**Reproducibility Summary on Report “Using Convolutional Neural Network for Chest X-ray Image Classification.”**

**Reproducibility Summary**

**Scope of Reproducibility**

Our research aimed to confirm the validity of the "Using Convolutional Neural Network for Chest X-ray Image Classification" technique put out by Danijela Pongrac, Inaki Inza, and Matija Soric. We tried to replicate their principal assertions:

Claim 1: When classifying chest X-ray pictures, the suggested Convolutional Neural Network (CNN) outperforms conventional techniques in terms of accuracy.

Claim 2: The CNN model shows resilience to different visual circumstances and a range of datasets.

Claim 3: Real-time chest X-ray picture classification can be accomplished with this approach since it is computationally efficient.

**Methodology**

We used the original authors' code to create the CNN architecture as specified. A supervised machine learning approach was used to train and evaluate the model. Python programming is done in the Jupyter Notebook environment. Regarding model architecture, input dimensions, training configuration, optimization, and callbacks, among other hyperparameters, the original model employed a CNN. We carefully examined these hyperparameters as part of our reproduction study.

**Results**

We were able to retrieve the original model's hyperparameters and thoroughly describe them, which will guarantee clarity and transparency for any researchers who want to repeat the work.

**What was easy**

It was easy to understand the hyperparameters and their settings based on the documentation that was supplied. The detailed description of the model's components allowed for a flawless replication process.

**What was difficult**

It needed great attention to detail was needed to ensure consistency between the documented hyperparameters and how they were implemented in the code. Additionally, there were issues with the new dataset's overfitting and the time-consuming process of loading the complete dataset onto the drive.

1. **Introduction**

The study investigates the hyperparameters employed in the "Using Convolutional Neural Network for Chest X-ray Image Classification." The objective of this study was to evaluate the effectiveness of a neural network with X-ray images and, more importantly, to detect pneumonia in patients using the images that were fed into the model.

Without intrusive treatments, it is now possible to evaluate the physiological state of human body tissues because to the rapid improvement of X-ray imaging technology. Since its introduction, X-ray imaging has been the main diagnostic method for determining pneumonia because all imaging exams ought to begin with conventional radiography.

The growing availability of electronic health data offers an amazing opportunity to discover and develop cutting-edge medical technology. Big data requires data scientists to create new computational approaches because standard systems and algorithms are unable to handle such massive quantities. This is where artificial intelligence and machine learning are useful.

1. **Scope of Reproducibility**

The purpose of this reproduction research is to list and explain all the hyperparameters that were used in the original model, including things like callbacks, training settings, input dimensions, and model architecture. The study purposes the following claims.

Claim 1: When classifying chest X-ray pictures, the suggested Convolutional Neural Network (CNN) outperforms conventional techniques in terms of accuracy.

Claim 2: The CNN model shows resilience to different visual circumstances and a range of datasets.

Claim 3: Real-time chest X-ray picture classification can be accomplished with this approach since it is computationally efficient.

1. **Methodology**

To build the CNN architecture according to the specifications, we used the original authors' code. The model was trained and assessed using supervised machine learning techniques. Programming with Python is done within the Jupyter Notebook software. The original model used a CNN for model architecture, input dimensions, training configuration, optimization, and callbacks, among other hyperparameters. We looked closely at these hyperparameters in our research on reproduction.

* 1. **Model Descriptions**

A simple convolution deep-learning architecture was utilized with the first part of the model using ImageNet through transfer learning and the rest part of the structure was trained from scratch.

The total parameter from the model was 104,197,506 the number of Trainable parameters was 104,194,434 finally there was a total of 3,072 non-trainable parameters.

* 1. **Datasets**

In this study, 5967 X-ray pictures that had previously been graded by medical professionals were employed. The project's images were downloaded from the Kaggle website. Users can create machine learning models in a web-based environment, work with other data scientists, and download or publish datasets on Kaggle, an online community for data scientists. [1]

