Principles in Deep Learning: Project Report

Dr. Moshe Botman

Pneumonia Detection Using Chest x-rays.

Jonathan Cwengel, Tal Zigelnik

Deep Learning Internship 2024

Department of Computer Science

College of Management Academic Studies

1. Introduction

Chest X-Rays Image classification to detect pneumonia and identify anomalies indicative of other potential lung conditions (bacterial, viral). Utilizing a comprehensive dataset of chest X-ray images categorized into normal and pneumonia-affected, the project employs convolutional neural networks (CNNs) for classification tasks and methods used to preprocess images, generate feature embeddings, utilize a k-Nearest Neighbour classifier for predictions, and apply t-SNE for visualization of the high-dimensional data. In addition another goal was to implement an anomaly detection system for chest X-ray images and subsequently detect anomalies that deviate from this norm.

2. Problem Definition and Algorithm

2.1 Task Definition

The problem addressed in this project is the classification and anomaly detection in chest X-ray images. The primary objectives are two-fold:

2.1.1 **Classification**: To categorize X-ray images into two distinct classes – ‘Normal’, ‘Pneumonia’ via Binary classification and then between three distinct classes - 'Normal', 'Pneumonia Bacterial', and 'Pneumonia Viral' via Categorical classification. This classification helps in differentiating between healthy individuals and those affected by two types of pneumonia.

2.1.2 **Anomaly Detection**: To identify abnormal chest X-ray images that deviate from the 'Normal' class using an autoencoder. This detection is crucial for flagging potential cases of pneumonia or other lung conditions that are not typical of the healthy population.

2.2 Algorithm Definition

2.2.1 Classification Algorithm:

A convolutional neural network (CNN) is employed for image classification. The CNN model is designed with several convolutional layers with ReLU activations, followed by max pooling and dropout for regularization. Batch normalization is applied to accelerate training and stabilize the learning process. The final layer uses a softmax activation function to output a probability distribution over the three classes.

2.2.2 **Anomaly Detection Algorithm:**

An autoencoder neural network is used for anomaly detection. The autoencoder is structured to learn a compressed representation of 'Normal' X-ray images. It uses dense layers with a bottleneck architecture, where the middle layer (encoding layer) has fewer neurons than the input layer, forcing the autoencoder to learn a compact representation of the data. The autoencoder is trained to minimize the reconstruction error, which is measured as the mean squared error between the input and the reconstructed output. Anomalies are detected based on a threshold set on the reconstruction error, identifying images that the autoencoder reconstructs with significant error.

2.2.3 **t-SNE Visualization:**

To interpret the feature space learned by the model, t-Distributed Stochastic Neighbor Embedding (t-SNE) is utilized. This algorithm reduces the high-dimensional embedding vectors to two dimensions for visualization purposes, allowing us to observe the clustering of the different classes and the separability of the anomalies from the normal instances.

3. Experimental Evaluation

3.1 Methodology

Criteria for Evaluation:

The evaluation of the model is based on its ability to accurately classify chest X-ray images and detect anomalies. The specific criteria include:

- **Classification Accuracy**: The proportion of total correct predictions out of all predictions made, calculated for both the validation and test sets.
- **Precision and Recall**: Especially for anomaly detection, where false negatives and false positives have significant consequences, precision (the proportion of true positives among all positives) and recall (the proportion of true positives among actual cases) are crucial.
- **F1 Score**: The harmonic mean of precision and recall, providing a single metric for the balance between them.
- **Reconstruction Error**: For the autoencoder, the average difference between the input and reconstructed image, which is key for anomaly detection.

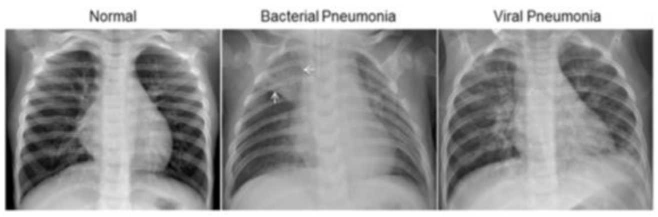
Hypotheses Tested:

1. The CNN can classify X-ray images into 'Normal', 'Pneumonia_Bacterial', and 'Pneumonia_Viral' with high accuracy.
2. The autoencoder can effectively distinguish between 'Normal' and anomalous chest X-ray images by learning to reconstruct 'Normal' images with minimal error.

4. Dataset and Features

4.1 Description

4.1.1 Context



Illustrative Examples of Chest X-Rays in Patients with Pneumonia, Related to Figure 6
The normal chest X-ray (left panel) depicts clear lungs without any areas of abnormal opacification in the image. Bacterial pneumonia (middle) typically exhibits a focal lobar consolidation, in this case in the right upper lobe (white arrows), whereas viral pneumonia (right) manifests with a more diffuse "interstitial" pattern in both lungs.

4.1.2 Content

The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).

Chest X-ray images (anterior-posterior) were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou. All chest X-ray imaging was performed as part of patients' routine clinical care.

For the analysis of chest x-ray images, all chest radiographs were initially screened for quality control by removing all low quality or unreadable scans. The diagnoses for the images were then graded by two expert physicians before being cleared for training the AI system. In order to account for any grading errors, the evaluation set was also checked by a third expert.

Acknowledgements
Training Data Set & Resources:
https://www.kaggle.com/datasets/paultimothymooney/chestxray-pneumonia/data

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

License: CC BY 4.0

Citation: http://www.cell.com/cell/fulltext/S0092-8674(18)30154-5

4.2 Data Handling

4.2.1 Color Channel

The dataset originally featured a color depth of three channels. We opted to convert it to grayscale, reducing the color depth to a single channel. This decision was informed by the nature of medical imaging, where many modalities, such as X-rays, intrinsically lack color, rendering grayscale processing not only appropriate but also without substantial loss of pertinent information. Furthermore, models trained on grayscale images are inherently less complex compared to their color image-trained counterparts, as they process fewer data channels (one instead of three). Consequently, this simplification can facilitate swifter training periods and diminish computational demands, aligning with the objectives of efficiency and effectiveness in our modelling approach.

5. Experiments/Results/Discussion

5.1 Section A – X-Ray Classification via CNN

5.1.1 Part I - Binary Classification

5.1.1.1 Data Processing

- Normalization: By diving the image pixel values by 255, the code is performing normalization which helps in stabilizing and acceleration the training of our neural network by ensuring that the input features are within a similar range.

- Reshaping: By reshaping the images we facilitate the normalization of the pixel values and ensure that the input data it in the appropriate format. As prior specified we chose to change the color dimension to be grey scaled due to the dataset being 'x-ray' images, thus by grey scaling them we are almost not losing data and by doing that we facilitate swifter training periods and diminish computational demands.

- Hyper Parameters

Loss function – we chose 'Binary cross entropy' in order to 'tackle' a classification problem between two classes.

Batch size – we used a batch size of 32 for efficient processing (after trial and error).

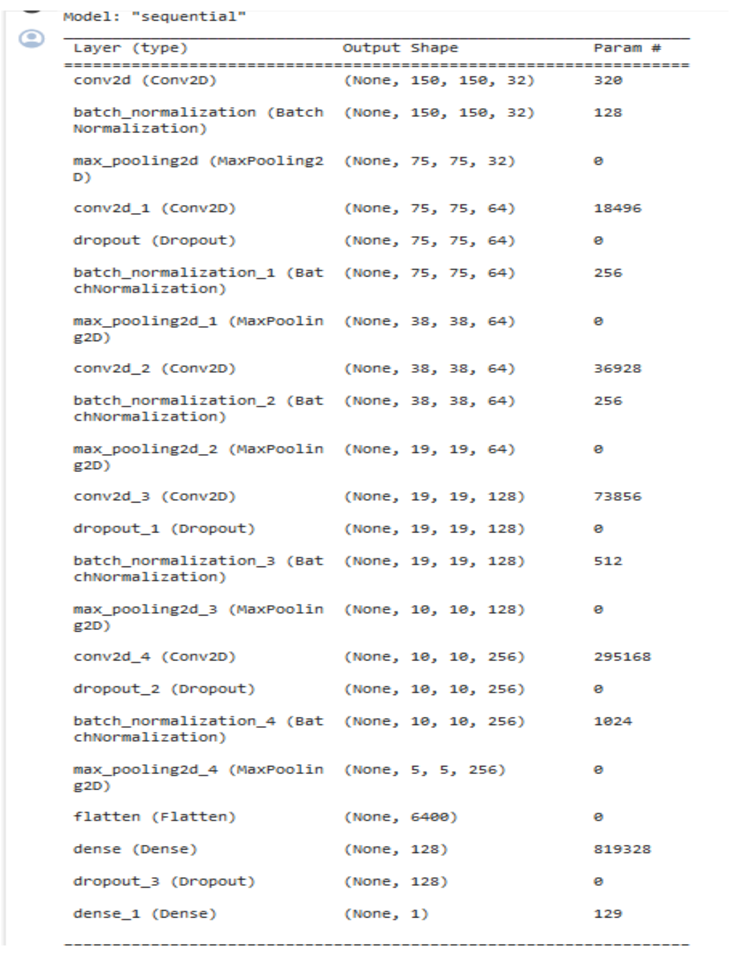
Optimizer - We utilized the 'Adam' optimizer for training our deep learning model. 'Adam' combines the advantages of two other extensions of stochastic gradient descent, namely Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). This optimizer is well-suited for tasks with large datasets and varying gradients.

5.1.1.2 Architecture

- The main objective of our project was to accurately classify X-ray images into two categories: 'Pneumonia' and 'Normal'. This binary classification task was accomplished through a deep neural network architecture.

- The architecture we constructed contains several convolutional layers followed by max-pooling layers to extract features from the input images. The final layer consists of one neuron with a sigmoid activation function, enabling binary classification by outputting probabilities.

- The architecture was designed to progressively learn intricate patterns and features from the X-ray images. We initially started with a simple architecture comprising fewer layers. Through experimentation, we expanded the network, leading to improved results.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      320

 batch_normalization (Batch  (None, 150, 150, 32)      128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 75, 75, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 75, 75, 64)        18496

 dropout (Dropout)           (None, 75, 75, 64)        0

 batch_normalization_1 (Bat  (None, 75, 75, 64)        256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 38, 38, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 38, 38, 64)        36928

 batch_normalization_2 (Bat  (None, 38, 38, 64)        256
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 19, 19, 64)        0
 g2D)

 conv2d_3 (Conv2D)           (None, 19, 19, 128)       73856

 dropout_1 (Dropout)         (None, 19, 19, 128)       0

 batch_normalization_3 (Bat  (None, 19, 19, 128)       512
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 10, 10, 128)       0
 g2D)

 conv2d_4 (Conv2D)           (None, 10, 10, 256)       295168

 dropout_2 (Dropout)         (None, 10, 10, 256)       0

 batch_normalization_4 (Bat  (None, 10, 10, 256)       1024
 chNormalization)

 max_pooling2d_4 (MaxPoolin  (None, 5, 5, 256)         0
 g2D)

 flatten (Flatten)           (None, 6400)              0

 dense (Dense)               (None, 128)               819328

 dropout_3 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

_________________________________________________________________
```

5.1.1.3 Addressing Data Imbalance and Overfitting Prevention

- In our dataset, there was an imbalance between 'Pneumonia' and 'Normal' classes, which could lead to biased predictions. To mitigate this, we employed stratified splitting during data preprocessing.

- **Overfitting**, a common concern in deep learning, was addressed through several strategies:

- **Data Augmentation -** To enrich our training dataset and improve model generalization, we employed data augmentation techniques. Data augmentation involves applying random transformations to the existing dataset, creating variations of the original images without altering their underlying features.
    - Our data augmentation strategy included:
        - Rotation by 30 degrees
        - Zoom range of 0.2
        - Horizontal and vertical shifts of 0.1
        - Horizontal flipping

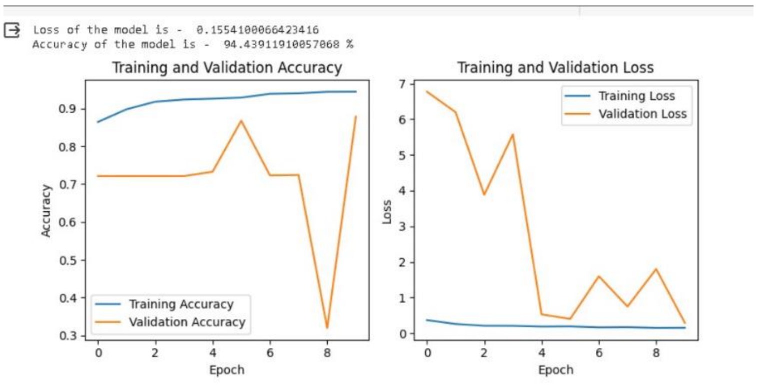
- **Dropout Layers**: We included dropout layers after some convolutional layers to prevent over-reliance on certain features and enhance generalization.

- **Learning Rate Reduction:** Additionally, to improve model performance and prevent overshooting the minima, we employed a learning rate reduction technique. Specifically, we utilized the 'ReduceLROnPlateau' callback from Keras, which reduces the learning rate by a factor when a monitored metric (in our case, validation loss) has stopped improving.

- Validation Set Expansion:
Recognizing the importance of a robust validation set, we expanded the validation set size by splitting a portion from the training data. This larger validation set provided a better estimation of model performance on unseen data, helping to detect potential issues with generalization early on.

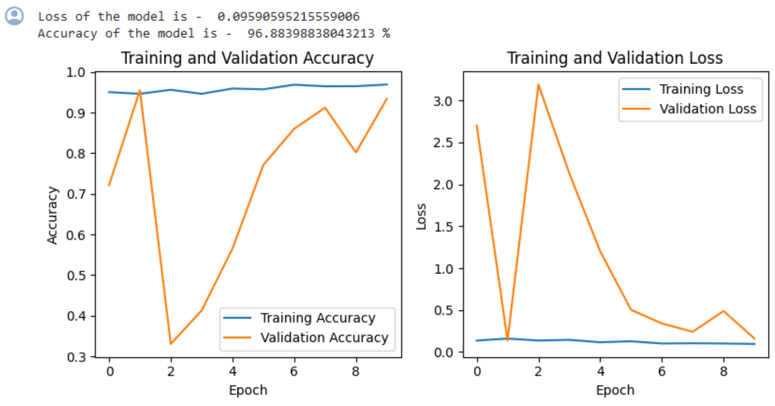
5.1.1.4 Model Evolution - Our approach to model building followed an iterative process. We began with a simpler architecture and gradually increased its complexity based on experimentation and evaluation of results. **-** Initially, the model consisted of a few convolutional layers. After observing its performance, we added more layers and included dropout for regularization. **-** The evolution of the architecture was guided by the desire to improve classification accuracy and address potential issues such as underfitting and overfitting.

- As we can see the model seems to be doing a good job of identifying both normal and pneumonia cases with high accuracy, precision, recall and F1 scores. However there is still a 21% FNR of pneumonia classification. This could stem from the graph results – even though the training accuracy and loss curves are quite smooth, the volatileness of the validation accuracy my suggest that the model is struggling to generalize to new data which can explain the relative high FNR
  -
  - First plot



```
Loss of the model is -  0.1554100066423416
Accuracy of the model is -  94.43911910057068 %
```

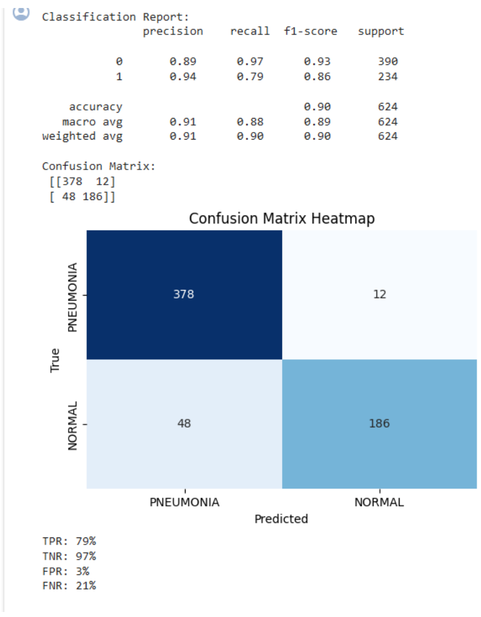
  - After ‘messing’ with the data augmentation.



```
Loss of the model is -  0.09590595215559006
Accuracy of the model is -  96.88398838043213 %
```

As we can see in the second plot the training and val accuracy and loss are approaching to the same points a bit better than the first plot.

5.1.1.5 Results Analysis - The model exhibited promising performance in classifying X-ray images as 'Pneumonia' or 'Normal'. However, there remains a 21% False Negative Rate (FNR) in pneumonia classification, indicating areas for further improvement. **-** Analyzing the training and validation loss and accuracy plots revealed insights into the model's behavior. The volatility in validation accuracy suggests that the model may struggle to generalize to new data, contributing to the relatively high FNR. **-** By adjusting data augmentation strategies, particularly reducing augmentation, we observed improvements in the model's ability to generalize while maintaining accuracy.



```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.97      0.93       390
           1       0.94      0.79      0.86       234

    accuracy                           0.90       624
   macro avg       0.91      0.88      0.89       624
weighted avg       0.91      0.90      0.90       624

Confusion Matrix:
 [[378  12]
 [ 48 186]]
```

```
TPR: 79%
TNR: 97%
FPR: 3%
FNR: 21%
```

5.1.2 <u>Part II – Categorial Classification</u>

5.1.2.1 Data Loading and Processing:

- Instead of splitting the data between 2 classes like in section 5.1.1 we needed to split the data to three classes – ‘Normal’, ‘Viral’, ‘Bacterial’, thus changing our problem from ‘Binary classification’ to ‘Categorical classification’.
- Images were traversed in the PNEUMONIA folder to differentiate between Bacterial and Viral pneumonia cases based on image names.
- The images were resized to 150x150 pixels and converted to grayscale to reduce computational complexity while retaining essential information.
- Pixel values were normalized to a range of [0, 1], ensuring uniformity and aiding in model convergence during training.
- We utilized the same techniques as in the previous model (data augmentation, grey scaling, batch sizes, and reshaping)

5.1.2.2 Architecture:

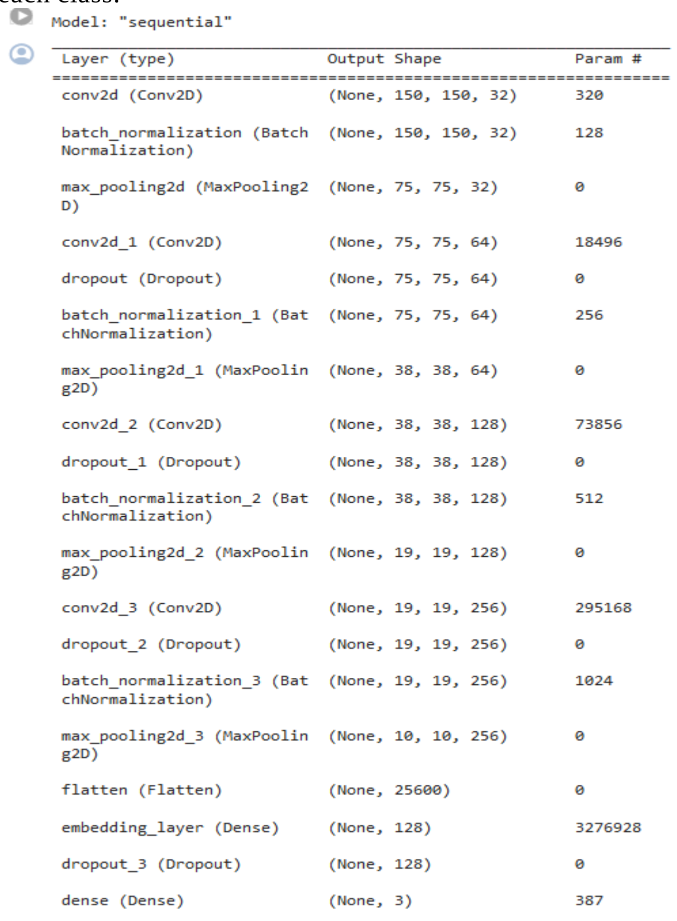
- For the multiclass classification task, we designed a convolutional neural network (CNN) architecture with three neurons in the output layer, each representing one of the three classes.

- The architecture includes several convolutional layers with increasing depth, followed by max-pooling layers to extract features and reduce spatial dimensions.

- Dropout layers were introduced after convolutional layers to prevent overfitting by randomly dropping a fraction of neuron units during training.

- Batch normalization was applied to normalize the inputs of each layer, improving convergence speed and stability.

- The final layer is a dense layer with a softmax activation function, producing probabilities for each class.

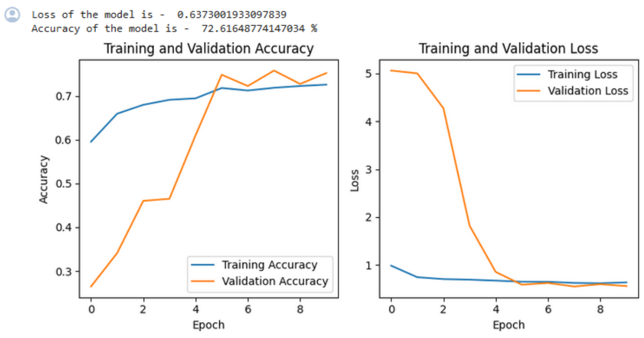


```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      320

 batch_normalization (Batch  (None, 150, 150, 32)      128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 75, 75, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 75, 75, 64)        18496

 dropout (Dropout)           (None, 75, 75, 64)        0

 batch_normalization_1 (Bat  (None, 75, 75, 64)        256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 38, 38, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 38, 38, 128)       73856

 dropout_1 (Dropout)         (None, 38, 38, 128)       0

 batch_normalization_2 (Bat  (None, 38, 38, 128)       512
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 19, 19, 128)       0
 g2D)

 conv2d_3 (Conv2D)           (None, 19, 19, 256)       295168

 dropout_2 (Dropout)         (None, 19, 19, 256)       0

 batch_normalization_3 (Bat  (None, 19, 19, 256)       1024
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 10, 10, 256)       0
 g2D)

 flatten (Flatten)           (None, 25600)             0

 embedding_layer (Dense)     (None, 128)               3276928

 dropout_3 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 3)                 387
```

5.1.2.3 Results Analysis

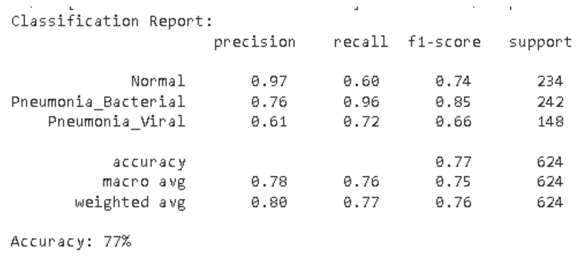
- The CNN model demonstrated promising results in classifying X-ray images into three categories: Normal, Bacterial Pneumonia, and Viral Pneumonia.
- The training and validation loss and accuracy were monitored over epochs, revealing the model's learning progress.
- The inclusion of three neurons in the output layer allowed the model to predict probabilities for each class, providing a more detailed understanding of the classification confidence.
- By adjusting data augmentation techniques and implementing a learning rate reduction strategy, the model achieved improved performance and generalization.

Results:

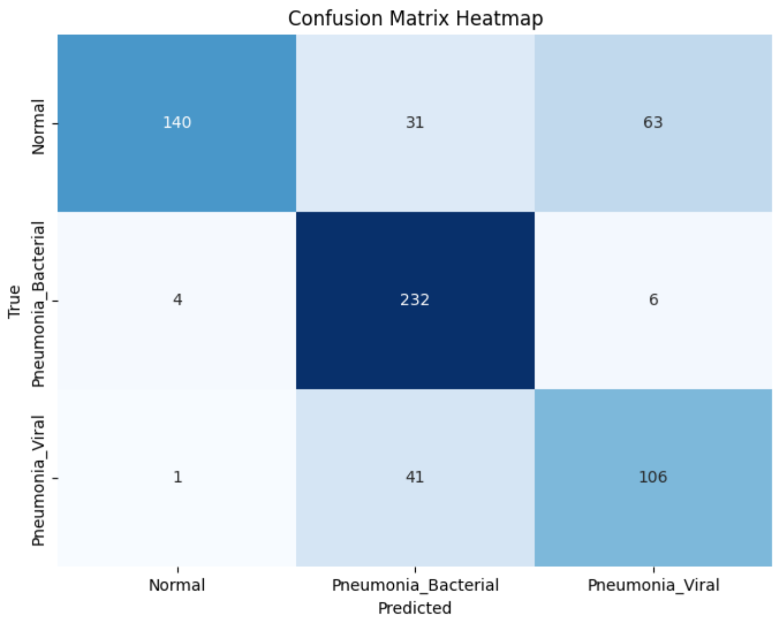


```
Loss of the model is -  0.6373001933097839
Accuracy of the model is -  72.61648774147034 %
```

As we can see from the ