COMPUTER VISION

PNEUMONIA DETECTION CHALLENGE

Capstone Project
Group 1

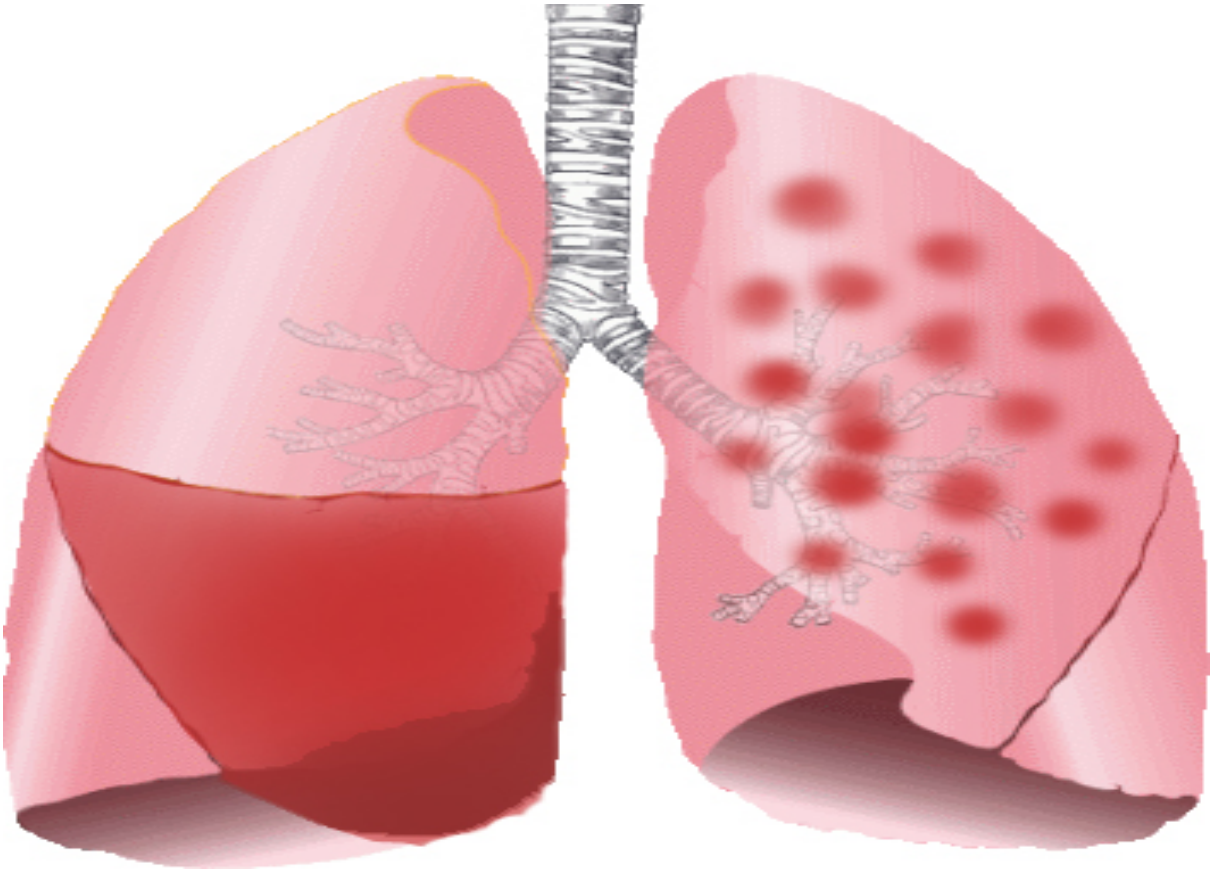
Team

Jyant Mahara	**Capstone Project Mentor**
Sneh Sweta	Capstone Group 1 Team
Harsha Bhat	Capstone Group 1 Team
Renjini Jayakumar	Capstone Group 1 Team
Shivee	Capstone Group 1 Team
Revathy Jayakumar	Capstone Group 1 Team

Table of Contents

Executive Summary:

Pneumonia is a life-threatening lung infection that affects millions of people worldwide. Traditional pneumonia diagnosis methods can be time-consuming and expensive. Artificial Intelligence (AI) has the potential to revolutionize pneumonia detection by enabling accurate and timely diagnoses.



AI algorithms can analyze chest X-rays and CT scans to identify patterns and anomalies associated with pneumonia. These algorithms can detect pneumonia with high accuracy, potentially reducing the need for invasive and expensive diagnostic procedures.

AI-powered pneumonia detection systems can also help healthcare professionals prioritize treatment for patients with severe cases of pneumonia, ensuring that they receive timely and appropriate care.

However, AI-powered pneumonia detection systems are not without their challenges. One major concern is the potential for bias in AI algorithms, which could lead to misdiagnosis or underdiagnosis of certain patient populations. Addressing these concerns will be crucial for ensuring the safe and effective use of AI in pneumonia diagnosis.

Overall, AI-powered pneumonia detection systems have the potential to improve the accuracy and speed of diagnosis. AI systems can analyze medical images and patient data to identify patterns and make predictions, allowing for more accurate diagnosis and treatment. This technology can also help healthcare professionals to make more informed decisions about patient care, reducing the risk of misdiagnosis and improving patient outcomes. However, there are still challenges to be addressed, such as the need for high-quality data and the need for AI algorithms to be validated in clinical settings. With continued development and refinement, AI- based pneumonia detection has the potential to revolutionize the field of medical imaging and improve the care of patients with respiratory infections.

PROBLEM STATEMENT

DOMAIN: Health Care

CONTEXT: Computer vision can be used in health care for identifying diseases. In Pneumonia detection we need to detect Inflammation of the lungs. In this challenge, you're required to build an algorithm to detect a visual signal for pneumonia in medical images. Specifically, your algorithm needs to automatically locate lung opacities on chest radiographs.