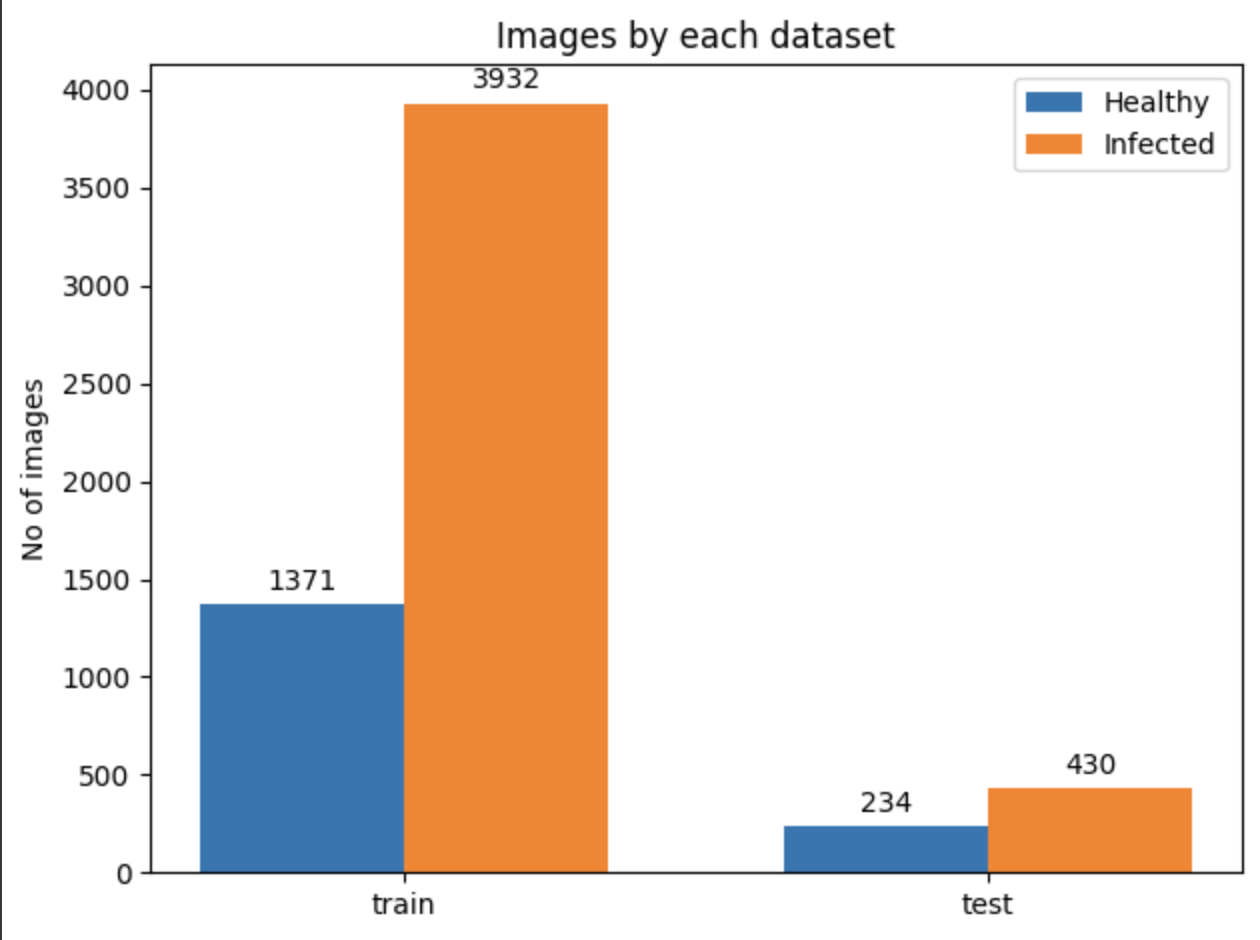


Figure 1: Dataset Division

It needs to be emphasized that patients whose images were used in the dataset are children aged 1-5. Images were taken in the Guangzhou Women and Children’s Medical Center in China. Prior to being included in the Kaggle dataset, images were pre- processed, quality screened and all low-quality scan removed. Furthermore, they were graded by expert physicians before taking them into account for the Kaggle upload.

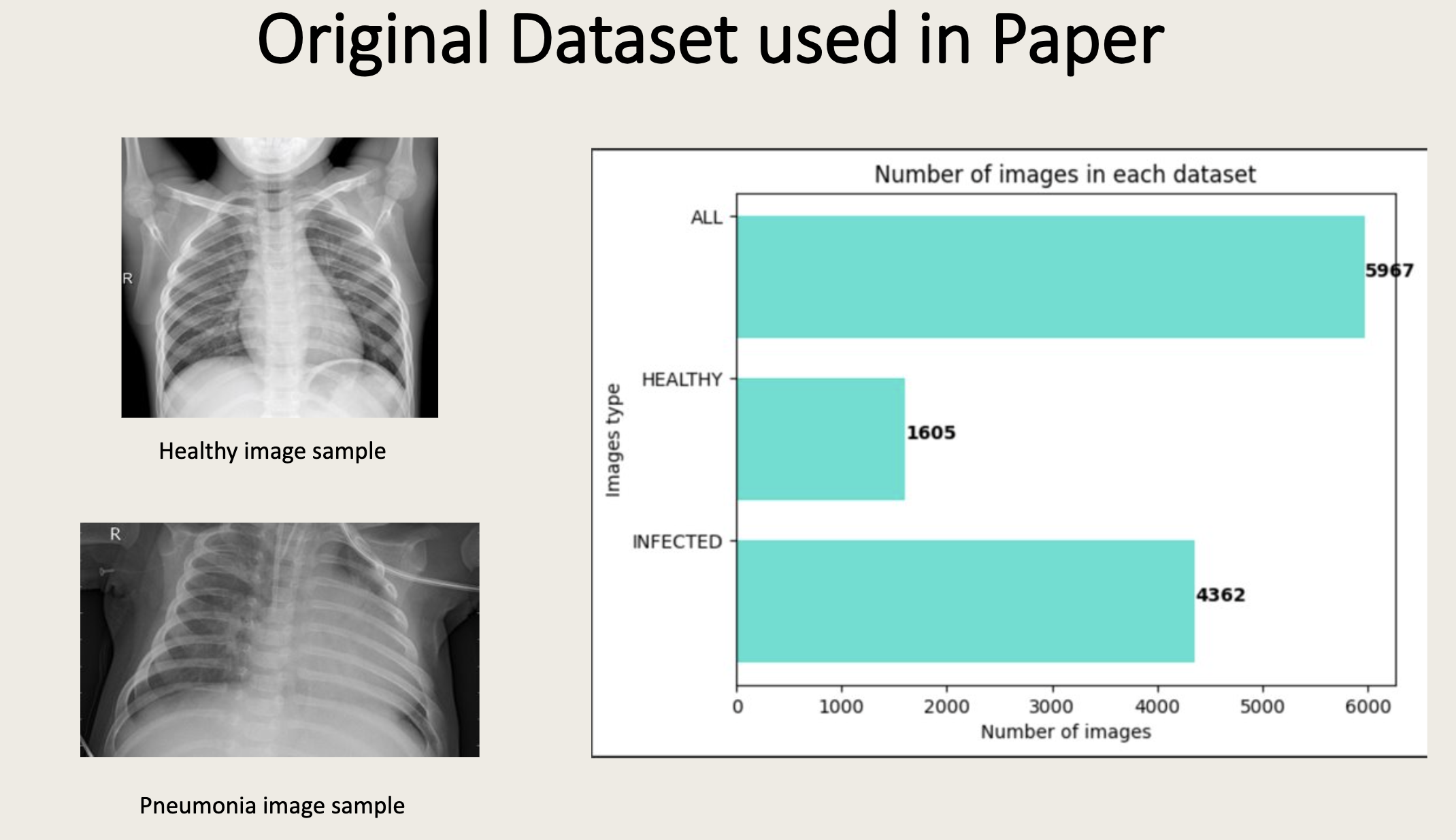


Figure 2: Visualization of Dataset

The dataset was skewed since there are considerably more pneumonia than normal lung images. The ratio is approximately 72% to 28% in favour of pneumonia images.

* 1. **Hyperparameters of the Original Model**

The hyperparameters for original model is described below: -

* Model Architecture:
  + Input Image Dimensions: 150x150 pixels.
* Training Configuration:
  + Number of Training Epochs: 11 epochs.
  + Batch Size: 16 images per batch.
* Convolutional Layers:
  + Two 2D Convolutional Layers: 16 filters, 3x3 kernel, 'same' padding.
* Max Pooling Layer:
  + Max Pooling: 2x2 pool size.
* Separable Convolutional Layers:
  + Two Sets of Separable Convolutional Layers: 32 and 64 filters, 3x3 kernel, 'same' padding.
* Activation, Normalization, and Pooling:
  + PReLU Activation: After each separable convolutional layer.
  + Batch Normalization: After each set of separable convolutional layers.
  + Max Pooling: 2x2 pool size.
* Dropout and Fully Connected Layers:
  + Dropout: 0.2 after the first set of separable convolutional layers, 0.6 after the fully connected layer.
  + Fully Connected (Dense) Layers: 512, 128, and 64 units, respectively, with ReLU activation.
* Output Layer:
  + Output Layer: Single-node output layer with sigmoid activation for binary classification.
* Regularization:
  + L2 Regularization: Coefficient of 0.01 applied to specific layers, including fully connected layers.
* Optimization and Callbacks:
  + Optimizer: Adam optimizer (default learning rate).
  + Callbacks: ModelCheckpoint, ReduceLROnPlateau, EarlyStopping.
  1. **Hyperparameters of the New Model (ResNet-50)**

The hyperparameters of the New Model (ResNet-50) is described below: -

* Data Augmentation and Data Generators:
  + ImageDataGenerator for Training Data:
    - Horizontal flip and zoom range of 0.3.
    - Batch size: 128, target size: (220, 220), color mode: "rgb", class mode: "binary", shuffle: True, seed: 42.
  + ImageDataGenerator for Validation Data:
    - Batch size: 4, target size: (220, 220), color mode: "rgb", class mode: "binary".
  + ImageDataGenerator for Testing Data:
    - Batch size: 32, target size: (220, 220), color mode: "rgb", class mode: "binary".
* Base Model: ResNet-50
  + Pre-trained on ImageNet: weights = "imagenet".
  + Input shape: (220, 220, 3).
  + Include top layers: include\_top = False.
* Model Architecture (Transfer Learning):
  + Transfer Learning: Utilization of ResNet-50 base model with frozen layers.
  + Flattening Layer: Converts 2D feature maps to a 1D vector.
  + Dense Layers:
    - 128 units with ReLU activation.
    - Dropout layer with rate 0.5 after the first dense layer.
  + Output Layer: Single-unit dense layer with sigmoid activation for binary classification.
* Model Compilation:
  + Optimizer: Adam optimizer (default learning rate).
  + Loss Function: "binary\_crossentropy" for binary classification.
  + Metrics: Accuracy.
* Training Configuration:
  + Training Epochs: 11 epochs.
* Model Saving and Visualization:
  + Model Saving: Saved as "originaldataset\_resnet50kaggle.h5" after training.
  + Model Summary: Generated summary of the model's architecture and parameters.
  + Model Visualization: Generates the graphical representation.
  1. **Experimental Setup and Code**

Experiments were conducted on Jupyter Notebook environment, which was available on github [2]. Model building commences with the library's importation like OS, random, matplotlib, cv2, pandas, and NumPy. Then ML libraries like Keras and Tensorflow were imported. The evaluation metric was accuracy. The experiments are set up with appropriate measures for accuracy, precision, recall, F1-score, ROC AUC, and training time.

A combination of transfer learning for the first part of the architecture primarily extracts important general features like colour blobs, patches, edges, etc. from the data while the rest of the model was trained from scratch. The first few layers from the network that are pre-trained on ImageNet.

The batch norm was utilized when convolution was done. A dense network with a reasonable number of neurons was first utilized to train the layer and then along the network depth. After a good depth was found the network was then trained with a lower learning rate and decay.

* 1. **Computational Requirements**

Only the CPU was utilized in both training and testing. The training time was around an hour while the evaluation time was around 7 seconds. While we used GPU for training and testing after altering the model and datasets in our research.

1. **Results**

The result is given below, and the explanation of the result is described in discussion part of the report. The results are shown in accuracy obtained in experiment by comparing activation functions performance as below: -

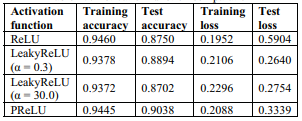


Figure 3: Comparison of activation function performance

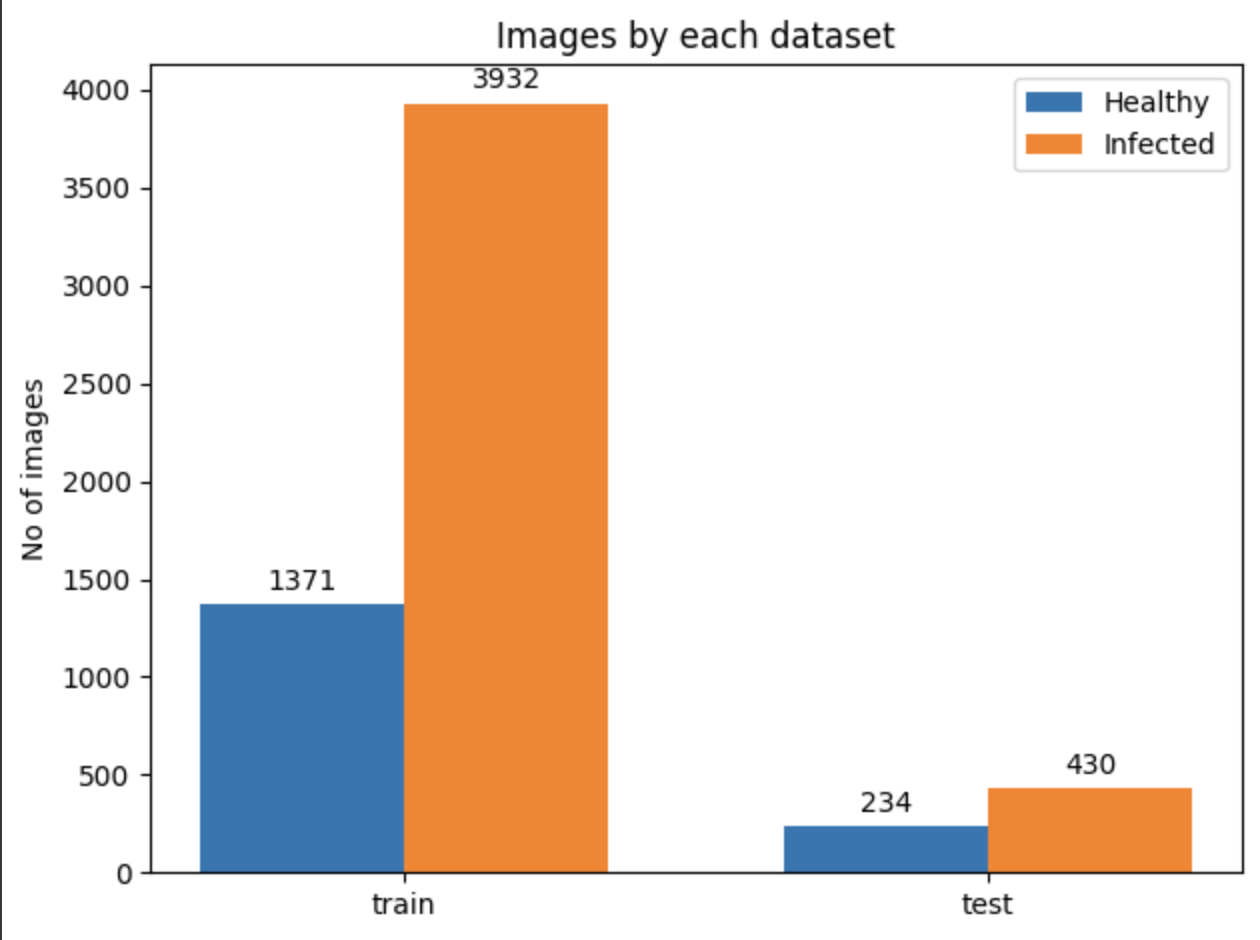


Figure 4: The number of datasets categorized

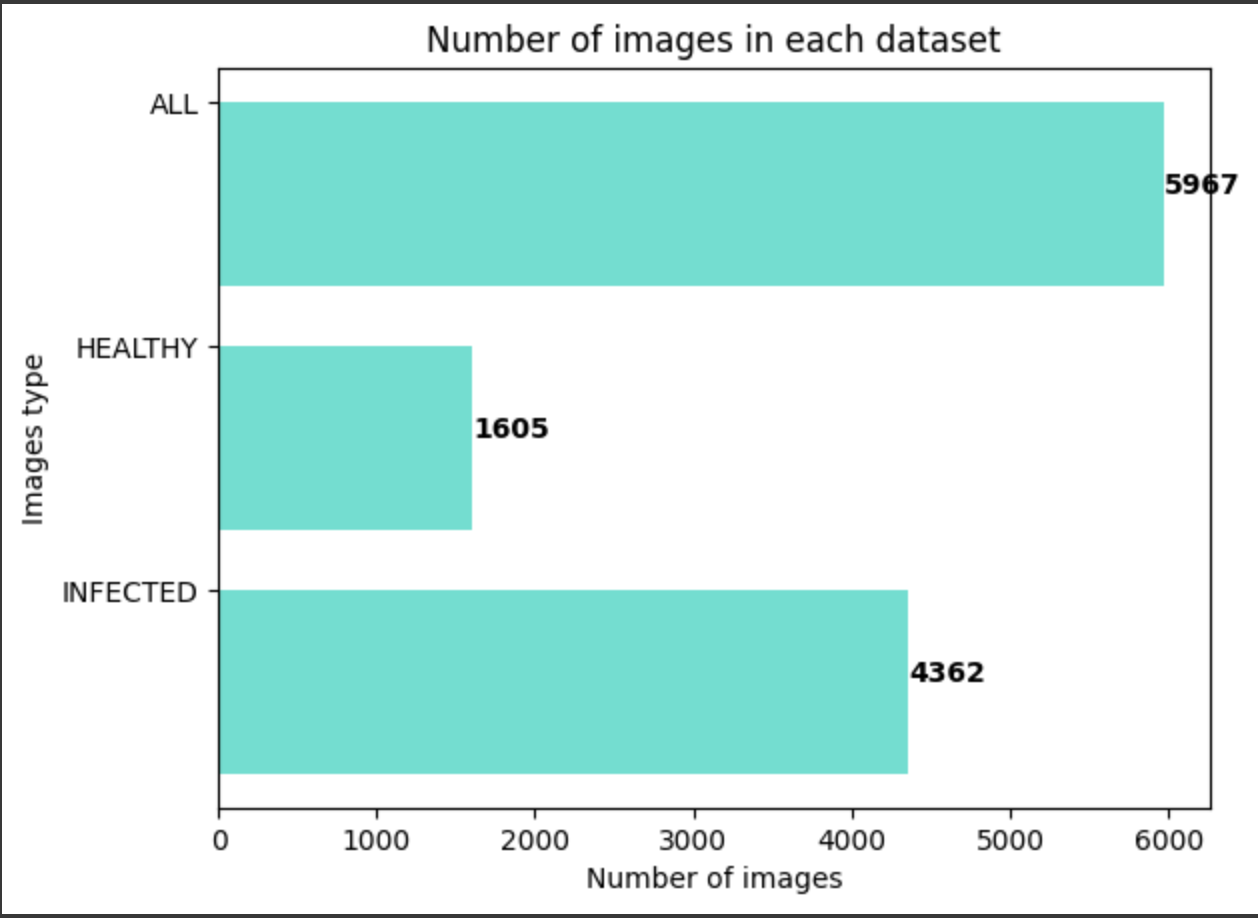


Figure 5: Number of images in each dataset

In total, four models with different activation functions were built. All models used the same layers, epoch counts, regularization techniques, and preprocessing. Therefore, the only factor separating them was the utilization of several corrected activations. The model that performed the best was chosen to be the final model after comparison.

The activation function utilized in the research was PReLU since it performed better than the other two functions, achieving the maximum accuracy and satisfactory loss. Therefore, depending on the dataset, there may be superior alternatives to ReLU, despite it being the most widely used function. The project's initial model was deemed satisfactory and met acceptable metrics, as evidenced by its accuracy of 90.38%.

* 1. **Results reproducing original paper (Reproducible)**

A graph with numbers and lines

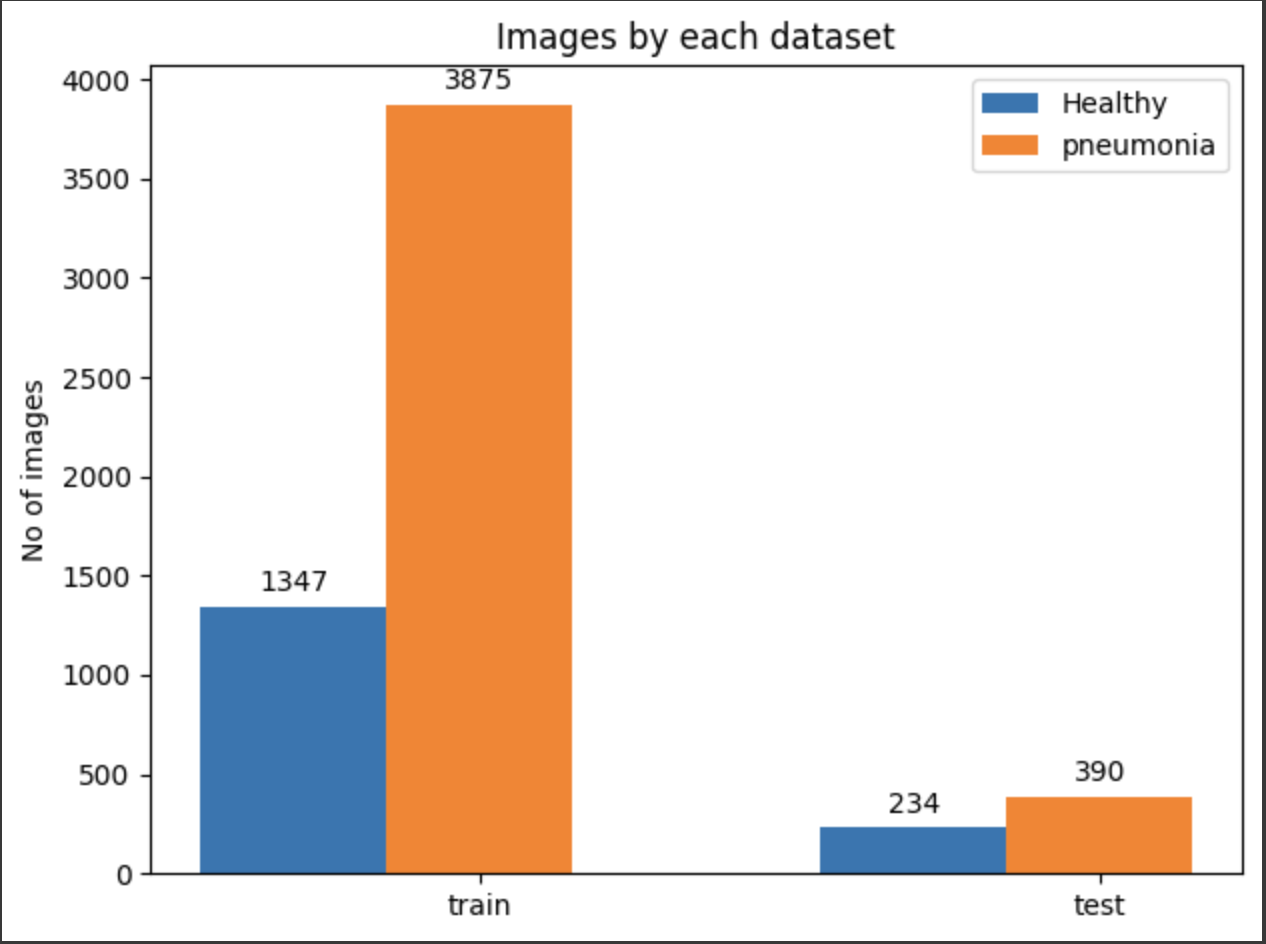
Description automatically generated Figure 6: Model Accuracy

