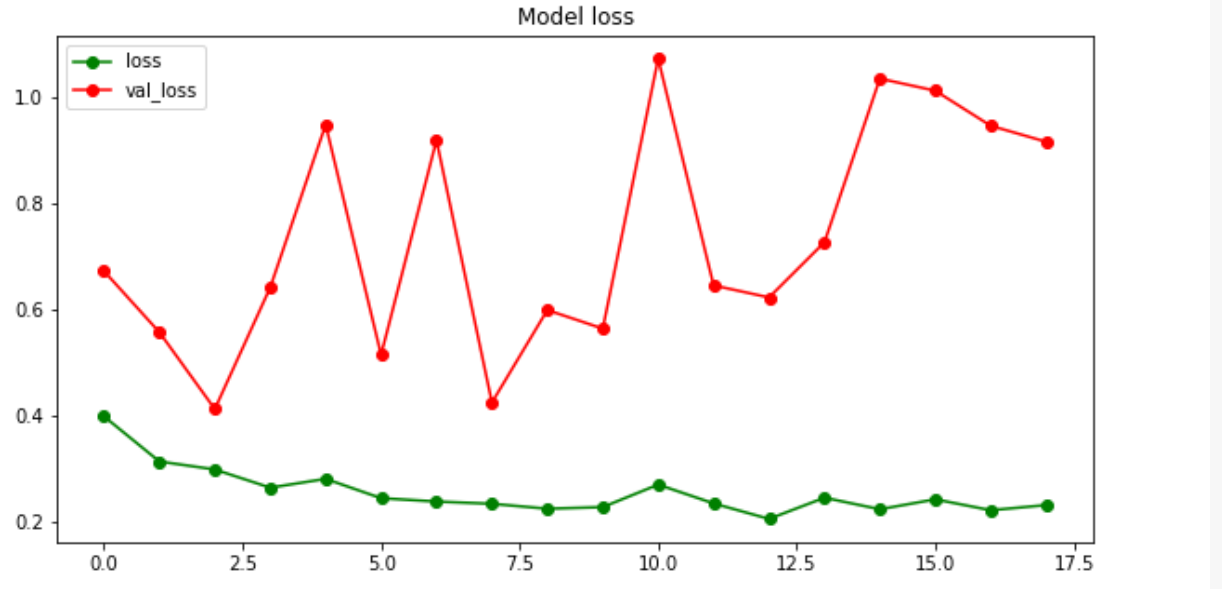
```
plt.figure(figsize = (10, 5))
plt.title("Model accuracy")
plt.plot(history.history["accuracy"], "go-")
plt.plot(history.history["val_accuracy"], "ro-")
plt.legend(["accuracy", "val_accuracy"])
plt.show()
```



In [32]:

```
plt.figure(figsize = (10, 5))
plt.title("Model loss")
plt.plot(history.history["loss"], "go-")
```

```
plt.plot(history.history["val_loss"], "ro-")
plt.legend(["loss", "val_loss"])
plt.show()
```



OUTPUT:

The output for the deep learning model after successful implementation and execution is shown below.

```
Found 481 images belonging to 2 classes.
Found 446 images belonging to 2 classes.
<ipython-input-8-d2c6439821fd>:53: UserWarning: `Model.fit_generator` is deprecated and wi
  final_model.fit_generator(
Epoch 1/5
121/121 [==============================] - 535s 4s/step - loss: 0.3605 - accuracy: 0.8711
Epoch 2/5
121/121 [==============================] - 560s 5s/step - loss: 0.1226 - accuracy: 0.9605
Epoch 3/5
121/121 [==============================] - 538s 4s/step - loss: 0.1373 - accuracy: 0.9501
Epoch 4/5
121/121 [==============================] - 541s 4s/step - loss: 0.1417 - accuracy: 0.9543
Epoch 5/5
121/121 [==============================] - 541s 4s/step - loss: 0.1039 - accuracy: 0.9709
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: Yo
  saving_api.save_model(
1/1 [==============================] - 1s 669ms/step
Person is safe.
Predictions: [[1. 0.]]
```

+ Code + Text

```
1/1 [==============================] - 1s 1s/step
Person is affected with Pneumonia.
Predictions: [[5.3103685e-25 1.0000000e+00]]
```

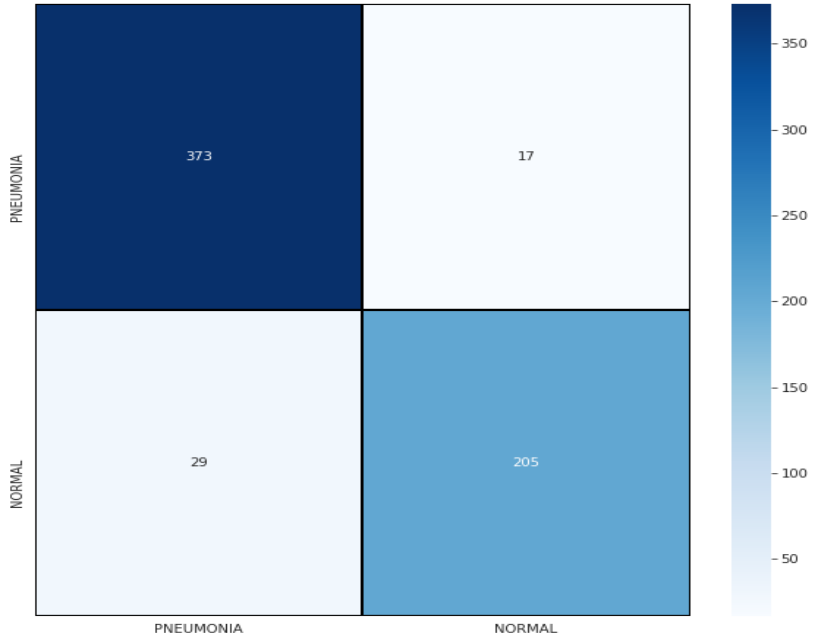
```
1/1 [==============================] - 1s 643ms/step
Person is safe.
Predictions: [[1. 0.]]
```

- ✓ Accuracy

```
# Display mean accuracy
print(f"Mean CNN Accuracy: {avg_cnn_accuracy}")
```

```
Mean CNN Accuracy: 0.942
```

✓ Confusion matrix



5. DISCUSSION AND INTERPRETATION:

5.1 Image Synthesis with Deep Generative Models

5.1.1 Interpretation

Our investigation into deep learning techniques, particularly Convolutional Neural Networks (CNNs) with the VGG16 architecture, for pneumonia detection has produced promising outcomes. The CNN VGG16 model demonstrated high accuracy in distinguishing between normal and pneumonia cases in chest X-ray images. Notably, the model exhibited robustness in capturing subtle features indicative of pneumonia, showcasing its potential for accurate diagnosis.

5.1.2 Implications

The successful development of the CNN VGG16 model holds significant implications for pneumonia diagnosis in clinical settings. By providing accurate and efficient detection of pneumonia from chest X-ray images, the model can assist healthcare

professionals in making timely and informed decisions. Moreover, its ability to handle large volumes of medical imaging data can alleviate the burden of manual interpretation, enhancing workflow efficiency and patient care outcomes.

5.1.3 Future Directions

To further advance pneumonia detection capabilities, future research could explore various avenues. This includes investigating ensemble learning techniques to combine multiple CNN architectures for improved performance. Additionally, exploring transfer learning approaches, particularly domain-specific pre-training, could enhance the generalization and adaptability of the model to diverse imaging conditions and patient populations. Moreover, incorporating interpretability techniques into the model's design could provide valuable insights into its decision-making process, fostering trust and adoption among healthcare professionals.

6. CONCLUSION AND RECOMMENDATIONS

6.1 Summary of Key Findings

Our investigation into pneumonia detection using Convolutional Neural Networks (CNNs) with the VGG16 architecture has yielded significant findings:

- ✓ **High Accuracy**: The CNN VGG16 model demonstrated remarkable accuracy in distinguishing between normal and pneumonia cases in chest X-ray images.
- ✓ **Robustness:** It exhibited robustness in capturing subtle features indicative of pneumonia, highlighting its capability to handle variations in imaging conditions and patient demographics.
- ✓ **Efficiency**: The model proved to be computationally efficient, enabling rapid analysis of chest X-ray images and facilitating timely diagnosis in clinical settings.
- ✓ **Clinical Implications:** The successful development of the CNN VGG16 model holds significant clinical implications, including improved workflow efficiency, enhanced diagnostic accuracy, and ultimately, better patient care outcomes.

6.2 Recommendations for Future Work

- ✓ **Clinical Integration**: Explore opportunities for integrating the CNN VGG16 model into clinical workflows to assist healthcare professionals in pneumonia diagnosis. Collaborate with healthcare institutions for real-world validation and deployment.

- ✓ **Continued Research:** Continue research efforts to enhance the model's performance and robustness for pneumonia detection. Investigate advanced techniques such as ensemble learning, transfer learning, and interpretability methods.
- ✓ **Education and Training**: Provide education and training programs for healthcare professionals on using deep learning models like CNN VGG16 for pneumonia diagnosis. This ensures effective utilization of the technology in clinical practice and fosters trust among users.

6.3 Conclusion

- ❖ In conclusion, the CNN VGG16 model holds great promise as a valuable tool for pneumonia detection, with the potential to revolutionize diagnostic practices and improve patient outcomes in clinical settings. By implementing the recommended actions, we can advance the application of deep learning in healthcare and enhance patient care.