DATA DESCRIPTION: The goal is to build a computer vision algorithm to detect pneumonia by identifying lung opacities in chest radiograph images. The dataset includes medical images in DICOM format, containing both pixel data and metadata. The classification task involves three classes:

- Pneumonia - Lung opacities indicating pneumonia.
- Not Normal No Lung Opacity - Abnormalities without pneumonia that might resemble pneumonia in appearance. This extra third class indicates that while pneumonia was determined not to be present, there was nonetheless some type of abnormality on the image and oftentimes this finding may mimic the appearance of true pneumonia.
- Normal - Healthy cases with no visible abnormalities.

The challenge is to automatically locate and identify lung opacities accurately, distinguishing true pneumonia cases from other abnormalities to assist in diagnosis.

Dicom original images: - Medical images are stored in a special format called DICOM files (*.dcm).

You can refer to the details of the dataset in the above link – Acknowledgements: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/overview/acknowledgements.

EDA and Pre-processing

Our approach is to have the following steps:

1. Data Collection

The dataset contains the following files and folders:

1. stage_2_train_labels.csv

Attributes:

- patientId: A unique identifier for each patient. This is used to group related data points and link them to the same individual.
- x, y, height, width: These coordinates represent the bounding box information for the lung opacity regions. The bounding box outlines the suspected areas of pneumonia in the medical images.
    - x: The x-coordinate of the upper-left corner of the bounding box.
    - y: The y-coordinate of the upper-left corner.
    - height: The vertical size of the bounding box.
    - width: The horizontal size of the bounding box.
- target: The dependent variable, indicating the presence or absence of lung opacity.
    - 0: No lung opacity (normal or other abnormalities not mimicking pneumonia).
    - 1: Lung opacity present, which may indicate pneumonia.

2. stage_2_detailed_class_info.csv

Attributes:

- patientId: A unique identifier for each patient. This is used to group related data points and link them to the same individual.
- class: Class label which mentions about the patient's Lung condition - No Lung Opacity/Not Normal , Lung Opacity, Normal (3 classes are there)

Apart from the above-mentioned data files (in csv format), the dataset also contains the images folders
stage_2_train_images
stage_2_test_images

The images in the above-mentioned folders are stored in a special format called DICOM files (*.dcm). They contain a combination of header metadata as well as underlying raw image arrays for pixel data.

2. Load and read the dataset:

Check Shape of the dataset

```
[4] train_labels = pd.read_csv('/content/drive/MyDrive/stage_2_train_labels.csv')
    print('First five rows of Training set:\n', train_labels.head())
```

```
First five rows of Training set:
                                  patientId      x      y  width  height  Target
0  0004cfab-14fd-4e49-80ba-63a80b6bddd6    NaN    NaN    NaN     NaN       0
1  00313ee0-9eaa-42f4-b0ab-c148ed3241cd    NaN    NaN    NaN     NaN       0
2  00322d4d-1c29-4943-afc9-b6754be640eb    NaN    NaN    NaN     NaN       0
3  003d8fa0-6bf1-40ed-b54c-ac657f8495c5    NaN    NaN    NaN     NaN       0
4  00436515-870c-4b36-a041-de91049b9ab4  264.0  152.0  213.0   379.0       1
```

Shape of dataset:

```
[8] print(f'Class labels dataframe has {class_labels.shape[0]} rows and {class_labels.shape[1]} columns')
```

```
Class labels dataframe has 30227 rows and 2 columns
```

3. Data Cleaning:

Check for Missing Values

```
# Checking nulls in class_labels:
print('Number of nulls in class columns: {}'.format(class_labels['class'].isnull().sum()))
```

```
Number of nulls in class columns: 0
```

+ Code + Text

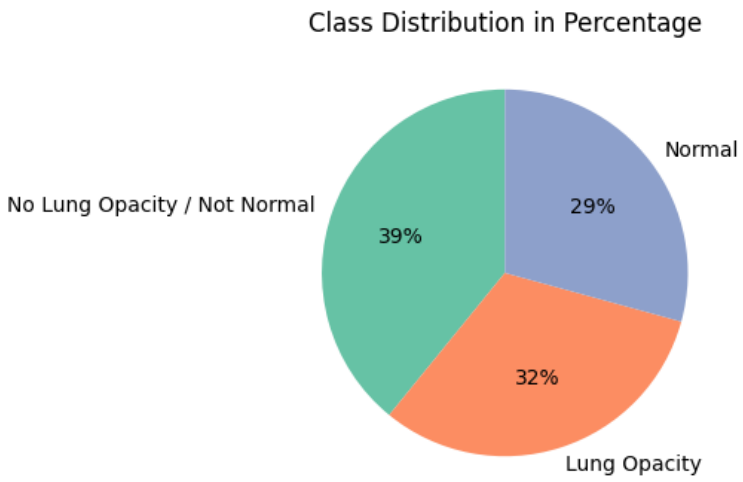
Check for Duplicate Value

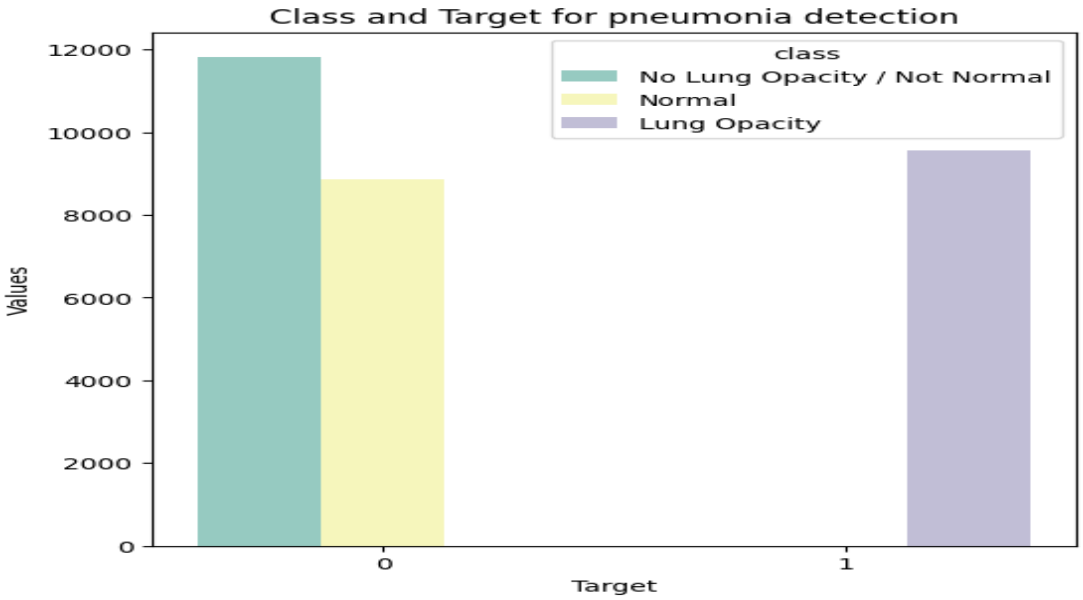
```
[9] # Number of duplicates in patients:

    print('Number of duplicates in patientID in class labels dataframe: {}'.format(len(class_labels) - (class_labels['patientId'].nunique())))
    print('Number of unique patientId are: {}'.format(class_labels['patientId'].nunique()))
```

```
Number of duplicates in patientID in class labels dataframe: 3543
Number of unique patientId are: 26684
```

4. Feature Analysis and Class Distribution

a. Class Distribution





To handle and analyze this data, we can break it down as follows:

1. **Data Overview:**

   We have a dataset with:

   Patient IDs (some of which are duplicated due to multiple bounding boxes per patient).

   Bounding box coordinates (defined by x, y, width, height).

   A binary Target column, where:

   Target = 1 indicates a finding of pneumonia.

   Target = 0 indicates no definitive evidence of pneumonia.

2. **Class Labels:**

   The dataset also has a class label with three possible values:

   Lung Opacity: Indicates evidence of pneumonia (corresponds to Target = 1).

   Normal or No Lung Opacity / Not Normal: Corresponds to Target = 0.

3. **Distribution of Bounding Boxes:**

   A significant number of patient IDs (23,286 or 87%) have only one bounding box.

   A few cases (13 patients) have as many as four bounding boxes, which may represent multiple findings within the same image.

5. Merge Data Sets

Merge the two data frames with labels and class dataset.

```
# Merging the two dataset - 'train_labels' and 'class_labels':


merge_data = pd.concat([train_labels, class_labels['class']], axis = 1)
print("Merged Dataset:")
merge_data.head()
```

Merged Dataset:

	patientId	x	y	width	height	Target	number_of_boxes	class
0	0004cfab-14fd-4e49-80ba-63a80b6bddd6	NaN	NaN	NaN	NaN	0	1	No Lung Opacity / Not Normal
1	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	NaN	NaN	NaN	NaN	0	1	No Lung Opacity / Not Normal
2	00322d4d-1c29-4943-afc9-b6754be640eb	NaN	NaN	NaN	NaN	0	1	No Lung Opacity / Not Normal
3	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	NaN	NaN	NaN	NaN	0	1	Normal
4	00436515-870c-4b36-a041-de91049b9ab4	264.0	152.0	213.0	379.0	1	2	Lung Opacity

```
print('Shape of the dataset: {}'.format(merge_data.shape))
```

```
Shape of the dataset: (30227, 8)
```

Map Training and Testing Images to Their Classes:

```
# Step 2: Map Training and Testing Images to Their Classes
# Map patient IDs to their respective classes
image_classes = dict(zip(class_labels['patientId'], class_labels['class']))

# Step 2: Display sample mappings to verify
print("Training and testing images mapped to their classes.\n")
print("Sample mappings:")

# Show the first 5 mappings as a sample
sample_mappings = list(image_classes.items())[:5]
for patient_id, class_label in sample_mappings:
    print(f"Patient ID: {patient_id} => Class: {class_label}")
```

```
Training and testing images mapped to their classes.

Sample mappings:
Patient ID: 0004cfab-14fd-4e49-80ba-63a80b6bddd6 => Class: No Lung Opacity / Not Normal
Patient ID: 00313ee0-9eaa-42f4-b0ab-c148ed3241cd => Class: No Lung Opacity / Not Normal
Patient ID: 00322d4d-1c29-4943-afc9-b6754be640eb => Class: No Lung Opacity / Not Normal
Patient ID: 003d8fa0-6bf1-40ed-b54c-ac657f8495c5 => Class: Normal
Patient ID: 00436515-870c-4b36-a041-de91049b9ab4 => Class: Lung Opacity
```

6. Dealing with DICOM Images:

DICOM images are typically used for storing medical images and are rich in information. Alongside the image data, they include important patient details such as the patient's name, age, sex, and the physician's name, among others. To preview DICOM images without extracting any information, you can utilize the following code. First, make sure to install the pydicom Python package by running pip install pydicom.

```
import pydicom
sample_patientId = train_labels['patientId'][0]
dcm_file = '/content/drive/MyDrive/zipOut/stage_2_train_images/'+'{}.dcm'.format(sample_patientId)
dcm_data = pydicom.dcmread(dcm_file)

print('Metadata of the image consists of \n', dcm_data)
```

Here is the META Data from sample DICOM image file

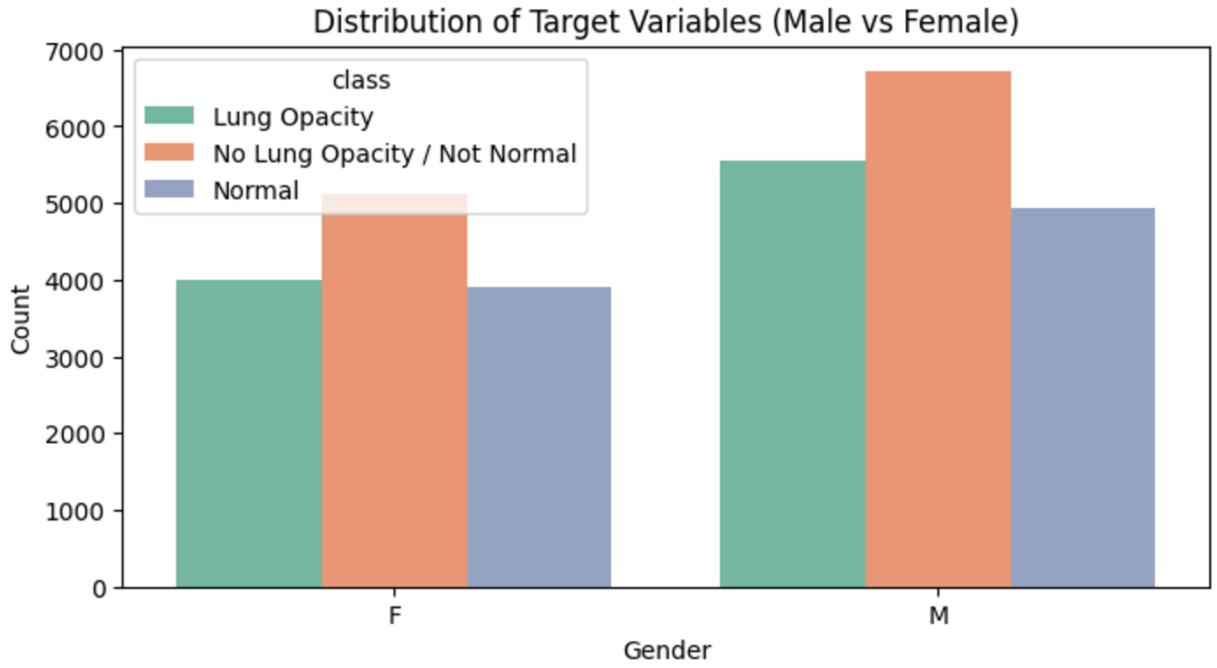
```
Dataset.file_meta -------------------------------
(0002, 0000) File Meta Information Group Length  UL: 202
(0002, 0001) File Meta Information Version       OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID         UI: Secondary Capture Image Storage
(0002, 0003) Media Storage SOP Instance UID      UI: 1.2.276.0.7230010.3.1.4.8323329.28530.1517874485.775526
(0002, 0010) Transfer Syntax UID                 UI: JPEG Baseline (Process 1)
(0002, 0012) Implementation Class UID            UI: 1.2.276.0.7230010.3.0.3.6.0
(0002, 0013) Implementation Version Name         SH: 'OFFIS_DCMTK_360'
-------------------------------------------------
(0008, 0005) Specific Character Set              CS: 'ISO_IR 100'
(0008, 0016) SOP Class UID                       UI: Secondary Capture Image Storage
(0008, 0018) SOP Instance UID                    UI: 1.2.276.0.7230010.3.1.4.8323329.28530.1517874485.775526
(0008, 0020) Study Date                          DA: '19010101'
(0008, 0030) Study Time                          TM: '000000.00'
(0008, 0050) Accession Number                    SH: ''
(0008, 0060) Modality                            CS: 'CR'
(0008, 0064) Conversion Type                     CS: 'WSD'
(0008, 0090) Referring Physician's Name          PN: ''
(0008, 103e) Series Description                  LO: 'view: PA'
(0010, 0010) Patient's Name                      PN: '0004cfab-14fd-4e49-80ba-63a80b6bddd6'
(0010, 0020) Patient ID                          LO: '0004cfab-14fd-4e49-80ba-63a80b6bddd6'
(0010, 0030) Patient's Birth Date                DA: ''
(0010, 0040) Patient's Sex                       CS: 'F'
(0010, 1010) Patient's Age                       AS: '51'
(0018, 0015) Body Part Examined                  CS: 'CHEST'
(0018, 5101) View Position                       CS: 'PA'
(0020, 000d) Study Instance UID                  UI: 1.2.276.0.7230010.3.1.2.8323329.28530.1517874485.775525
(0020, 000e) Series Instance UID                 UI: 1.2.276.0.7230010.3.1.3.8323329.28530.1517874485.775524
(0020, 0010) Study ID                            SH: ''
(0020, 0011) Series Number                       IS: '1'
(0020, 0013) Instance Number                     IS: '1'
(0020, 0020) Patient Orientation                 CS: ''
(0028, 0002) Samples per Pixel                   US: 1
(0028, 0004) Photometric Interpretation          CS: 'MONOCHROME2'
(0028, 0010) Rows                                US: 1024
(0028, 0011) Columns                             US: 1024
(0028, 0030) Pixel Spacing                       DS: [0.14300000000000002, 0.14300000000000002]
(0028, 0100) Bits Allocated                      US: 8
(0028, 0101) Bits Stored                         US: 8
(0028, 0102) High Bit                            US: 7
(0028, 0103) Pixel Representation                US: 0
(0028, 2110) Lossy Image Compression             CS: '01'
(0028, 2114) Lossy Image Compression Method      CS: 'ISO_10918_1'
(7fe0, 0010) Pixel Data                          OB: Array of 142006 elements
```

7. Visualize Data

Distribution of Target Variables Male vs Female:

Within Male and Female group, the rate of Pneumonia risk is more in Male compared to Female

```
Distribution of Target variable by Gender:
class       Lung Opacity  No Lung Opacity / Not Normal  Normal
PatientSex
F                   3995                          5111    3905
M                   5560                          6710    4946
```