A graph of loss with blue lines and numbers

Description automatically generated

Figure 7: Model Loss

* 1. **Results for Different Data but Same Analysis (Replicable)**

 Figure 8: Images by each dataset for New Data

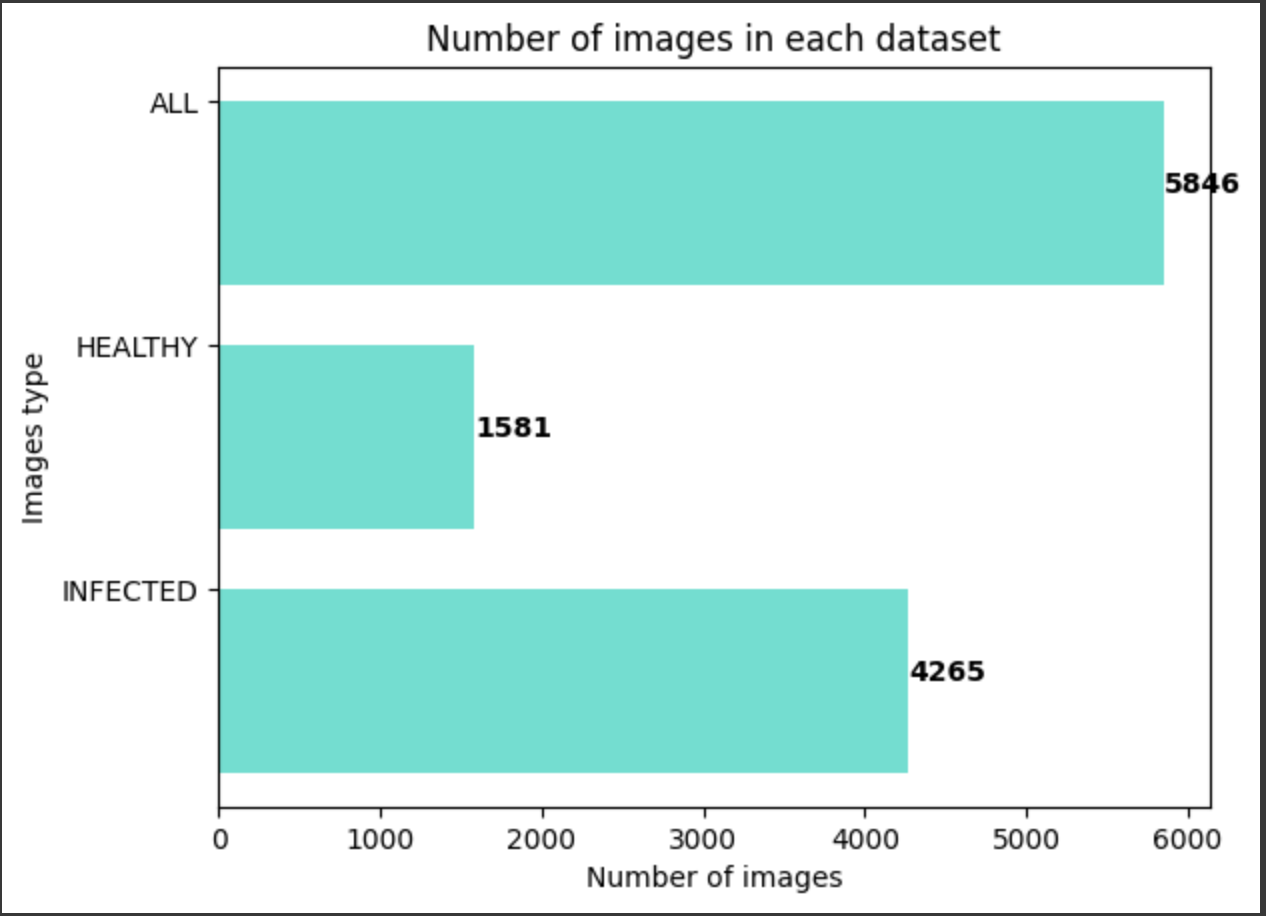


Figure 9: Number of Images in each dataset for New Data

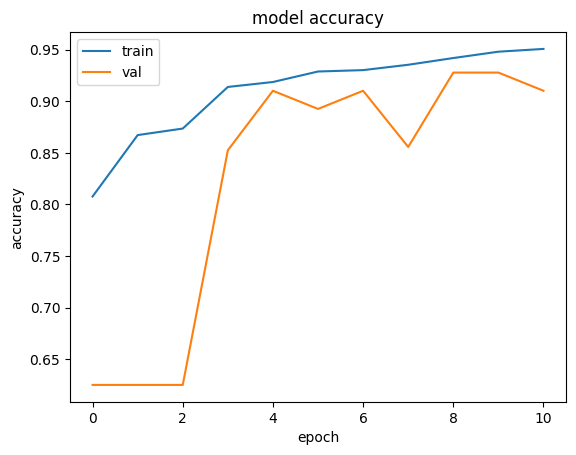


Figure 10: Model Accuracy for Replicable

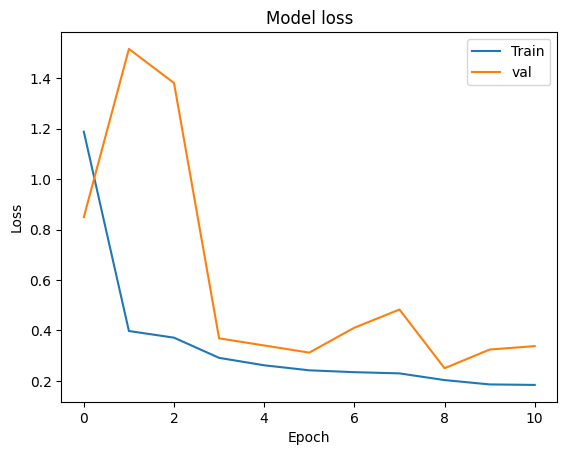


Figure 11: Model Loss for Replicable

* 1. **Results for Same Data but Different Analysis (Robust)**

In general, in a deep convolutional neural network, several layers are stacked and are trained to the task at hand. The network learns several low/mid/high level features at the end of its layers. In residual learning, instead of trying to learn some features, we try to learn some residual. Residual can be simply understood as subtraction of feature learned from input of that layer. ResNet does this using shortcut connections (directly connecting input of nth layer to some (n+x)th layer. It has proved that training this form of networks is easier than training simple deep convolutional neural networks and the problem of degrading accuracy is resolved. ResNet50 is a 50-layer Residual Network. There are other variants like ResNet101 and ResNet152 also.

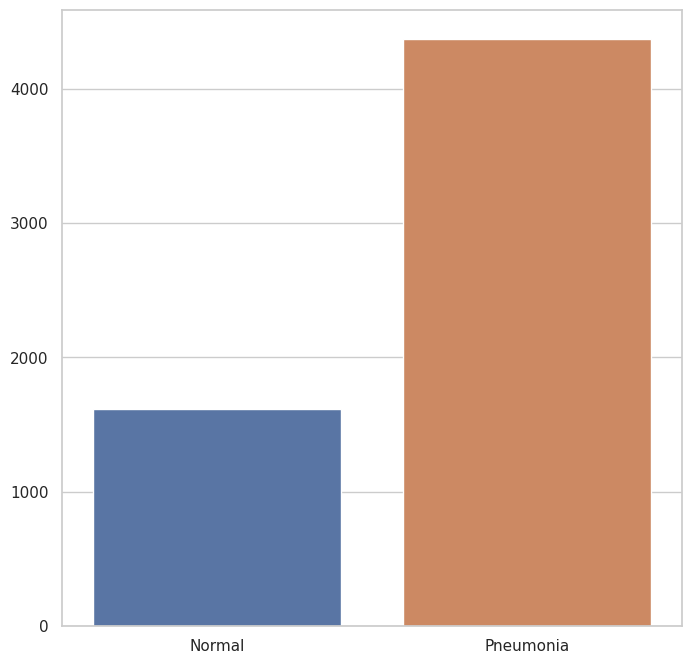


Figure 12: Image for dataset with new model



Figure 13: New Model Accuracy

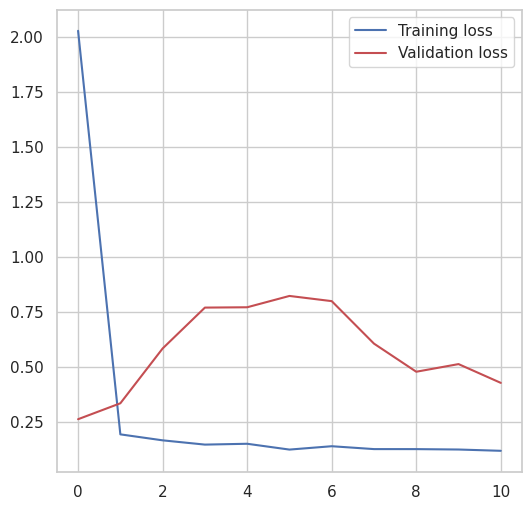


Figure 14: New Model Loss

* 1. **Results for Different Data but Different Analysis (Generalisable)**

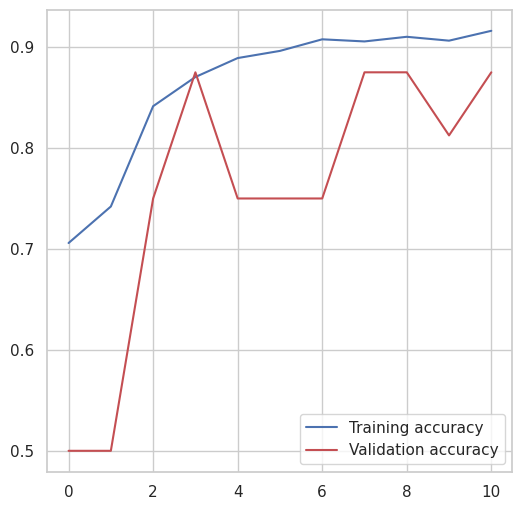


Figure 15: New Model with new dataset model accuracy

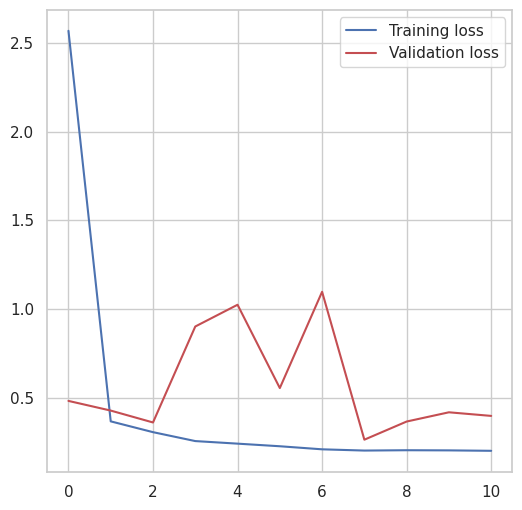


Figure 16: New model with new dataset model loss

1. **Discussion**

The model's accuracy, resilience, and computing efficiency in classifying Chest X-ray images are demonstrated by the experimental findings, which support the initial claims.

