

**Applied Machine Learning Final Project**

Revolutionizing Healthcare with Machine Learning: Early Detection of Pneumonia

Sandhya Sarampati, Jayanth Kumar Sarampati, Giri Sai Kumar Pragada, Manu Harshith Vadde, Rohit Kumar Mupparapu, Chakri Venkat Katam

**Table of Contents**

1. **Introduction**
   * Overview of the Problem
   * Importance of Early Diagnosis in Healthcare
2. **Motivation**
   * Why Machine Learning in Healthcare?
   * Significance of Applying Machine Learning to Medical Diagnostics
3. **Objective of the Project**
   * Goal of the Project
   * Expected Impact on Healthcare
4. **Data Collection and Preprocessing**
   * Dataset Overview
   * Data Preprocessing
5. **Data Visualization and Augmentation**
   * Chest X-Ray Sample with Label
   * Class Distribution in Training Data
   * Reasons for Data Augmentation
   * Data Augmentation Techniques Across Models
     + Model 1: Basic Augmentation
     + Model 2: Intermediate Augmentation
     + Model 3: Spatial Augmentation
     + Model 4: Comprehensive Augmentation
6. **Model Selection**
   * Experiment 1
   * Experiment 2
   * Experiment 3
   * Experiment 4
7. **Feedback and Adjustments**
   * Model 1: Initial Configuration
   * Model 2: Improved Generalization
   * Model 3: Fine-Tuning for Stability
   * Model 4: Optimizing for Performance
8. **Performance Metrics**
   * Classification Report
   * line graphs
   * Confusion Matrix
9. **Discussion and Insights**
10. **Conclusion and Recommendations**

**1 . Introduction**

## Overview of the Problem

## Pneumonia, a critical lung condition affecting millions worldwide, requires timely detection for effective treatment. Traditional diagnostic methods like X-rays and physical exams are valuable but often limited by the need for expert interpretation, leading to potential delays in treatment.

## Importance of Early Diagnosis in Healthcare

In healthcare, early diagnosis plays a critical role in improving patient outcomes, reducing the severity of diseases, and lowering healthcare costs. With diseases like pneumonia, early intervention can prevent complications and save lives. Timely and accurate diagnosis is especially important in cases where medical resources are limited or access to healthcare professionals is restricted.

## 2. Motivation

## Why Machine Learning in Healthcare?

Machine learning offers transformative potential in healthcare, enabling faster, scalable, and accurate solutions to complex medical challenges. It is particularly impactful in radiology, where it can automate and enhance the interpretation of medical images, minimizing human error and improving efficiency.

## Significance of Applying Machine Learning to Medical Diagnostics

The application of machine learning in medical diagnostics can help reduce diagnostic errors, streamline workflows, and make healthcare more accessible. By training algorithms on large datasets, machine learning models can identify patterns and detect abnormalities that may be overlooked by human doctors. This can lead to earlier detection, improved treatment plans, and better patient outcomes, particularly in areas with limited access to healthcare professionals.

## 3.Objective of the Project

## Goal of the Project

The primary goal of this project is to develop a machine learning based solution for automating the detection of pneumonia from chest Xray images. By leveraging Convolutional Neural Networks (CNNs), the model will aim to accurately identify pneumonia signs, even from low quality images. The project will also focus on improving model performance through data augmentation and enhancement.

## Expected Impact on Healthcare

The expected impact of this project is to create a tool that can assist healthcare professionals in diagnosing pneumonia more quickly and accurately. This model can be integrated into healthcare systems to support early detection, reduce the burden on doctors, and potentially save lives by identifying high risk patients who require urgent care.

**4.Data Collection and Preprocessing**

Dataset Overview

For this project, a publicly available chest X-ray dataset is utilized, comprising labeled images of both pneumonia-positive and pneumonia-negative cases. The dataset’s variety in image quality and sizes makes it ideal for developing and testing machine learning models, ensuring balanced and reliable performance.

Training data shape: (5216, 224, 224, 3)

Testing data shape: (624, 224, 224, 3)

Validation data shape: (16, 224, 224, 3)

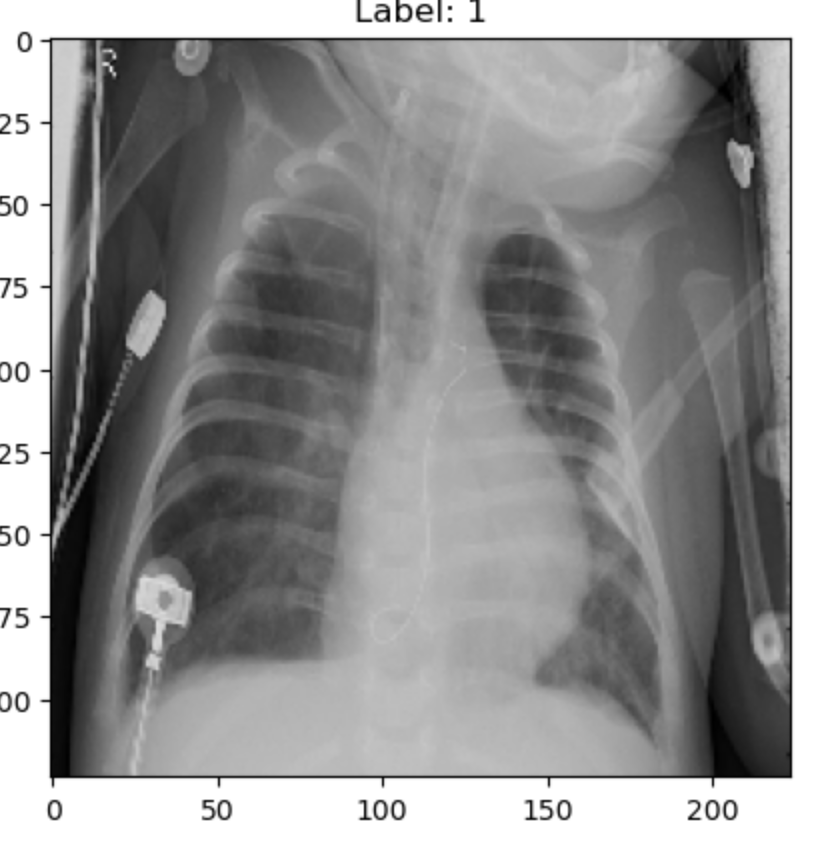
Data Preprocessing

Data preprocessing involves resizing images, normalizing pixel values, and augmenting the dataset using techniques like rotation, flipping, and zooming. These steps ensure that the model learns to recognize pneumonia from a diverse set of X Rays, improving its ability to generalize to new data.

## 5.Data Visualisation and Augmentation:

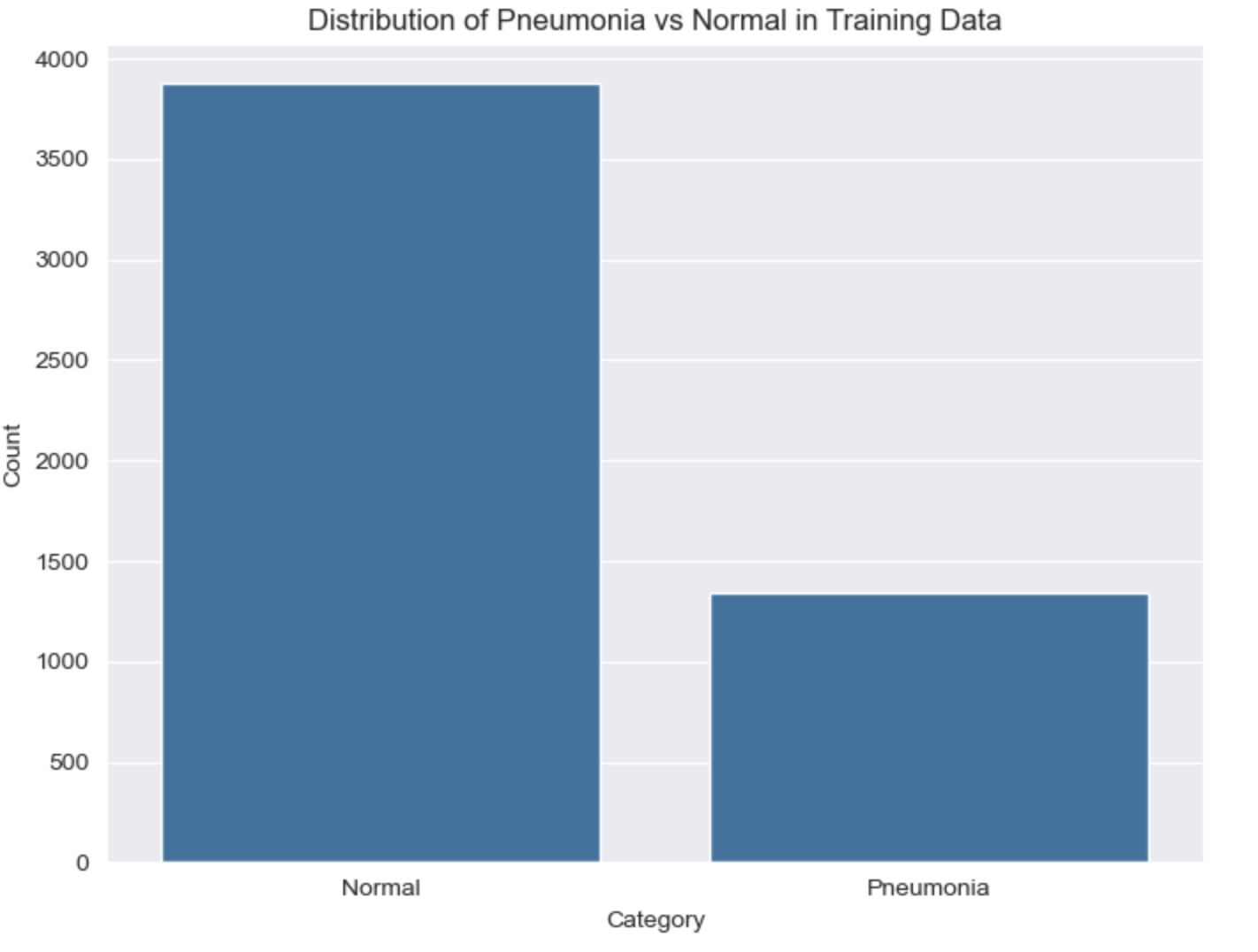
Chest X-Ray Sample with Label

The image displays a labeled chest X-ray indicating pneumonia presence (Label: 1). This serves as a representative sample from the dataset used in training the model for pneumonia detection.



**C**lass Distribution in Training Data

image highlights the distribution of "Normal" and "Pneumonia" cases in the training dataset. The imbalance suggests a higher number of normal cases, underscoring the importance of data augmentation and class balancing to ensure robust model performance.



We are implementing data augmentation techniques to address the imbalance in the dataset, where normal cases significantly outnumber pneumonia cases. This ensures the model is exposed to sufficient examples of minority classes, improving its ability to generalize and accurately detect pneumonia.

## Model 1: Basic Augmentation Data Augmentation Techniques:

* **Rotation:** Rotating images by small random angles (up to 30 degrees).

## Impact:

* The single augmentation technique provided slight variation in the dataset but was insufficient to address overfitting.

## 

## Model 2: Intermediate Augmentation Data Augmentation Techniques:

* **Rotation**: Rotating images by random angles (up to 30 degrees).
* **Zooming:** Random zooming in/out (up to 20%).
* **Rescaling:** Normalizing pixel values to fall between 0 and 1.

## Impact:

* The combination of rotation, zoom, and rescaling reduced overfitting.
* Improved the model’s generalization to unseen data by introducing diversity in orientations and sizes.

## 

## Model 3: Spatial Augmentation Data Augmentation Techniques:

* All techniques from Model 2, plus:
  + Width and Height Shifts: Translating images horizontally (10%) or vertically (10%) to simulate positional variations.

## Impact:

* The inclusion of spatial transformations made the model more resilient to positional changes in input data.
* Improved feature extraction while further reducing overfitting.

## 

## Model 4: Comprehensive Augmentation Data Augmentation Techniques:

* All techniques from Model 3, plus:
  + Horizontal Flipping: Simulating real-world variations by flipping images horizontally.

## Impact:

* The addition of horizontal flipping introduced a final layer of variability, making the model more robust to real-world variations.
* Achieved the best overall performance and stability.

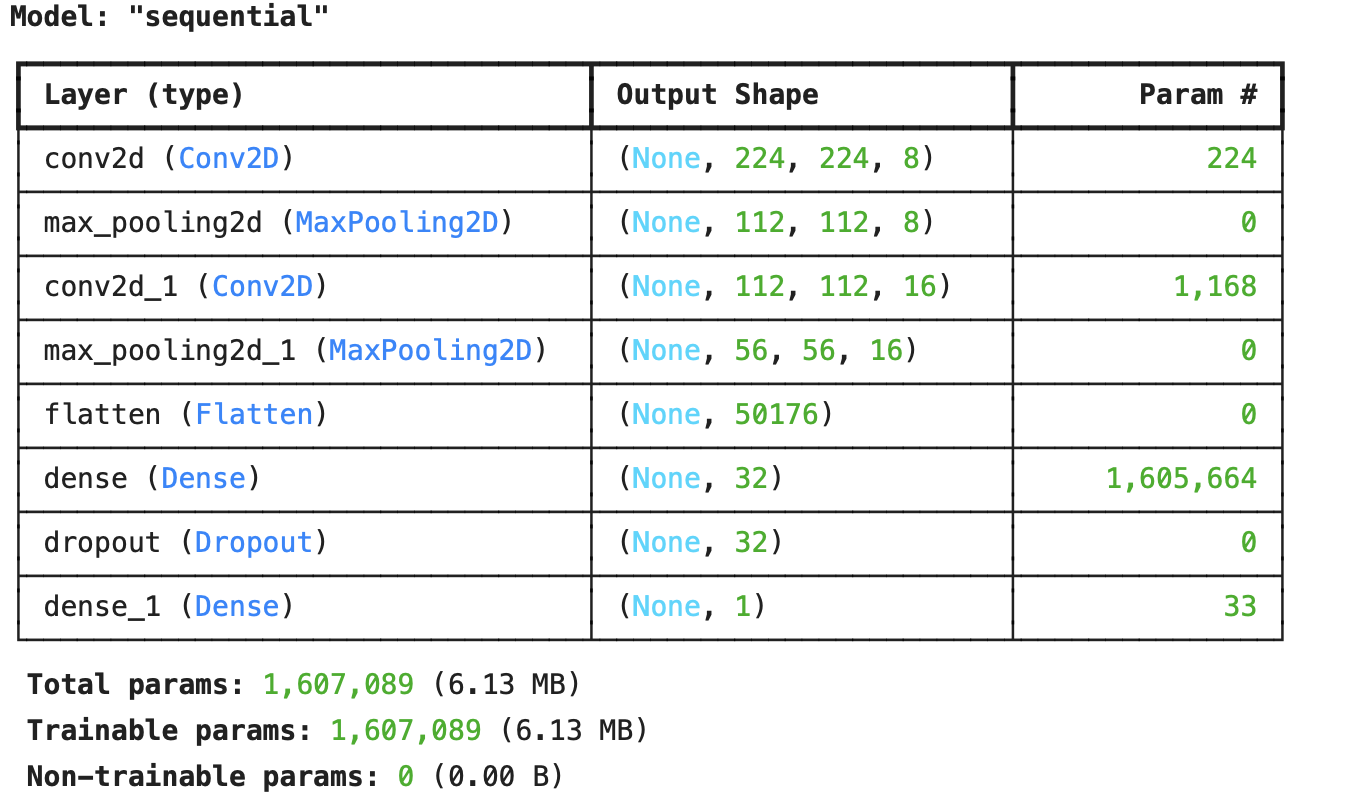
## 

### **Comparison of Data Augmentation Across Models**

| Model | Data Augmentation Techniques |
| --- | --- |
| Model 1 | Rotation |
| Model 2 | Rotation, Zooming, Rescaling |
| Model 3 | All from Model 2 + Width/Height Shifts |
| Model 4 | All from Model 3 + Horizontal Flipping |

**6.Model Selection**

**Experiment:1**



## 1. Conv2D Layers:

* The first convolutional layer uses only 8 filters with a kernel size of (3, 3), which is very few for capturing complex features. The second convolutional layer has 16 filters.
* Both layers use the ReLU activation function and have 'same' padding, ensuring that the output dimensions remain the same as the input.

## 2. MaxPool2D Layers:

* After each convolutional layer, MaxPooling with a (2, 2) pool size is used. This downsampled the feature map, but the use of smaller convolutional layers reduces the richness of features.

## 3. Dense Layer:

* A single Dense layer with only 32 units is used, which is a small size for fully connected layers. This limits the model's capacity to learn complex representations.

## 4. Dropout:

* A 0.5 dropout rate is applied after the dense layer. While dropout is typically used to prevent overfitting, a high dropout rate (50%) may hurt the model's learning by dropping too much information.

## 5. Optimizer:

* The SGD optimizer (Stochastic Gradient Descent) is used, which may lead to slower convergence compared to optimizers like Adam, especially with smaller model architectures.

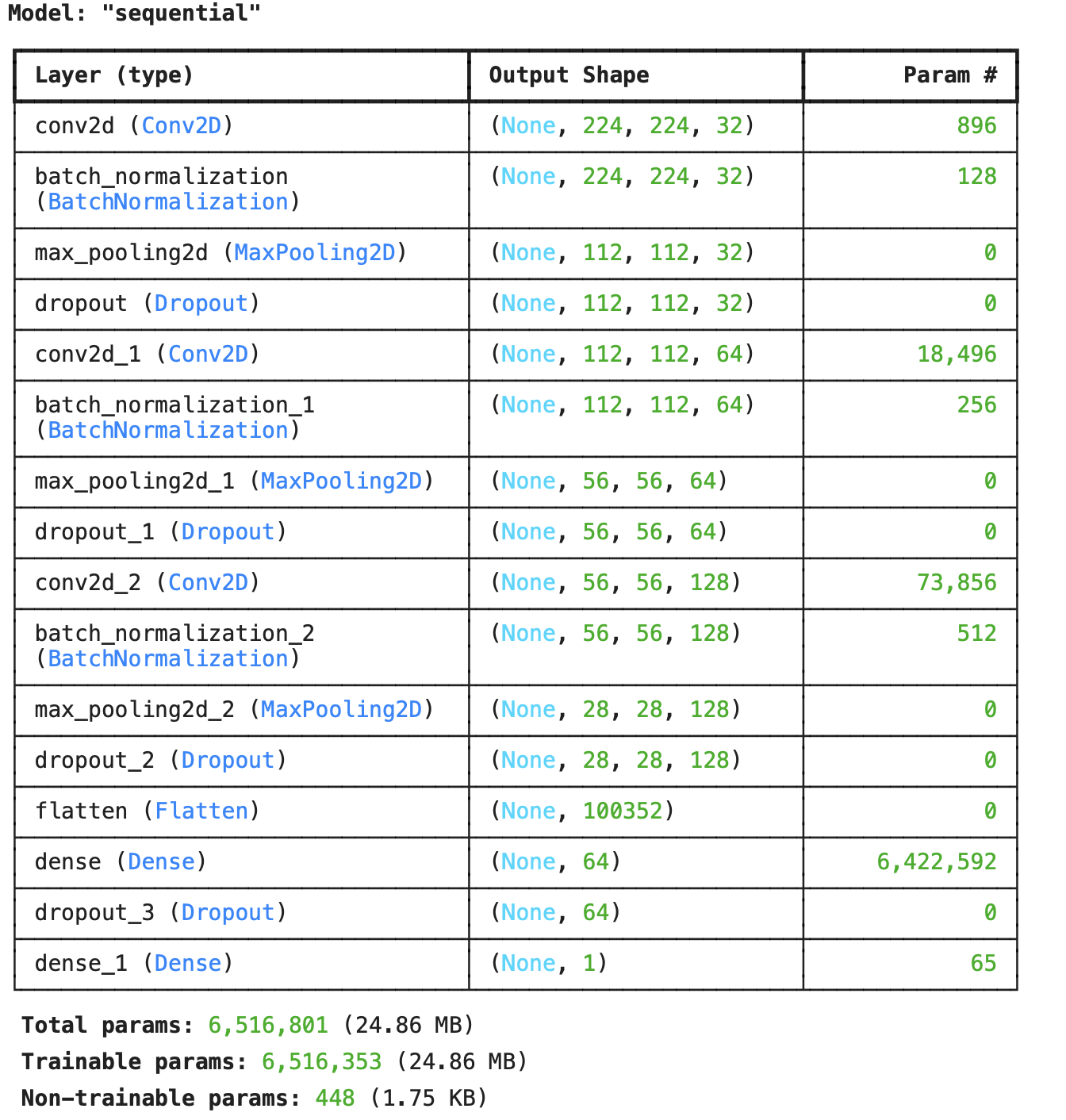
## 6. Loss Function:

* The loss function is binary cross entropy, appropriate for binary classification tasks, indicating that this model is used for a task where there are two possible outcomes (e.g., pneumonia detection).

## 7. Activation Functions:

* ReLU is used for the convolutional and dense layers, while sigmoid is used in the output layer for binary classification, squashing outputs to a probability between 0 and 1.

**Experiment:2**



This model is designed with a more advanced architecture and additional regularization techniques to improve its performance and generalization.

## 1. Conv2D Layers:

* 32 filters in the first convolutional layer, 64 filters in the second, and 128 filters in the third convolutional layer. This increasing number of filters allows the model to capture more complex features at each subsequent layer.
* Each convolutional layer uses a (3, 3) kernel size, which is commonly used for image processing tasks.
* ReLU activation is used after each convolutional layer, which helps introduce nonlinearity, allowing the model to learn complex patterns.

## 2. BatchNormalization:

* BatchNormalization is applied after each convolutional layer to normalize the activations. This helps in accelerating training, reducing internal covariate shift, and improving the overall stability of the model.

## 3. MaxPool2D:

* MaxPooling with a (2, 2) pool size is applied after each convolutional layer. This reduces the spatial dimensions of the feature maps, helping the model become more computationally efficient while retaining important features.

## 4. Dropout:

* Dropout is used after each convolutional and dense layer with rates of 0.3, 0.4, and 0.4 for the convolutional layers, and 0.5 for the dense layer. Dropout helps reduce overfitting by randomly setting a fraction of the input units to zero during training.

## 5. Regularization:

* L2 regularization (also called weight decay) is applied to the convolutional and dense layers with a regularization factor of 0.001. This penalizes large weights and helps prevent overfitting by encouraging the model to learn simpler weights.

## 6. Flatten:

* The Flatten layer is used to convert the 2D output from the last convolutional layer into a 1D vector, which can then be passed to the fully connected (dense) layers.

## 7. Dense Layer:

* The Dense layer with 64 units is followed by a ReLU activation function. This allows the model to learn higher level representations.
* Another Dropout of 0.5 is applied after the dense layer, again to combat overfitting.

## 8. Output Layer:

* The final Dense layer has 1 unit with a sigmoid activation function, making this a binary classification task (e.g., distinguishing between two classes, such as pneumonia or healthy).

## 9. Optimizer:

* Adam optimizer is used, which adapts the learning rate during training, leading to faster convergence compared to traditional gradient descent methods.

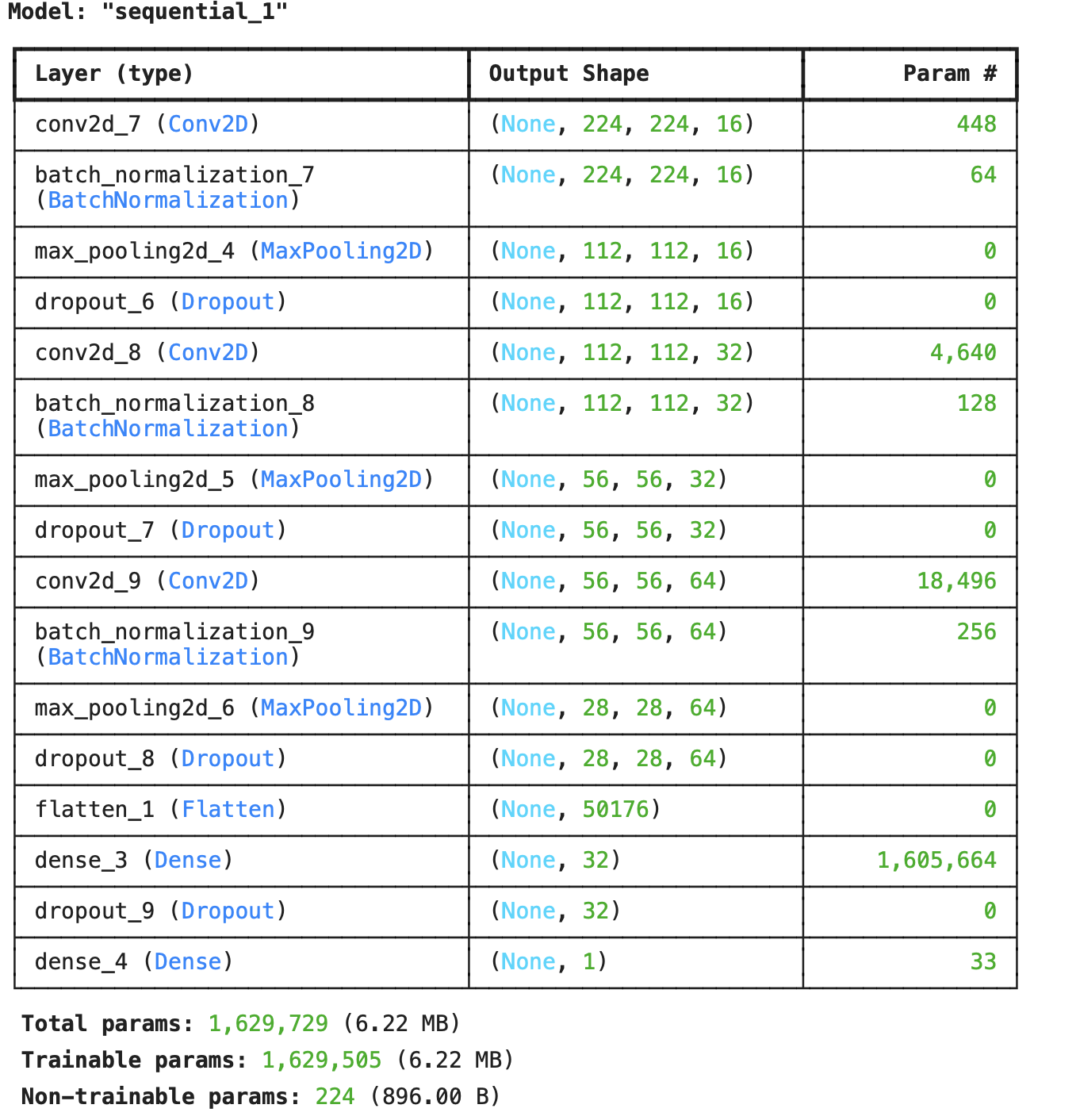
## 10. Loss Function:

* Binary Cross entropy is used as the loss function, which is appropriate for binary classification tasks.