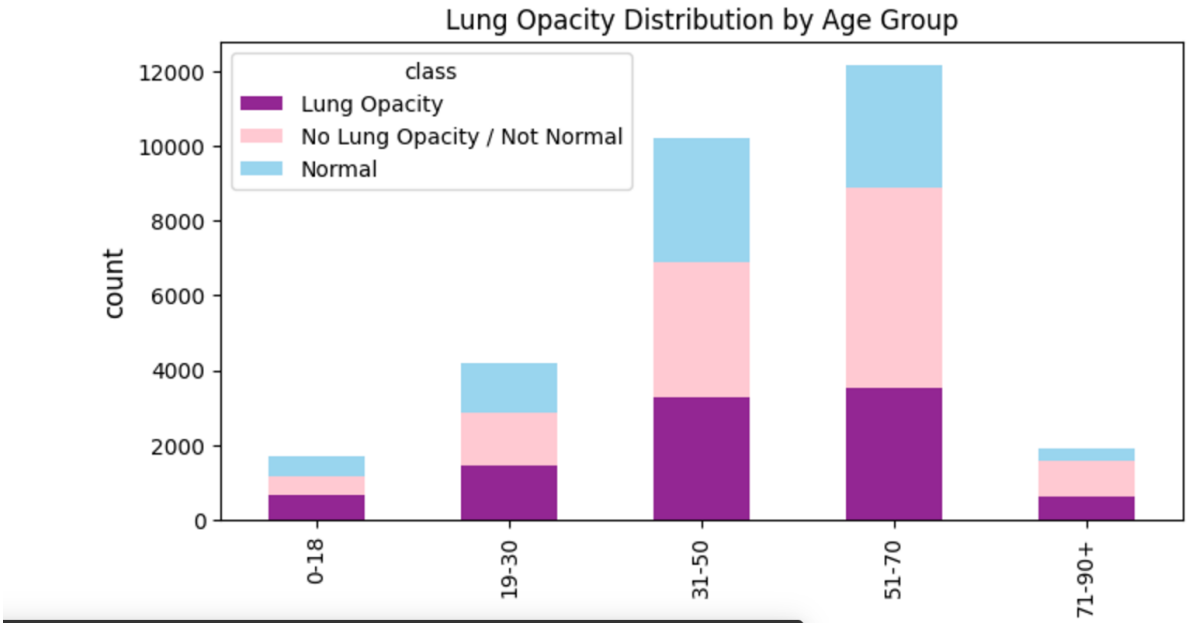


Distribution of Age vs Lung Opacity:

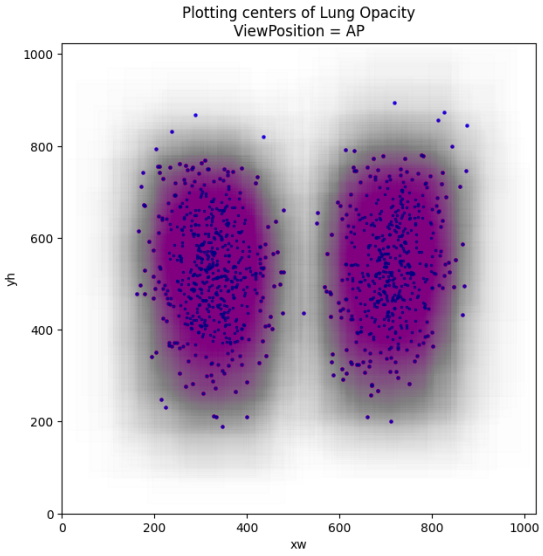
- Lung opacity is more common in older age groups, particularly in the 51-70 age range, suggesting an age-related increase in prevalence or risk.
- Younger age groups (0-18 and 19-30) are predominantly "Normal," indicating lower susceptibility or fewer cases of lung opacity.
- As age increases, the proportion of "Lung Opacity" cases grows, while the proportion of "Normal" cases decreases.
- The "No Lung Opacity / Not Normal" class remains relatively stable across all age groups.

- Healthcare Implications: Screening and preventative measures should target middle-aged and older adults (31-70+) due to higher prevalence of lung opacity in these groups.

```
Distribution of Age vs Lung Opacity:
class     Lung Opacity  No Lung Opacity / Not Normal  Normal
AgeGroup
0–18               678                           490     530
19–30             1451                          1395    1363
31–50             3266                          3631    3325
51–70             3539                          5361    3276
71–90+             621                           944     357
```



Centers of Lung Opacity AP vs PA:

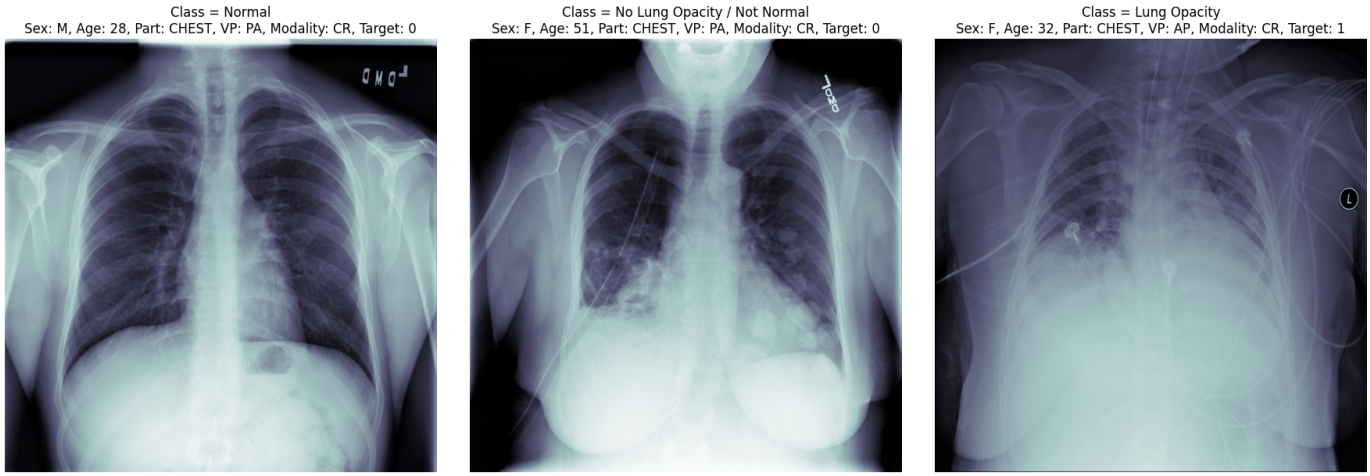


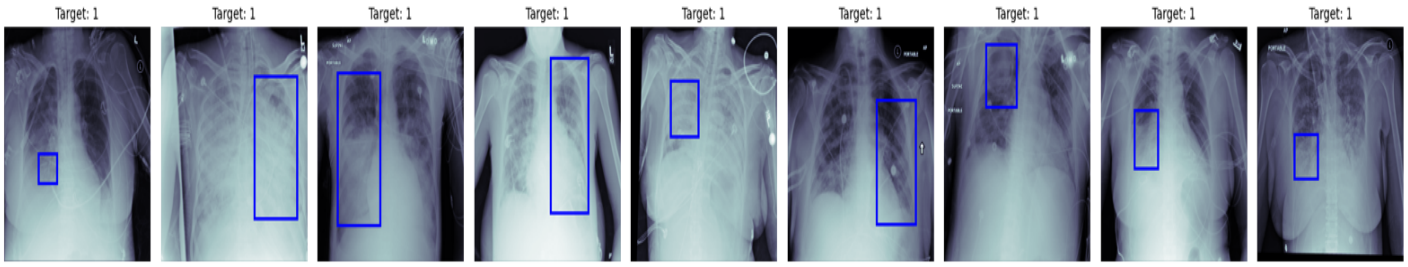
- BodyPart examined is unique for all cases and is CHEST in the training dataset and that was also expected.
- Unique in Modality is CR i.e. Computer Radiography
- Overall ViewPosition is almost equally distributed in the training dataset but for cases where. Target=1, most of the view position are AP.

8. Sample Images:

Here are some sample images from various categories, including Normal, Lung Opacity, and No Lung Opacity/Normal. For the objectives of this project, we will focus solely on predictions related to Normal and Lung Opacity.

Additionally, here are some random images from the training dataset. For the images depicting lung opacity, we have included bounding boxes to highlight the affected areas.


Class = Normal
Sex: M, Age: 28, Part: CHEST, VP: PA, Modality: CR, Target: 0
Class = No Lung Opacity / Not Normal
Sex: F, Age: 51, Part: CHEST, VP: PA, Modality: CR, Target: 0
Class = Lung Opacity
Sex: F, Age: 32, Part: CHEST, VP: AP, Modality: CR, Target: 1


Target: 1
Target: 1
Target: 1
Target: 1
Target: 1
Target: 1
Target: 1
Target: 1
Target: 1

Model and Model Building:

1. Model Architecture:

- **Input Dimensions:**
  - Input Shape: (128, 128, 3) (image size is 128×128128 \times 128128×128 with RGB channels).
  - Output Classes: 3 (classification targets).
- **Architecture Overview:**
  - **Convolutional Layers:**
    - 3 convolutional layers with ReLU activation:
      - First layer: 323232 filters of size 3×33 \times 33×3.
      - Second layer: 646464 filters of size 3×33 \times 33×3.
      - Third layer: 128128128 filters of size 3×33 \times 33×3.
    - Each convolutional layer is followed by:
      - **Batch Normalization**: Stabilizes and accelerates training.
      - **MaxPooling**: Down samples feature maps by 2×22 \times 22×2.
  - **Flatten Layer:**
    - Converts the 3D output from the convolutional layers into a 1D vector.
  - **Dense (Fully Connected) Layers:**
    - Two fully connected layers with 128128128 neurons and ReLU activation.
    - Dropout (50%): Applied to both dense layers to mitigate overfitting.
  - **Output Layer:**
    - Dense layer with 333 neurons (for the 3 classes) and Softmax activation to output class probabilities.
- **Compilation Details:**
  - Optimizer: **Adam** (adaptive learning rate optimization).
  - Loss Function: **Categorical Crossentropy** (suitable for multi-class classification).
  - Metric: **Accuracy**.

2. Model Summary:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
batch_normalization (BatchNormalization)	(None, 126, 126, 32)	128
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 61, 61, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 28, 28, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_3 (Conv2D)	(None, 12, 12, 256)	295,168
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 256)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 256)	2,359,552
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32,896
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 3)	387

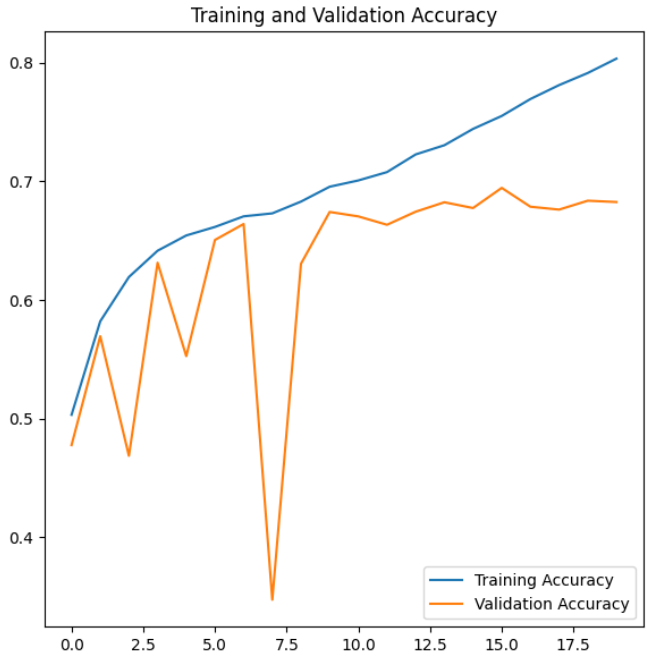
Total params: 2,782,147 (10.61 MB)
Trainable params: 2,781,699 (10.61 MB)
Non-trainable params: 448 (1.75 KB)

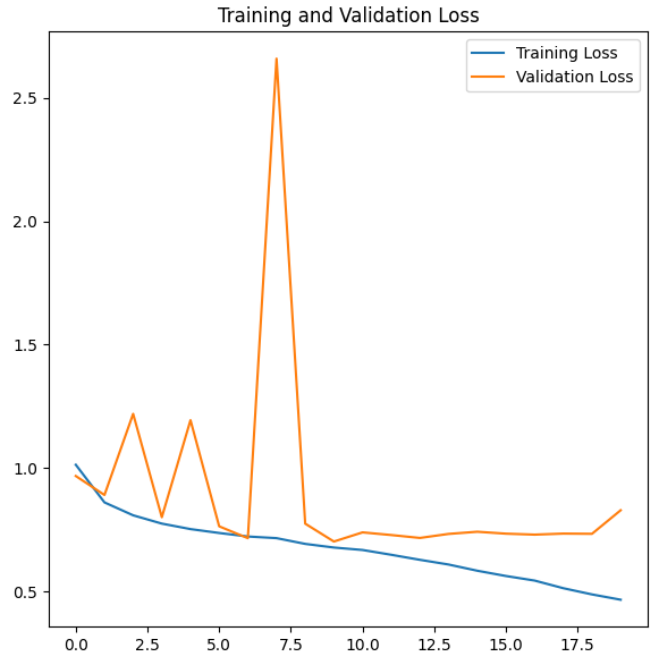
Strengths of this Model:

- Batch Normalization: Enhances model stability and performance by normalizing layer inputs.
- Dropout Layers: Reduces overfitting, especially with the higher dropout rate (50%50\%50%).
- Flexible Design: Modularized to accept any input dimensions and number of classes.