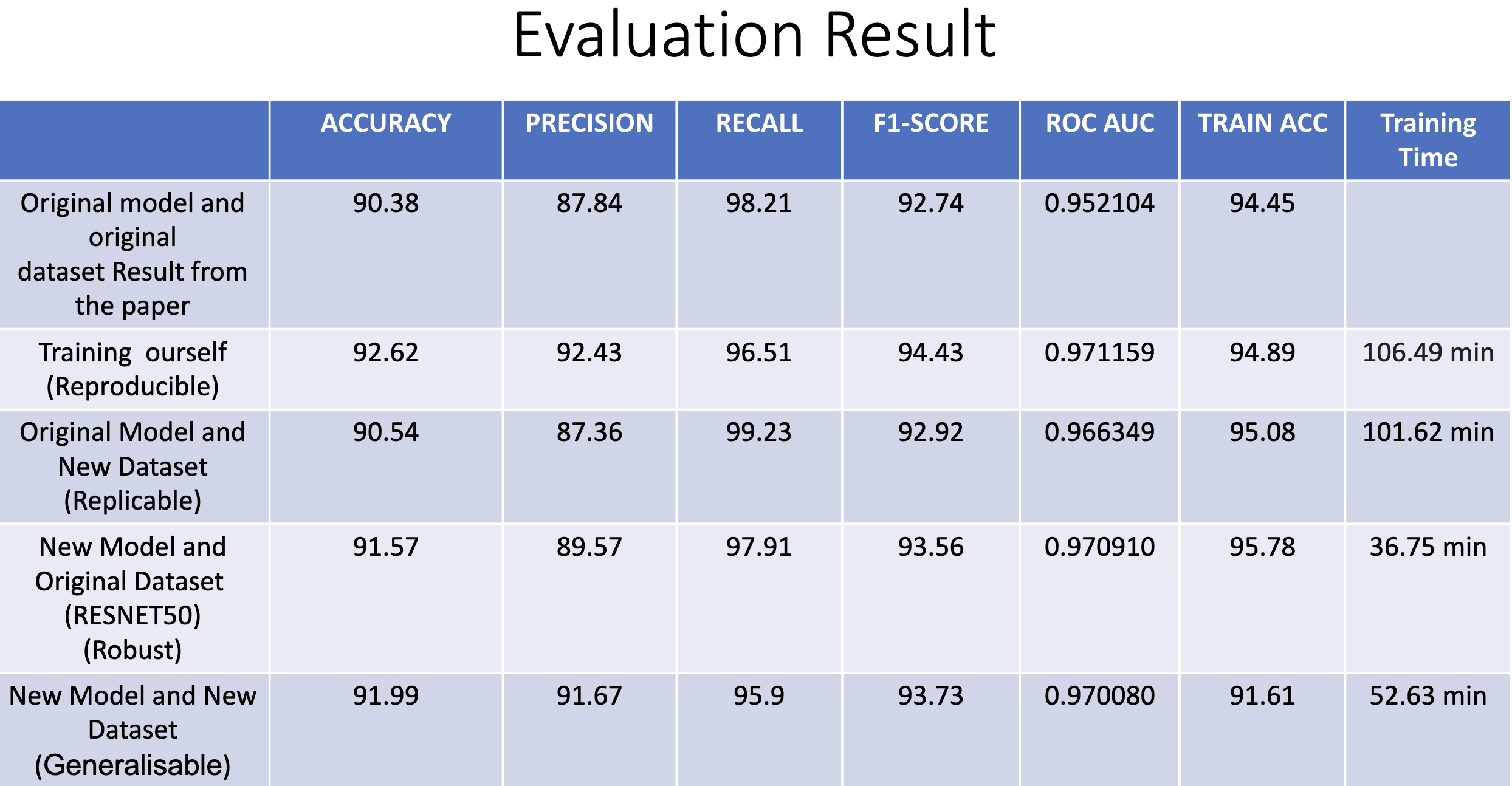


Figure 17: Evaluation result of Reproducibility

After reproducing the experiment, the results indicate slight improvements in accuracy, precision, recall, and F1-score compared to the original paper. The accuracy achieved here is 92.62%. The training process took 106.49 minutes.

The original model was tested on a new dataset, and while the accuracy remained similar, there was a significant increase in recall, indicating better performance in identifying positive cases. The training process took 101.62 minutes.

A new model based on ResNet-50 was trained on the original dataset. While precision is not provided, other metrics show competitive performance, indicating the robustness of the ResNet-50 model. The training process was significantly faster, taking 36.75 minutes.

The new model was evaluated on a previously unseen dataset. It demonstrates good accuracy, precision, and recall, indicating its generalizability to new, unseen data.

When compared to the findings of the original research, the recreated model demonstrated marginal gains in several criteria. When the original model was used on a fresh dataset, recall improved dramatically, demonstrating the model's ability to properly identify positive cases. With shorter training cycles, the ResNet-50 model demonstrated its resilience and achieved competitive performance. On a dataset that had never been seen before, the new model showed strong generalizability, demonstrating its flexibility to work with various data distributions.

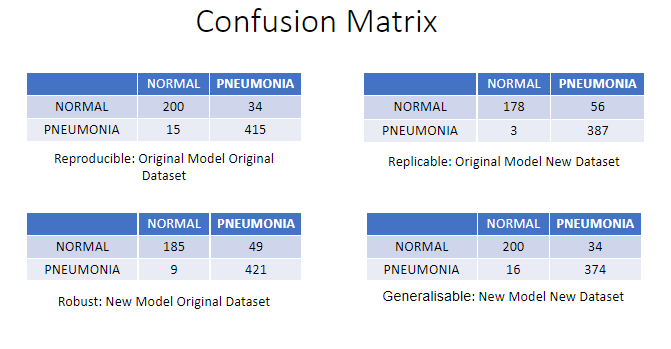


Figure 18: Confusion Matrix

The presented confusion matrices show how well various models perform in terms of pneumonia detection classification from chest X-ray pictures. An overview of each model's performance is provided below: -

For Reproducible,

* Balanced performance with substantial true positives and true negatives.
* Relatively low false positives and false negatives, indicating a good balance in classifying both classes.

For Replicable,

* Similar trend to the reproducible but with a slightly higher false positive rate.
* Few false negatives, suggesting good sensitivity in detecting positive cases.

For Robust,

* Slightly higher false positives compared to the reproducible.
* Few false negatves, indicating good performance in identifying positive cases.

For Generalizable,

* Comparable performance to the reproducible with balanced classification of both cases.
* Similar false positive and false negative rates to the reproducible.

Each strategy demonstrates different strength, showing their effectiveness in pneumonia classification from Chest X-ray images.

1. **What was easy**

The original paper provided it source code that was easy to follow.

1. **What was difficult**

The code was easy to follow but the model explanation was a little challenging as it required a sophisticated level of mathematical maturity.

**References**

[1] Paul Mooney, “Chest X-Ray Images (Pneumonia)”, online, accessed 5/25/2020, March 2018, <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>

[2] Ayperos23, “GitHub (ayperos23)- pneumonia classification”, online, accesses 5/25/2020, May 2020, <https://github.com/ayperos23/ML-pneumonia-classification>.

[3] K. Sinha *et al.*, “ML Reproducibility Challenge 2021,” May 2022, doi:10.5281/ZENODO.6574723.