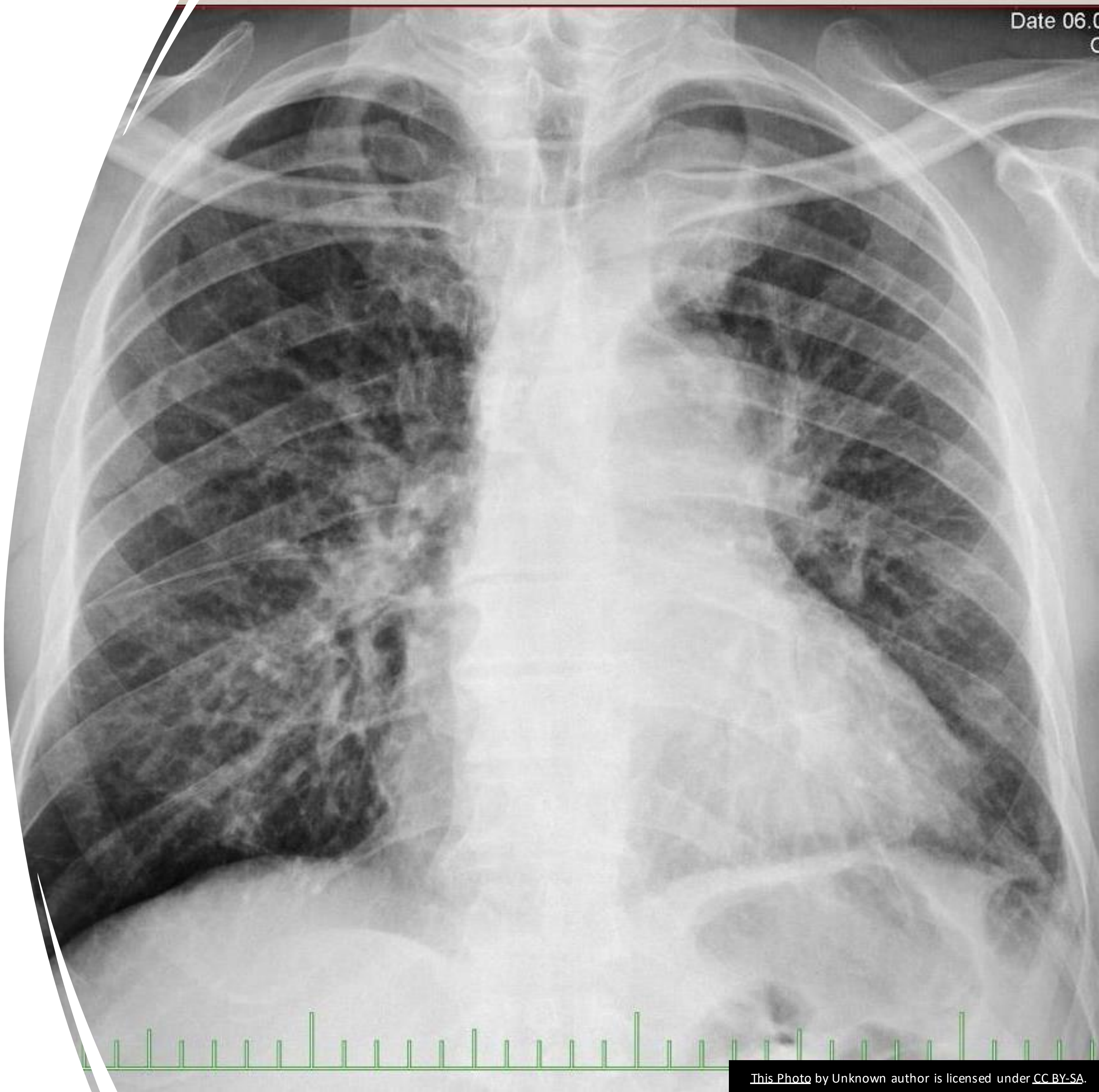
Deep Learning for Medical Image Analysis: Diagnosis and Segmentation of Lung Cancer