3. Validation Loss and Validation Accuracy:

The following graph represent the validation loss and validation accuracy for the improved model.





Training and Validation Accuracy:

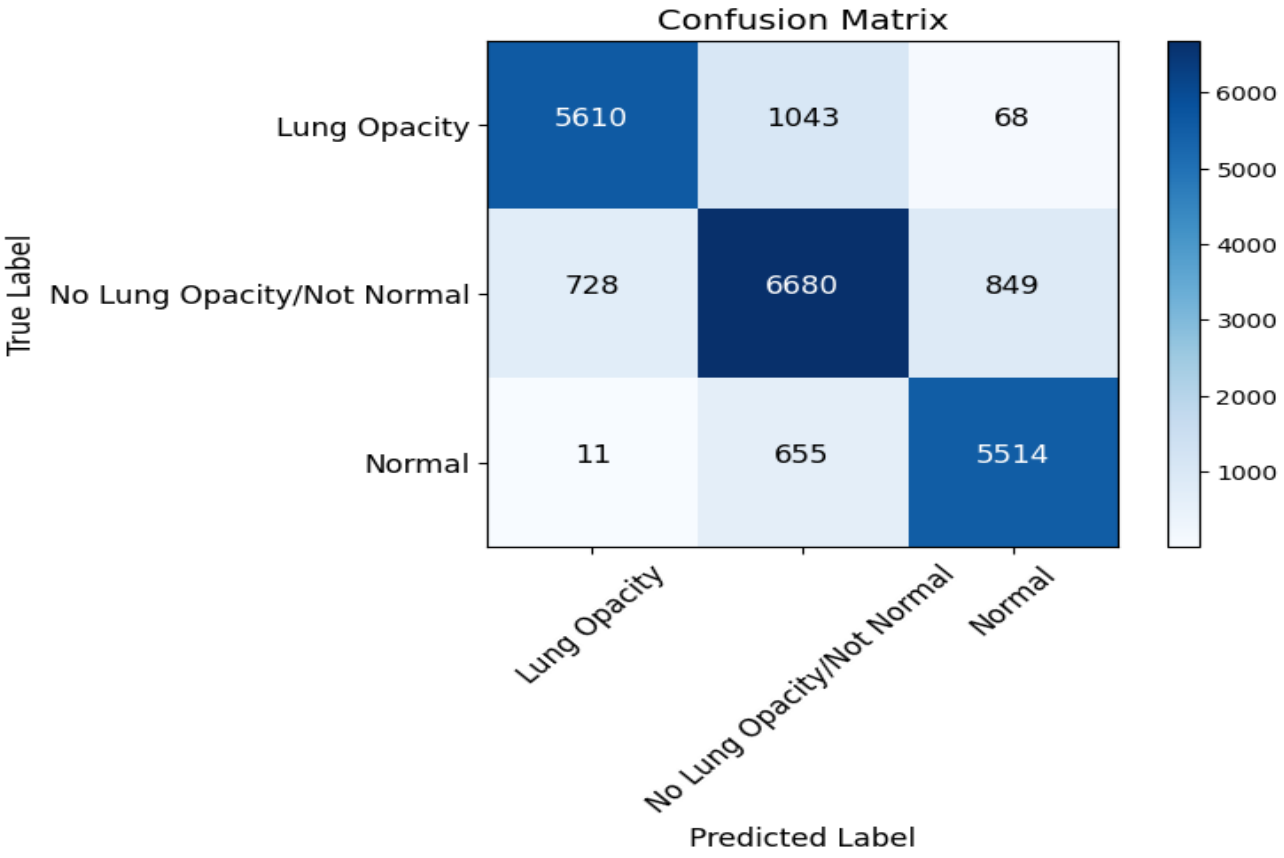
- Observation:
  - Training accuracy is steadily increasing over the epochs, indicating that the model is learning from the training data effectively.
  - Validation accuracy shows fluctuations but stabilizes at a slightly lower level compared to training accuracy. This may point to mild overfitting or the need for additional regularization techniques.

Training and Validation Loss:

- Observation:
  - Training loss is gradually decreasing, reflecting consistent improvement in the model's ability to minimize error on the training set.
  - Validation loss decreases initially but shows some large spikes and fluctuations, which could indicate:
    - Unstable learning: The model might struggle to generalize.
    - Overfitting: Validation loss diverging from training loss at later epochs.

4. Confusion Matrix:

Train set:



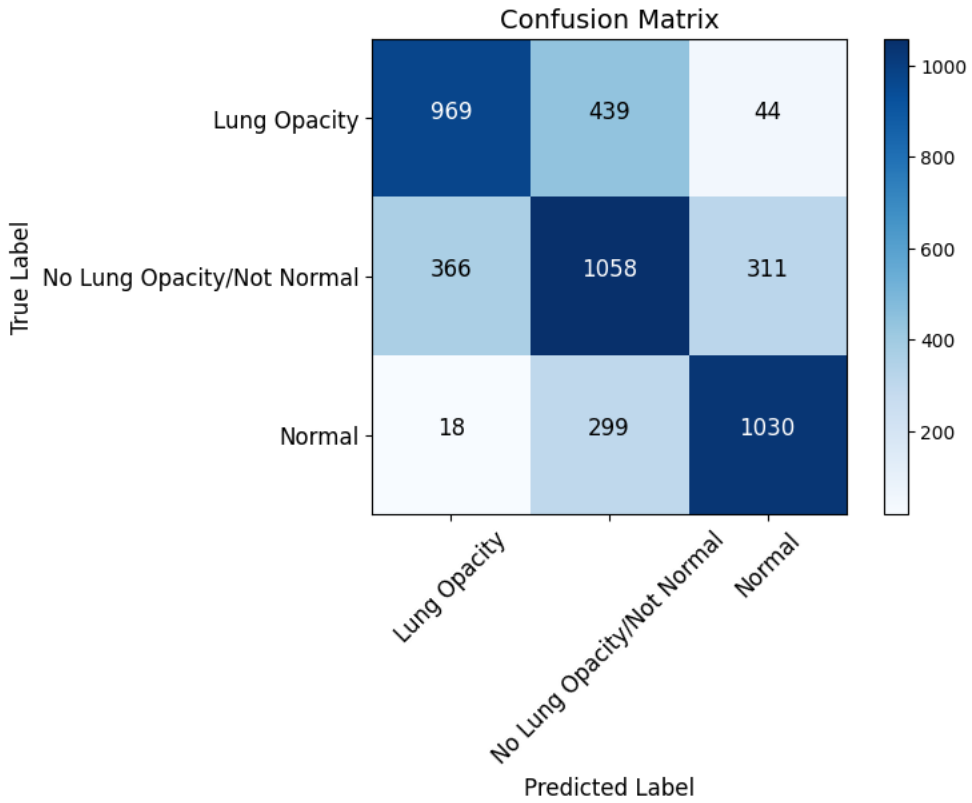
Classification Report:

	precision	recall	f1-score	support
Lung Opacity	0.88	0.83	0.86	6721
No Lung Opacity/Not Normal	0.80	0.81	0.80	8257
Normal	0.86	0.89	0.87	6180
accuracy			0.84	21158
macro avg	0.85	0.85	0.85	21158
weighted avg	0.84	0.84	0.84	21158

Observations:

- The No Lung Opacity/Not Normal class shows slightly lower precision and recall compared to the others. This could be due to:
  - Overlap or similarity between the No Lung Opacity/Not Normal and other classes, leading to confusion.
  - Potential class imbalance or inherent complexity in this label.
- Normal and Lung Opacity classes are performing well, with higher precision and recall, indicating good differentiation by the model.

Test Set:



Observations and Insights:

1. Highest Confusion:
   - The model struggles most between Lung Opacity and No Lung Opacity/Not Normal, with a significant number of samples being misclassified in both directions (439 and 366, respectively).
   - There is moderate confusion between No Lung Opacity/Not Normal and

Normal (311 samples misclassified).

2. Normal Class:
    - The Normal class has the highest accuracy, with relatively fewer misclassifications.
3. Class Imbalance or Ambiguity:
    - The high confusion between Lung Opacity and No Lung Opacity/Not Normal might indicate overlapping features between these classes or an imbalance in the training data.

```
Classification Report:

                           precision    recall  f1-score   support

              Lung Opacity       0.72      0.67      0.69      1452
No Lung Opacity/Not Normal       0.59      0.61      0.60      1735
                    Normal       0.74      0.76      0.75      1347

                  accuracy                           0.67      4534
                 macro avg       0.68      0.68      0.68      4534
              weighted avg       0.68      0.67      0.67      4534
```

Observations

1. Performance :
    - The "Normal" class performs the best, with the highest F1-score (0.75).
    - The "No Lung Opacity/Not Normal" class performs the worst, with the lowest F1-score (0.60). This aligns with confusion observed in the confusion matrix.
2. Precision vs Recall:
    - For "Lung Opacity" and "Normal," precision and recall are balanced.
    - For "No Lung Opacity/Not Normal," precision is slightly lower than recall, indicating false positives are more common.

3. Class Support:
    - The "No Lung Opacity/Not Normal" class has the highest support (1735 samples), but its performance metrics are the weakest, suggesting potential challenges in feature extraction or data quality for this class.

Strategies to Improve Model Performance

1. Data Preparation

- Diversify the Dataset: Use data augmentation techniques like flipping, rotating, scaling, or cropping to increase variability without collecting new samples.
- Normalize and Scale: Ensure data consistency by scaling image pixel values to a range like [0, 1] or applying normalization with mean-zero and unit variance.

2. Enhance Model Architecture

- Optimize Network Design: Experiment with increasing or reducing the number of layers and neurons to find the optimal balance.
- Incorporate Regularization:
    - Add dropout layers to mitigate overfitting.
    - Use weight regularization (e.g., L1/L2 penalties) to control model complexity.

3. Refine Optimization Techniques

- Tune Learning Rates:
    - Use learning rate schedulers, such as ReduceLROnPlateau or cosine annealing, for dynamic adjustment.
    - Implement warm-up strategies by starting with a low learning rate and gradually increasing it.

- Choose Effective Optimizers: Test optimizers like Adam, RMSprop, SGD with momentum, or AdamW to identify the best fit for your task.

4. Hyperparameter Tuning

- Conduct systematic searches using methods like grid search or random search to optimize key parameters.

5. Improve Training Techniques

- Adjust Batch Sizes: Test different batch sizes to find a balance between performance and generalization.
- Shuffle Data: Randomize the training order to eliminate biases from sequential data.
- Apply Early Stopping: Stop training once validation loss stabilizes to avoid overfitting.

6. Leverage Pretrained Models

- Transfer Learning: Fine-tune models trained on large datasets for your specific problem.
- Feature Extraction: Freeze initial layers of a pretrained model and only train the task-specific layers.

7. Monitor and Evaluate Performance

- Use cross-validation to split data into training and validation sets, ensuring your model generalizes well.

8. Debug and Fine-Tune

- Control Gradients: Apply gradient clipping to manage exploding gradients.
- Optimize Weight Initialization: Experiment with methods like Xavier or He initialization.
- Validate Code: Check dataset pipelines, loss functions, and optimizer updates for errors.

9. Use Ensemble Methods

- Combine predictions from multiple models to improve accuracy and robustness.