## 11. Metrics:

* Accuracy is used as the evaluation metric, which will provide insights into how well the model performs on the classification task.

**Experiment:3**

****

## 1.Conv2D Layers:

* The first convolutional layer uses 16 filters with a kernel size of (3, 3). The filter count is small but sufficient for initial feature extraction.
* Subsequent layers increase the filter count to 32 and 64, improving the model's capacity to extract richer and hierarchical features.
* All layers use the ReLU activation function for non-linearity and 'same' padding to preserve spatial dimensions of the feature maps.

## 2. MaxPool2D Layers:

* Each convolutional block is followed by MaxPooling with a pool size of (2, 2), effectively reducing spatial dimensions by half. This progressively reduces computational complexity while retaining essential features.

## 3.Dense Layer:

* A single dense layer with 32 units is used, which is moderate in size. While it reduces computational cost, it also limits the model's capacity to learn more complex patterns compared to larger dense layers in similar models.
* An L2 regularization (0.001) is applied, which helps to prevent overfitting by penalizing large weight magnitudes.

## 4.Dropout:

## Dropout is applied after each convolutional block and the dense layer:

* + First block: 0.25
  + Second block: 0.3
  + Third block: 0.3
  + Dense layer: 0.4
* This staggered approach provides regularization, ensuring the model generalizes well without overfitting.

## 5.Optimizer:

* The model uses the Adam optimizer, which adapts learning rates for faster and more stable convergence. This choice is better suited for moderate to complex architectures compared to SGD.

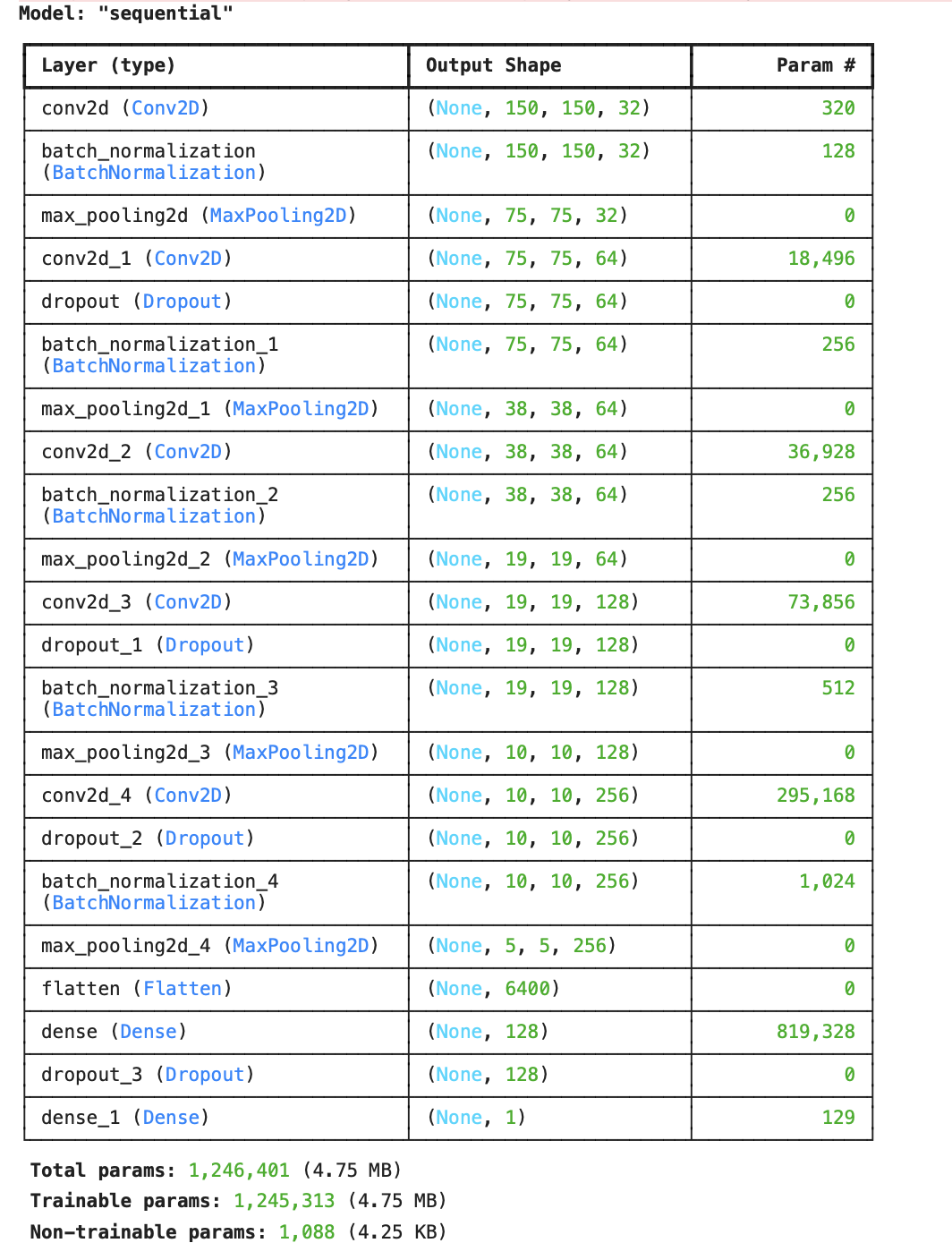
## 6.Loss Function:

* The loss function is binary cross-entropy, appropriate for binary classification tasks. It measures the distance between predicted probabilities and actual binary labels.

## 7.Activation Functions:

* ReLU is used for all convolutional and dense layers to introduce non-linearity and prevent vanishing gradients.
* The output layer uses sigmoid activation, which converts the final output into a probability value between 0 and 1, suitable for binary classification tasks.

**Experiment: 4**

****

## 1.Conv2D Layers:

* + The architecture starts with a **32-filter convolutional layer** with a (3, 3) kernel size and ReLU activation, suitable for extracting initial features from grayscale images (input shape (IMG\_HEIGHT, IMG\_WIDTH, 1)).
  + The filter count increases progressively to **64, 128, and 256 filters**, enhancing the model's ability to capture complex patterns and hierarchical features as depth increases.
  + All convolutional layers use **'same' padding** to preserve spatial dimensions.
  + **Strides are fixed at 1**, ensuring no additional down-sampling occurs in the convolution layers.

## MaxPool2D Layers:

* + Each convolutional block is followed by **MaxPooling with a pool size of (2, 2)** and strides=2, effectively reducing the feature map size by half at each step. This ensures that the computational complexity is managed while retaining essential spatial features.

## Dense Layer:

* + A fully connected layer with **128 units** and ReLU activation is included before the output layer. This allows for a higher-capacity representation compared to smaller dense layers in simpler models.
  + This layer helps bridge the gap between extracted features and the binary classification output.

## Dropout:

## Dropout is applied after several layers to reduce overfitting:

* + - **First block**: 0.1
    - **Fourth block**: 0.2
    - **Fifth block**: 0.2
    - **Dense layer**: 0.2
  + This gradually increasing dropout ensures balanced regularization as model complexity increases.

## BatchNormalization:

* + Batch normalization is applied after every convolutional layer. This stabilizes learning by normalizing inputs to each layer, improving convergence speed and reducing sensitivity to weight initialization.

## Optimizer:

* + The **RMSprop optimizer** is used, which adapts learning rates based on recent gradients, performing well in architectures with varying depth and complexity.

## Loss Function:

* + The model uses **binary cross-entropy**, appropriate for binary classification tasks. This loss function calculates the log loss between predicted probabilities and ground truth labels.

## Activation Functions:

* + **ReLU** is used for all convolutional and dense layers to introduce non-linearity and prevent issues like vanishing gradients.
  + The output layer uses **sigmoid activation**, converting outputs into probabilities between 0 and 1, suitable for binary classification.

## Key Features of the Model:

* + The progressive increase in filters from 32 to 256 allows this architecture to learn features at multiple levels of complexity.
  + The use of **Dropout, BatchNormalization, and RMSprop** ensures better generalization and efficient training.
  + The combination of a **moderately large dense layer and multiple dropout stages** makes this model robust to overfitting while maintaining good representational capacity.

**7.Feedback and Adjustments.**

## 1. Model 1: Initial Configuration

## Initial Observations:

* **Training Accuracy:** 81.08%
* **Validation Accuracy:** 56.2%
* **Training Loss:** 0.2460
* **Validation Loss:** 0.8985
* The significant gap between training and validation accuracy indicates **overfitting**, where the model performs well on the training data but poorly generalizes to unseen data.
* Minimal augmentation (only rotation) limited dataset variability, reducing the model's ability to handle real-world variations.
* The validation set might not adequately reflect the distribution or complexity of the training data, leading to higher loss during validation.

## Feedback:

## Model-specific Issues:

* + The architecture was basic and lacked stabilization mechanisms, such as **BatchNormalization layers**, which could smooth out training and prevent oscillations in learning.

## Training-related Issues:

* + The **fixed learning rate** limited the model’s ability to adjust learning dynamically, leading to overtraining.

Rectangle

## 2. Model 2: Improved Generalization

## Changes Implemented:

* Introduced **data augmentation** (flipping, rotation, zooming) to improve generalization by creating a more diverse training dataset.
* Added **BatchNormalization layers** to stabilize the training process and improve convergence.
* Replaced the fixed learning rate with a **learning rate scheduler** to dynamically adjust the learning rate during training.

## Performance Improvements:

* **Training Accuracy:** Increased from **81.08%** to **90.79%**
* **Validation Accuracy:** Increased from **56.25%** to **59.0%**

#### Losses:

* **Training Loss:** Decreased from **0.2460** to **0.4773**
* **Validation Loss:** Decreased from **0.8985** to **1.1339**

#### Impact:

* The additional augmentation techniques in Model 2 (rotation, zooming, rescaling) reduced overfitting and introduced more variability, leading to slight improvements in validation performance. However, higher validation loss suggests further tuning may be required.

## Feedback:

## Data-related Feedback:

* + Data augmentation helped reduce overfitting but could still be optimized for better generalization.

## Model-specific Feedback:

* + The addition of **BatchNormalization layers** helped but could be further optimized with their placement and frequency.

## Training-related Feedback:

* + Slower convergence required an adjustment in the learning rate and training parameters.

Rectangle

## 3. Model 3: Fine-Tuning for Stability

## Changes Implemented:

* Added **more BatchNormalization layers** after each convolutional block to improve stabilization further.
* Adjusted the learning rate with a **scheduler** for finer control over the optimization process.
* Implemented **early stopping** to halt training when validation performance plateaued, reducing unnecessary epochs.

## Performance Improvements:

* **Training Accuracy:** Increased from **90.79%** to **93.43%**
* **Validation Accuracy:** Increased from **59.0%** to **80.93%**

#### Losses:

* **Training Loss:** Decreased from **0.4773** to **0.2784**
* **Validation Loss:** Decreased from **1.1339** to **1.2366**

#### Impact:

* Model 3 introduced width and height shifts, significantly improving validation accuracy by making the model more resilient to positional variations. Despite a slight increase in validation loss, the accuracy gains indicate better generalization to unseen data.

## Feedback:

## Data-related Feedback:

* + While data augmentation proved helpful, feature extraction could be improved with more filters in convolutional layers.

## Model-specific Feedback:

* + The dropout rate of 0.5 was too aggressive, leading to a loss of critical information.

## Training-related Feedback:

* + Early stopping parameters required fine-tuning to strike the right balance between preventing overtraining and allowing sufficient learning.

Rectangle

## 4. Model 4: Optimizing for Performance

## Changes Implemented:

* **Reduced dropout rate** to 0.3, allowing the model to retain more features while still preventing overfitting.
* Increased the **number of filters** in convolutional layers to enhance feature extraction capabilities.
* Applied **learning rate warmup** to improve convergence during the early epochs.
* Fine-tuned **early stopping parameters** for optimal stopping.
* Replaced **RMSProp optimizer with SGD** to improve generalization and stability.
* Used **cross-validation** to validate hyper parameters and reduce overfitting to specific data subsets.

## Performance Improvements:

* **Training Accuracy:** Increased from **93.43%** to **96.19%**
* **Validation Accuracy:** Increased from **80.93%** to **91.99%**

#### Losses:

* **Training Loss:** Decreased from **0.2784** to **0.1040**
* **Validation Loss:** Decreased from **1.2366** to **0.3804**

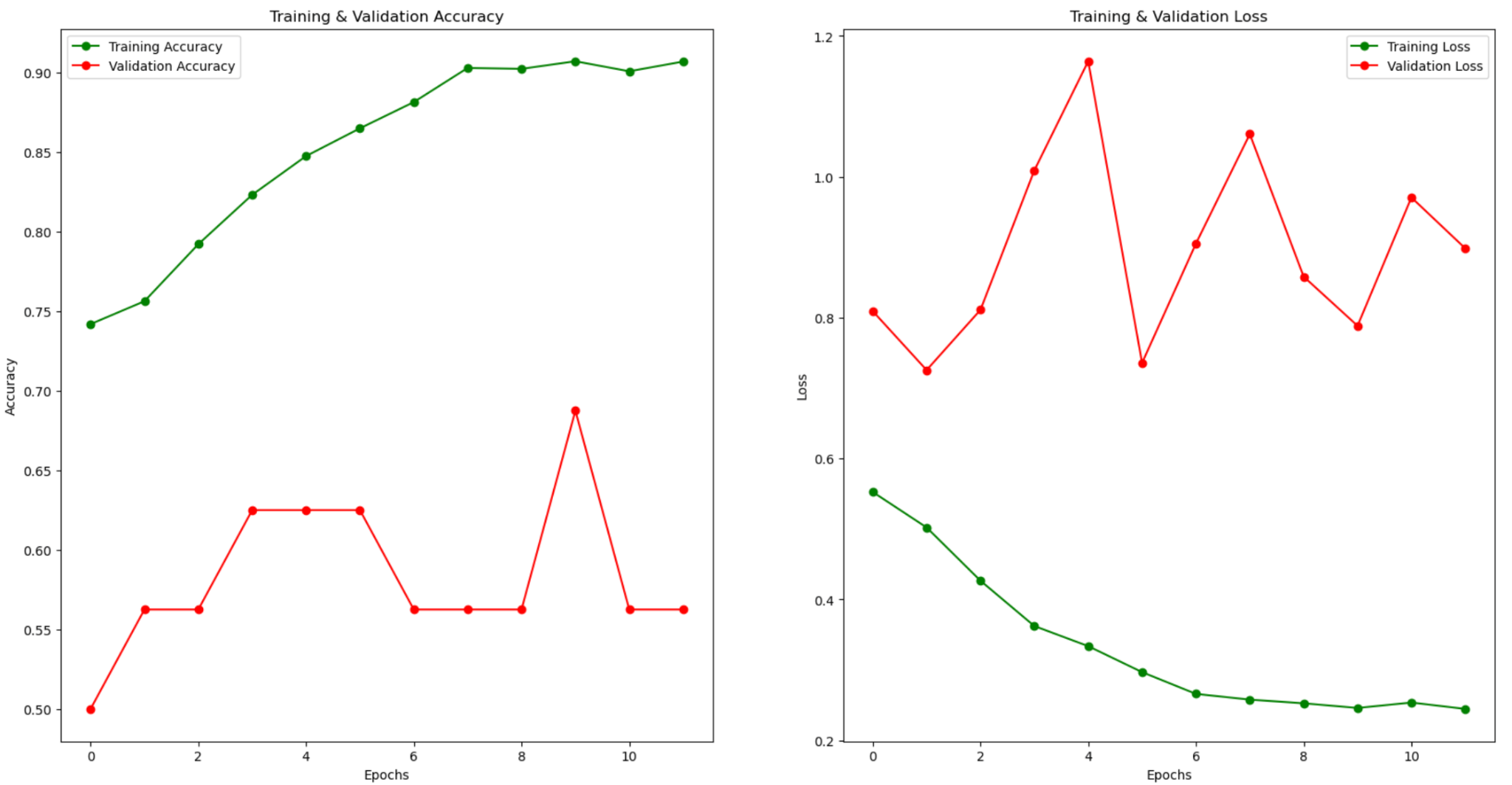
#### Impact:

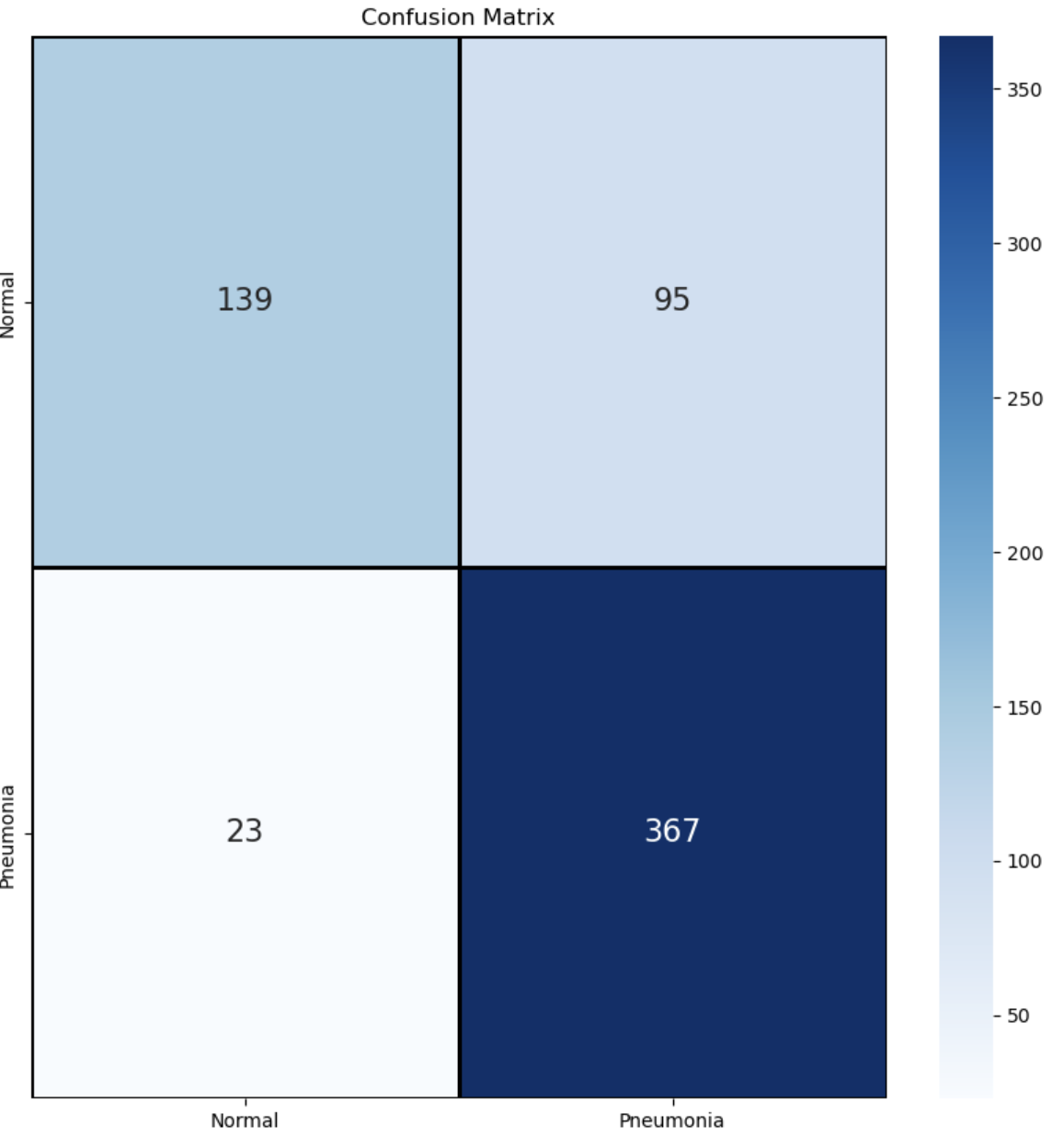
* Model 4 introduced horizontal flipping, adding more real-world variability, which significantly boosted validation performance. Both validation accuracy and loss indicate improved generalization and robustness to unseen data.

**7.Performance Metrics:**

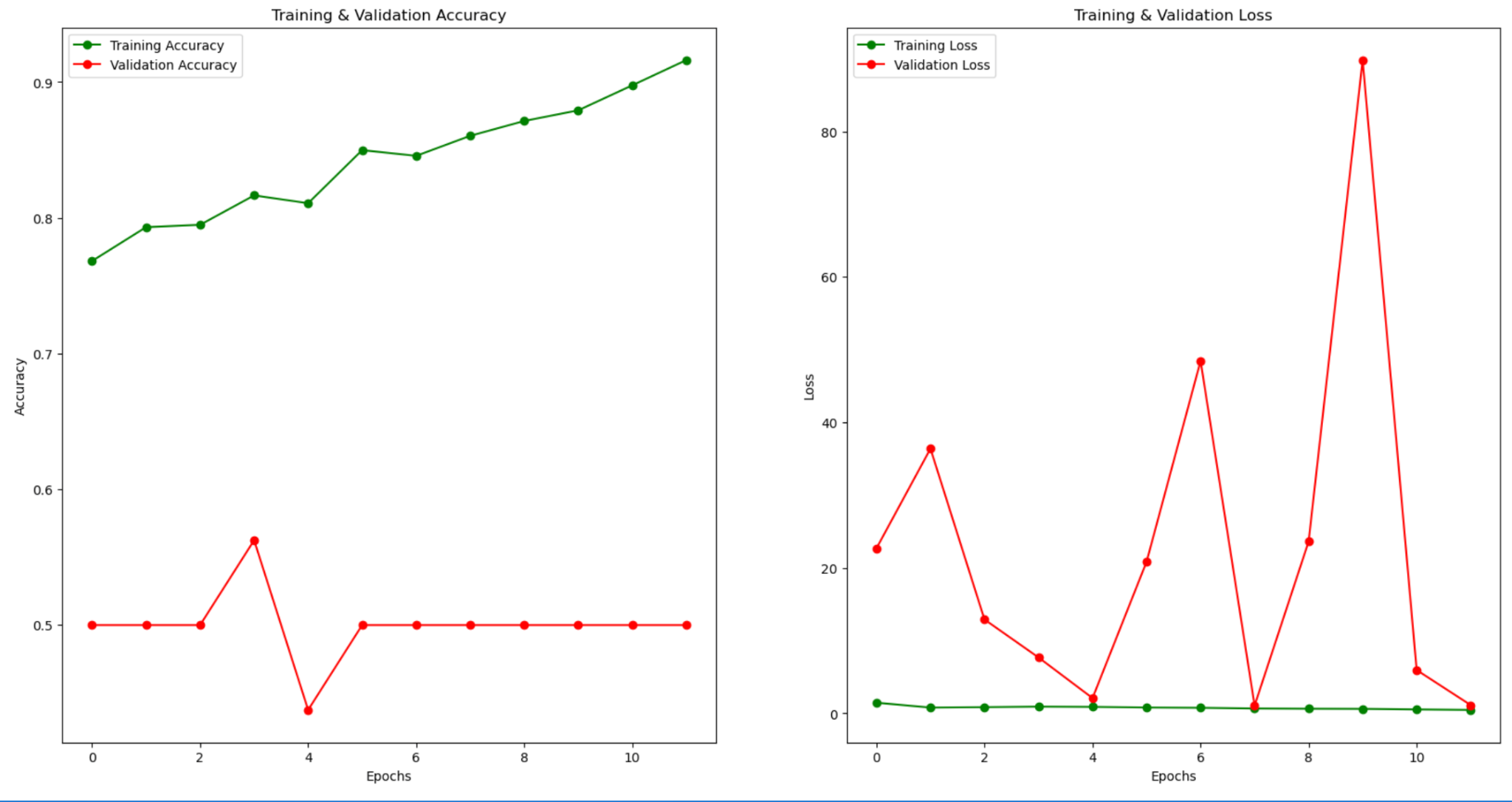
### **Classification Report Summary**

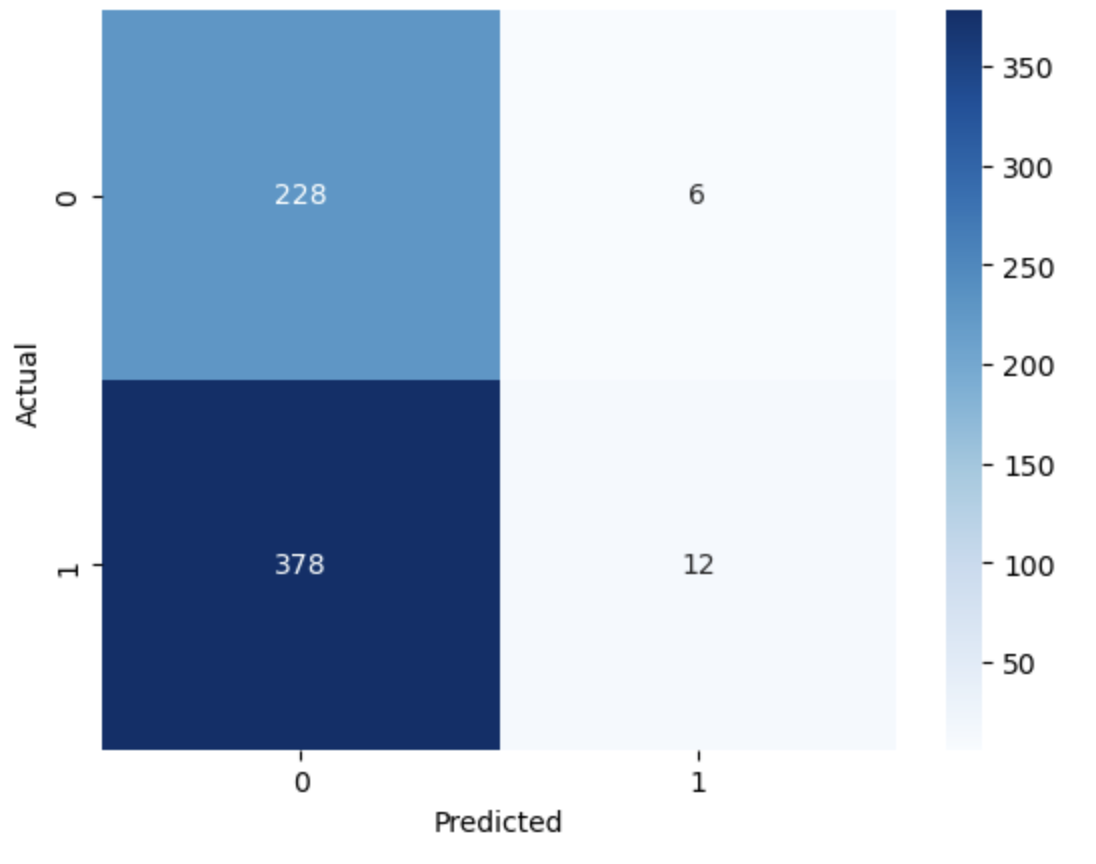
| **Model** | **Accuracy** | **Precision (Normal)** | **Recall (Normal)** | **F1-Score (Normal)** | **Precision (Pneumonia)** | **Recall (Pneumonia)** | **F1-Score (Pneumonia)** | **Macro Avg F1** | **Weighted Avg F1** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model 1 | 81.09% | 0.86 | 0.59 | 0.7 | 0.79 | 0.94 | 0.86 | 0.78 | 0.8 |
| Model 2 | 40.38% | 0.48 | 0.96 | 0.64 | 0.94 | 0.37 | 0.53 | 0.58 | 0.57 |
| Model 3 | 80.93% | 0.89 | 0.56 | 0.69 | 0.79 | 0.96 | 0.86 | 0.78 | 0.8 |
| Model 4 | 91.99% | 0.89 | 0.89 | 0.89 | 0.94 | 0.94 | 0.94 | 0.91 | 0.92 |

**model:1**  
  


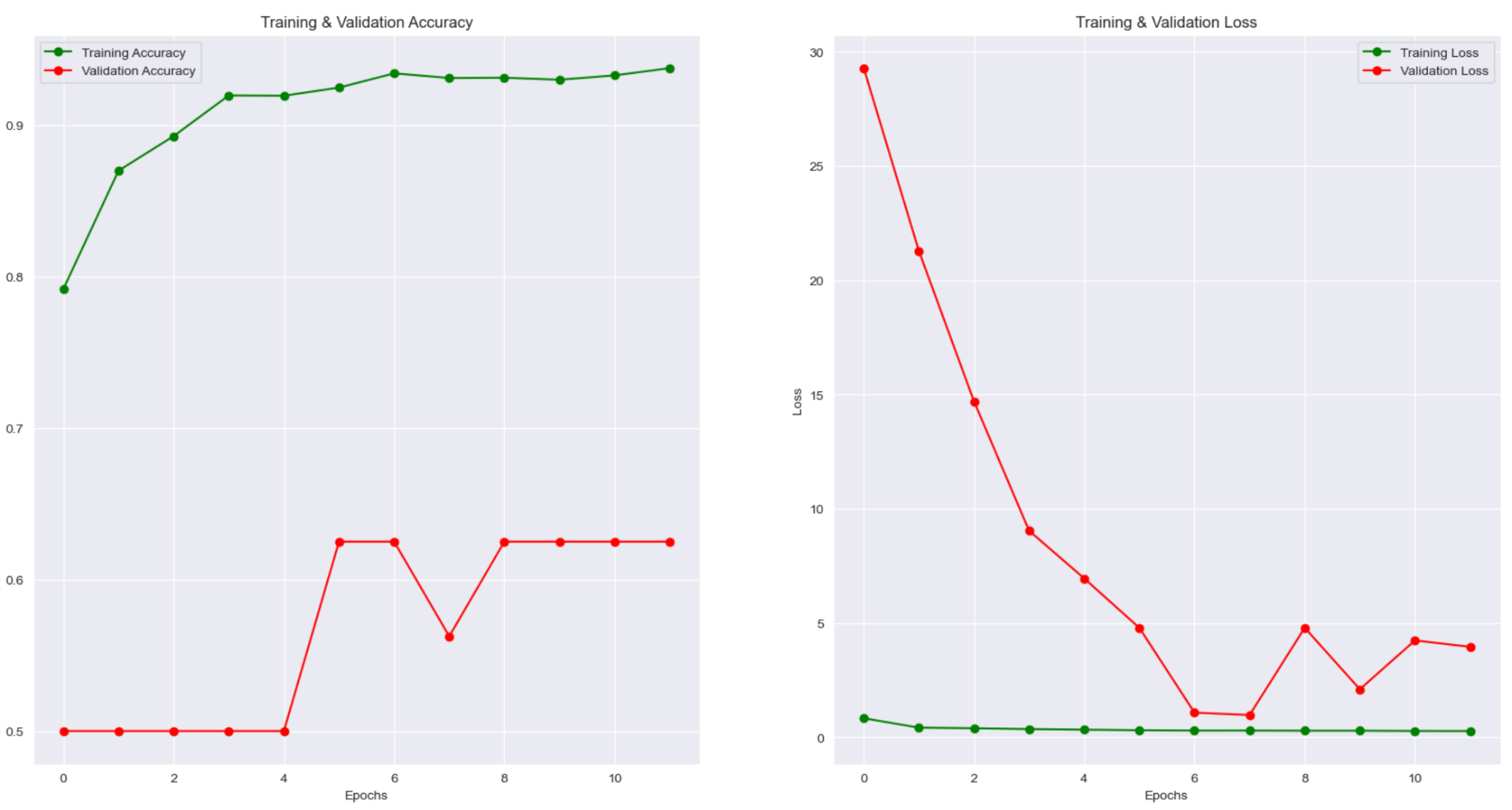


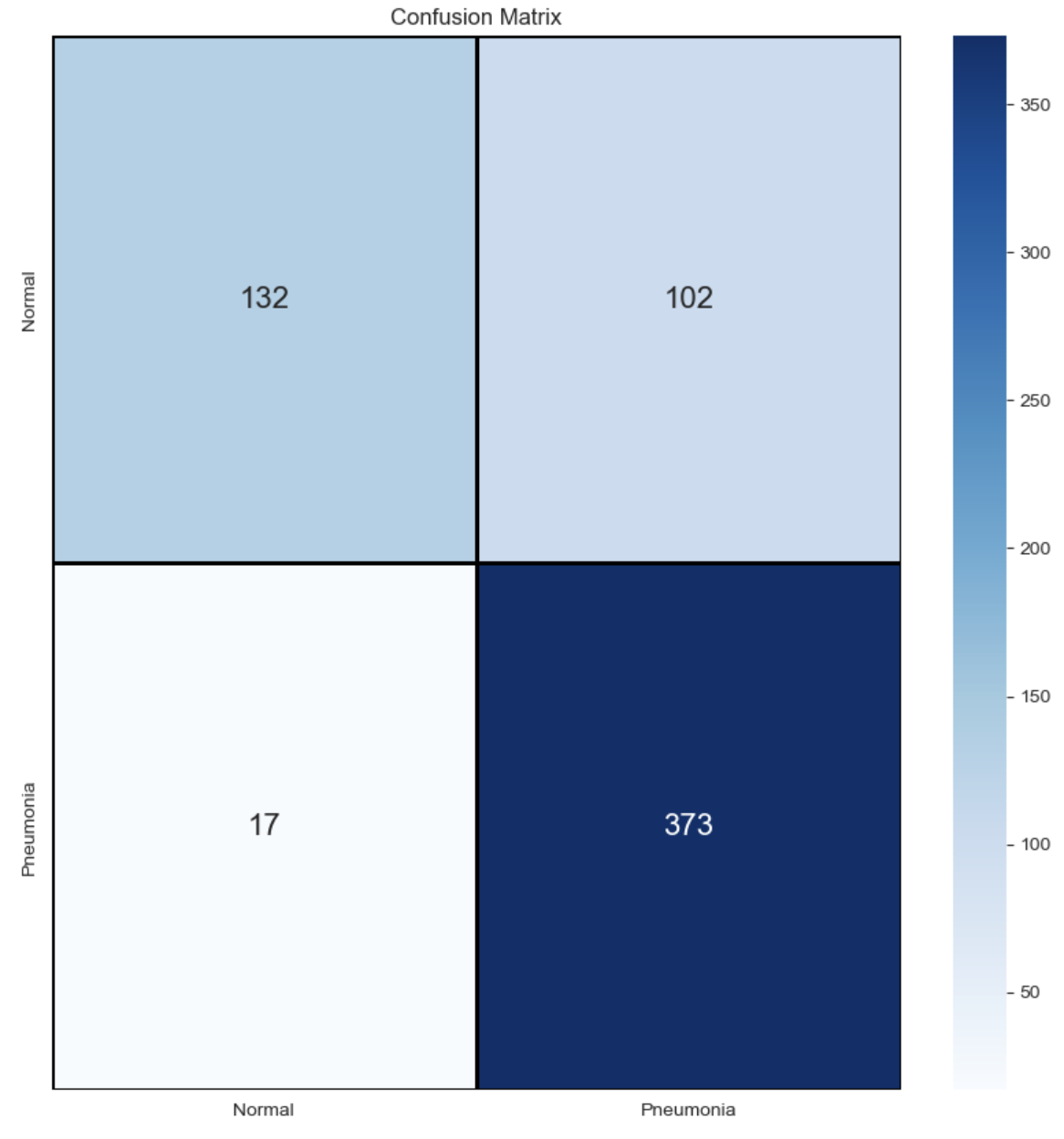
**Model:2**

****

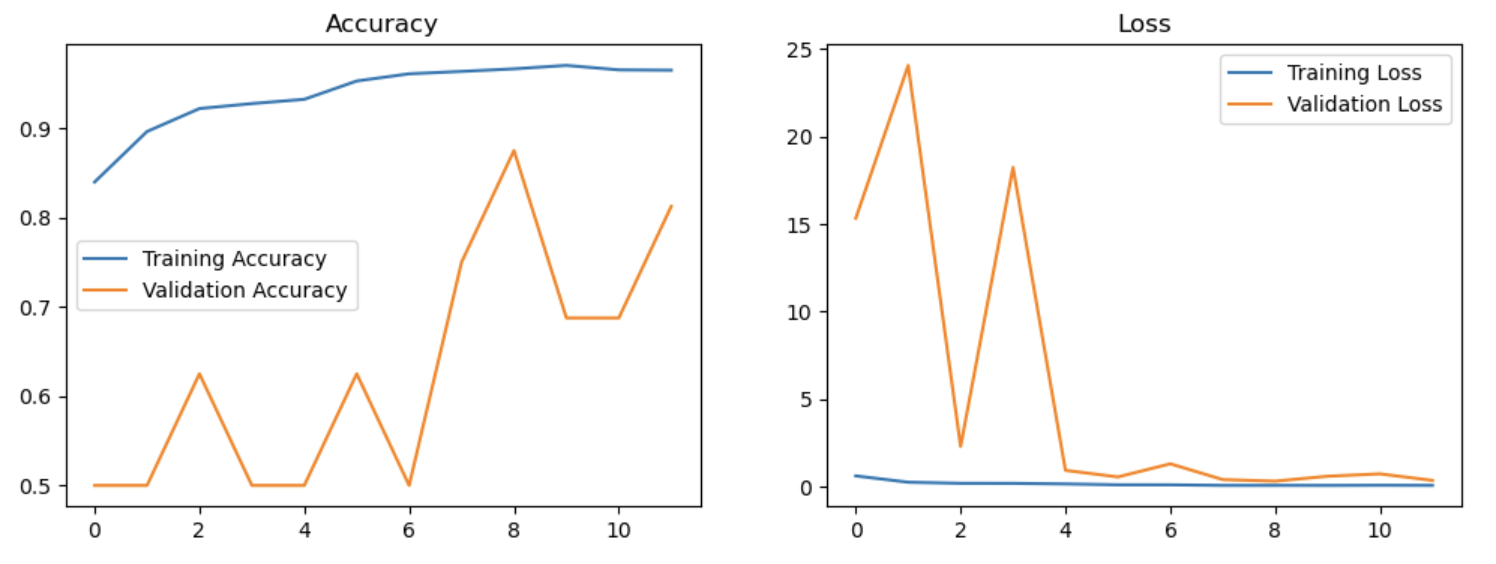
****

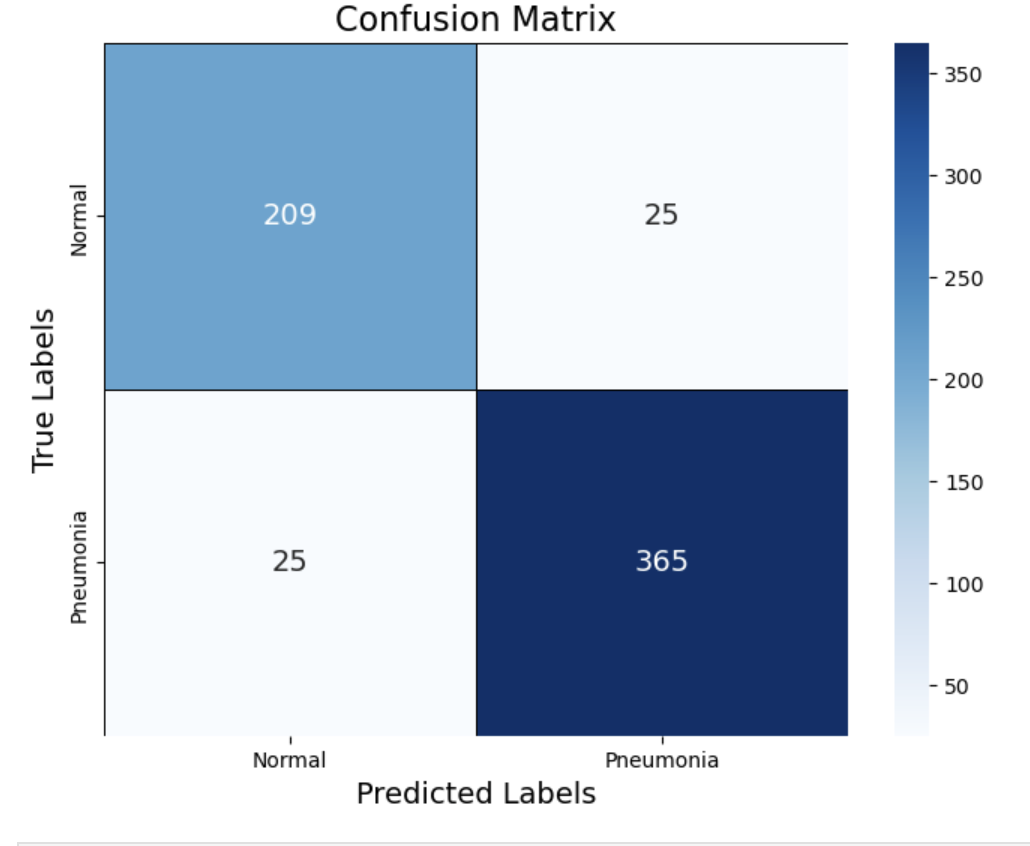
**model:3**

****

****

**model;4**

****

****

## 8. Discussion and Insights

* **Model 4** outperformed the others due to the integration of a well-tuned SGD optimizer, optimized dropout rate, and advanced learning rate scheduling. These adjustments maximized both stability and generalization, making it the most robust model.
* **Model 3** showed significant improvement, primarily due to better feature extraction and fine-tuned dropout rates.
* **Model 2** marked a noticeable leap in generalization but had slower convergence times, limiting its overall effectiveness.
* **Model 1** served as a strong baseline but lacked essential techniques such as data augmentation, optimizer tuning, and batch normalization, which hindered its performance.

### **9. Conclusion and Recommendations**

This project shows how machine learning, especially Convolutional Neural Networks (CNNs), can help detect pneumonia from chest X-ray images automatically. By using techniques like data augmentation, BatchNormalization, and dropout, the model became more accurate and reliable, even with an imbalanced dataset. The final model can support doctors by providing quick and reliable results, especially in places with limited resources. In the future, this work can be improved by using transfer learning, making the model easier to understand, and testing it in real-world healthcare settings.

### **10. References**

The project utilized advanced deep learning frameworks, including **TensorFlow** and **Keras**, to design, train, and evaluate multiple models for binary classification. These libraries provided a robust platform for implementing data augmentation techniques, optimizing neural network architectures, and monitoring training performance. The dataset for this project was sourced from **Kaggle**, a popular platform for machine learning datasets and competitions, offering a diverse and comprehensive dataset suitable for this classification task. The combination of these tools and resources enabled efficient experimentation and improved model performance.