Team:
Joshi Komarigiri
Pavan Marturu
Priya(Simran)

PROBLEM

- WHO has reported that Lung cancer causes of 2.09 million deaths annually.
- It is a well-established fact that early diagnosis leads to better prognosis. One of the most difficult tasks in this field is detecting and segmenting lung cancer in CT scan
- Hence my group's project focuses on using Convolutional Neural networks (CNN) to analyze lung scans to detect if the lesions are cancerous. Using AI in healthcare could be a turning point in early detection and as a result early intervention which could save many lives.



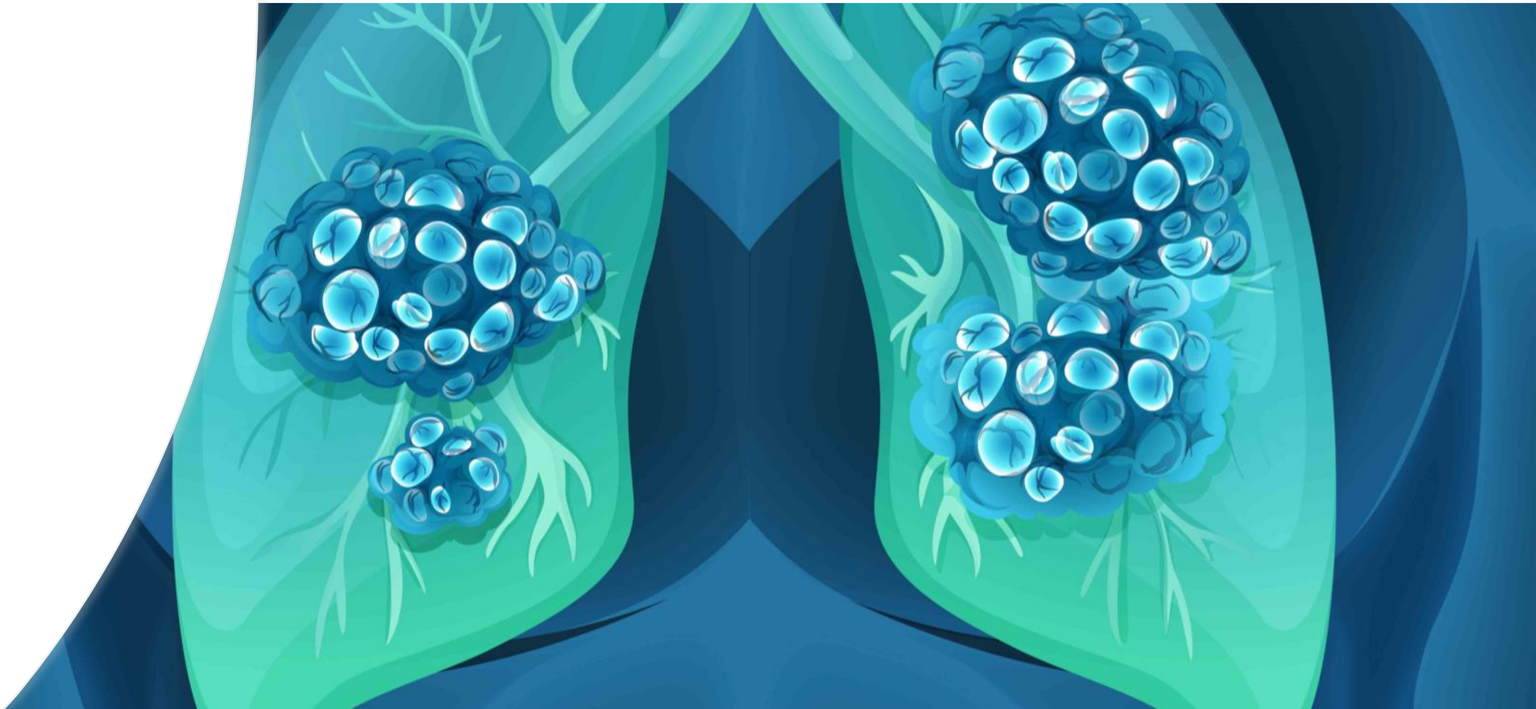
LONG TERM GOALS

- The long-term goal we are trying to achieve is to develop an advanced and more robust image analysis AI model that can improve healthcare outcomes.
- Further we would like to add features and classes that would not only detect cancer but also classify it.
- Major applications of this:

Clinical decision support

Personalized medicine/healthcare

Improved diagnostic accuracy



PROJECT WORKFLOW

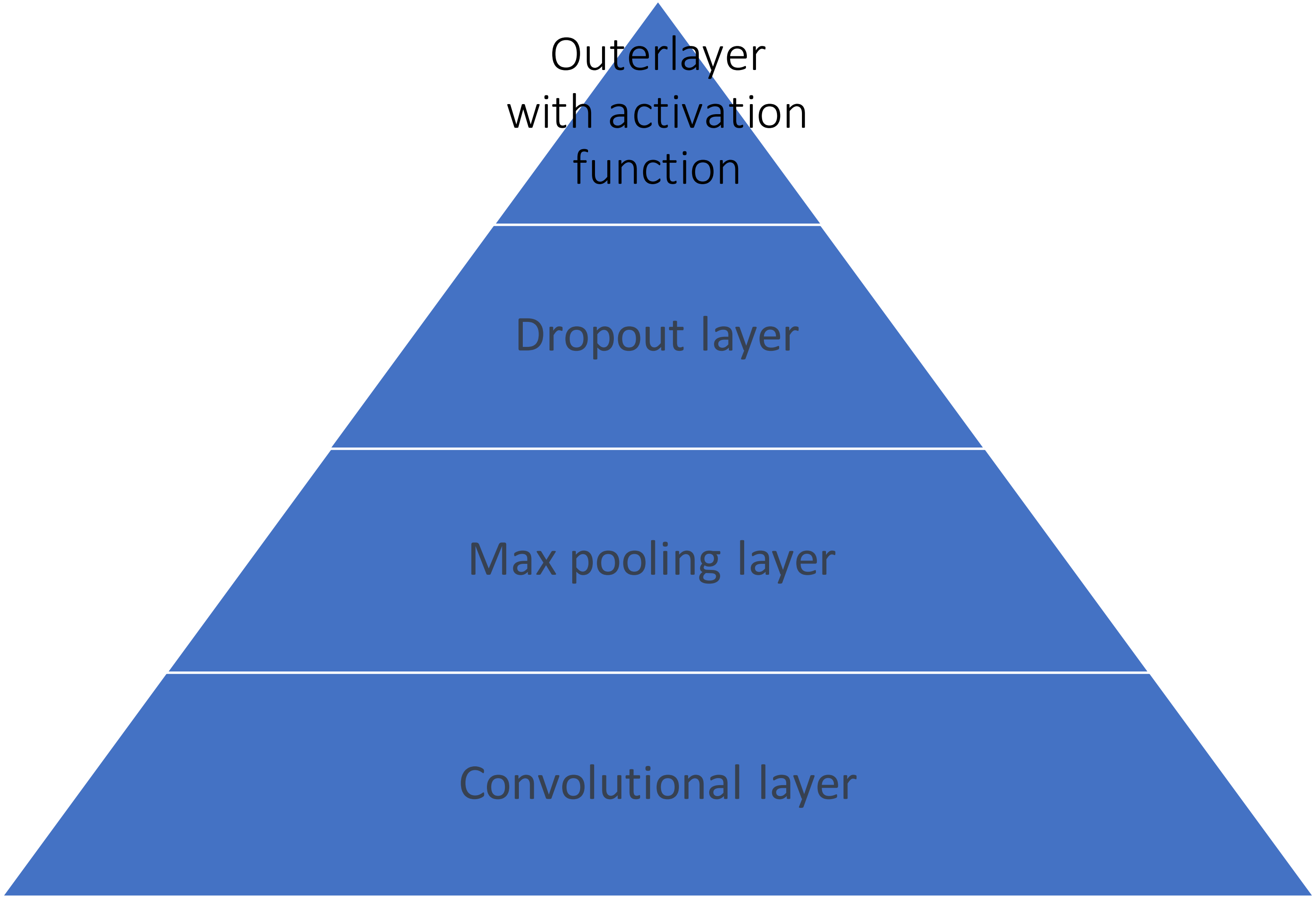
Downloaded dataset from kaggle

the necessary libraries and dependencies are imported

CNN model is created

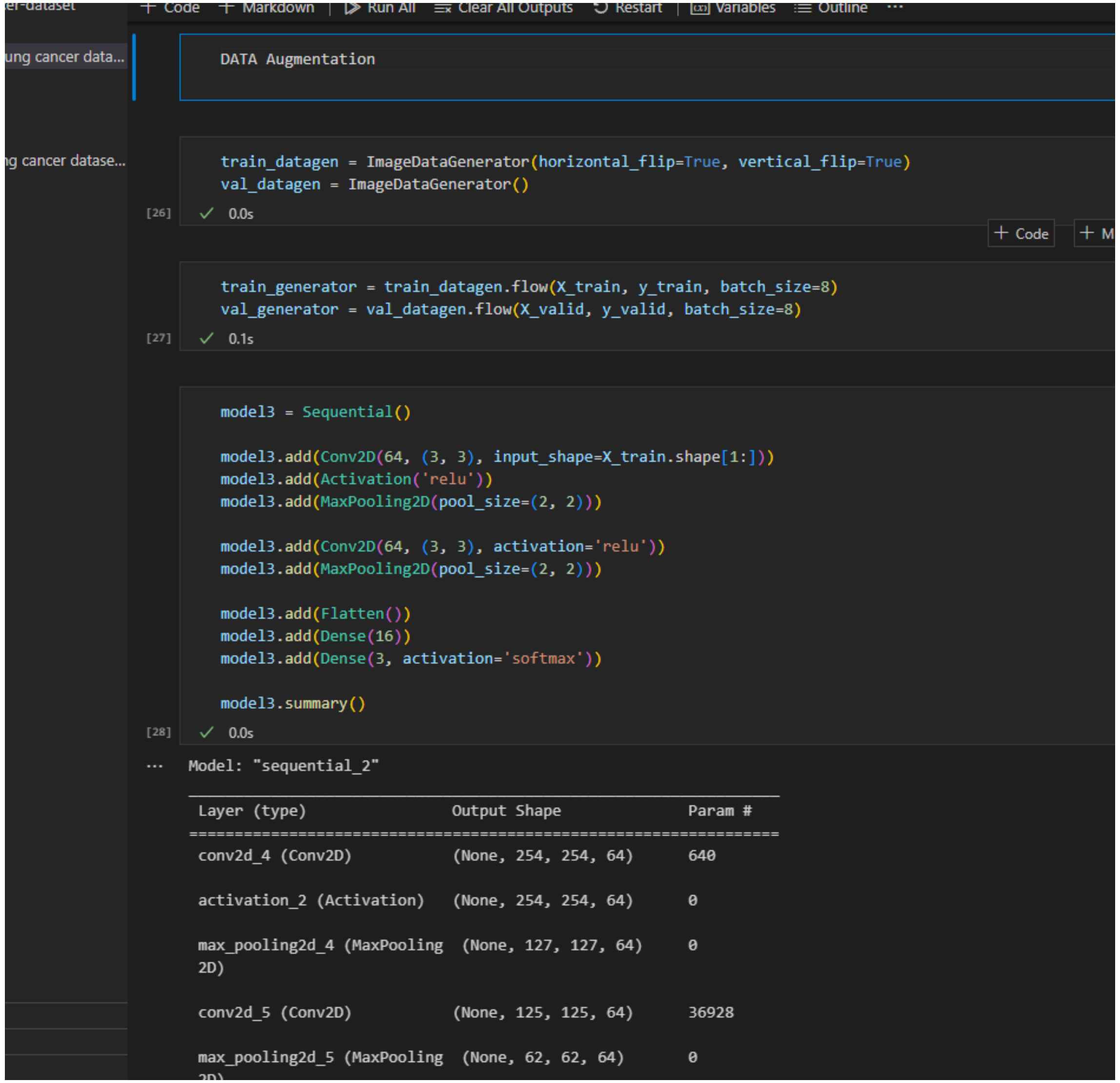
model is compiled and trained on the training data

model's performance is evaluated on the validation data


Outerlayer with activation function
Dropout layer
Max pooling layer
Convolutional layer

DATA AUGMENTATION

- Data augmentation is done using SMOTE TECHNIQUE and 'Imagedatagenerator'
- This function allows for techniques such as random rotations, flips, zooms, and brightness adjustments to be applied on input images during training.
- This helps to increase the variability of input data, thus improving the generalization of data.



IMPLEMENTATION

```python
val_generator = val_datagen.flow(X_valid, y_valid, batch_size=8)
```

[27] ✓ 0.1s

```python
model3 = Sequential()

model3.add(Conv2D(64, (3, 3), input_shape=X_train.shape[1:]))
model3.add(Activation('relu'))
model3.add(MaxPooling2D(pool_size=(2, 2)))

model3.add(Conv2D(64, (3, 3), activation='relu'))
model3.add(MaxPooling2D(pool_size=(2, 2)))

model3.add(Flatten())
model3.add(Dense(16))
model3.add(Dense(3, activation='softmax'))

model3.summary()
```

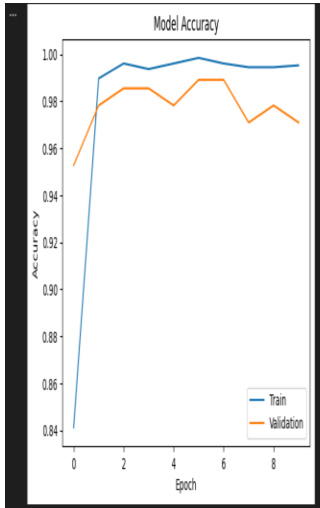
[28] ✓ 0.0s

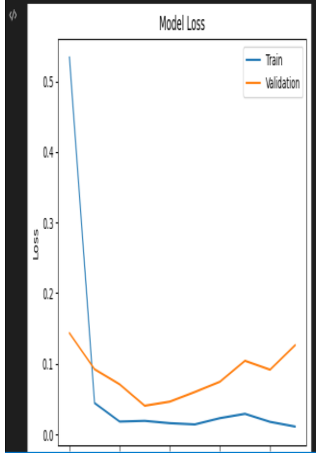
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 254, 254, 64)      640

 activation_2 (Activation)   (None, 254, 254, 64)      0

 max_pooling2d_4 (MaxPooling  (None, 127, 127, 64)     0
 2D)

 conv2d_5 (Conv2D)           (None, 125, 125, 64)      36928

 max_pooling2d_5 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 flatten_2 (Flatten)         (None, 246016)            0

 dense_4 (Dense)             (None, 16)                3936272

 dense_5 (Dense)             (None, 3)                 51

=================================================================
Total params: 3,973,891
Trainable params: 3,973,891
Non-trainable params: 0
_________________________________________________________________
```

```python
model2 = Sequential()

model2.add(Conv2D(64, (3, 3), input_shape=X_train.shape[1:]))
model2.add(Activation('relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))

model2.add(Conv2D(64, (3, 3), activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))

model2.add(Flatten())
model2.add(Dense(16))
model2.add(Dense(3, activation='softmax'))

model2.summary()
```

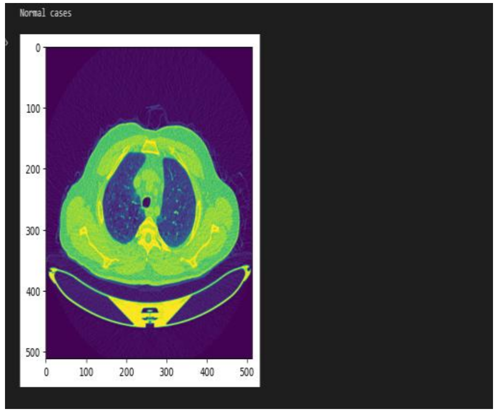
✓ 0.0s

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 254, 254, 64)      640

 activation_1 (Activation)   (None, 254, 254, 64)      0

 max_pooling2d_2 (MaxPooling  (None, 127, 127, 64)     0
 2D)

 conv2d_3 (Conv2D)           (None, 125, 125, 64)      36928

 max_pooling2d_3 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 246016)            0

 dense_2 (Dense)             (None, 16)                3936272

 dense_3 (Dense)             (None, 3)                 51

=================================================================
Total params: 3,973,891
Trainable params: 3,973,891
Non-trainable params: 0
_________________________________________________________________
```






```
model3.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
[29] ✓ 0.0s

history = model3.fit_generator(train_generator, epochs=5, validation_data=val_generator, class_weight=
[30] ✓ 1m 56.5s

Epoch 1/5
C:\Users\sjkom\AppData\Local\Temp\ipykernel_43684\2485142853.py:1: UserWarning: `Model.fit_generator` is
  history = model3.fit_generator(train_generator, epochs=5, validation_data=val_generator, class_weight=n
103/103 [==============================] - 24s 225ms/step - loss: 1.5052 - accuracy: 0.4732 - val_loss: 0
Epoch 2/5
103/103 [==============================] - 23s 228ms/step - loss: 0.6682 - accuracy: 0.7372 - val_loss: 0
Epoch 3/5
103/103 [==============================] - 23s 225ms/step - loss: 0.3581 - accuracy: 0.8698 - val_loss: 0
Epoch 4/5
103/103 [==============================] - 23s 225ms/step - loss: 0.2861 - accuracy: 0.9185 - val_loss: 0
Epoch 5/5
103/103 [==============================] - 23s 223ms/step - loss: 0.1762 - accuracy: 0.9392 - val_loss: 0

y_pred = model3.predict(X_valid, verbose=1)
y_pred_bool = np.argmax(y_pred, axis=1)

print(classification_report(y_valid, y_pred_bool))

print(confusion_matrix(y_true=y_valid, y_pred=y_pred_bool))
[31] ✓ 1.8s

9/9 [==============================] - 2s 188ms/step
              precision    recall  f1-score   support

           0       1.00      0.80      0.89        30
           1       0.99      0.91      0.95       141
           2       0.85      0.99      0.92       104

    accuracy                           0.93       275
   macro avg       0.95      0.90      0.92       275
weighted avg       0.94      0.93      0.93       275

[[ 24   0   6]
 [  0 129  12]
 [  0   1 103]]
```



CHALLENGING PART

- Most challenging part was working with Keras. It took some time for us to understand which pre-trained model to use.

REFERENCES

- Kermany, Daniel; Zhang, Kang; Goldbaum, Michael (2018), “Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification”, Mendeley Data, V2, doi: 10.17632/rscbjbr9sj.2

  https://data.mendeley.com/datasets/rscbjbr9sj/2

- Shimazaki, A., Ueda, D., Choppin, A. *et al.* Deep learning-based algorithm for lung cancer detection on chest radiographs using the segmentation method. *Sci Rep* **12**, 727 (2022). https://doi.org/10.1038/s41598-021-04667-w
- R. Pandian, V. Vedanarayanan, D.N.S. Ravi Kumar, R. Rajakumar, Detection and classification of lung cancer using CNN and Google net, Measurement: Sensors,Volume 24, 2022, 100588, ISSN 2665-9174, https://doi.org/10.1016/j.measen.2022.100588.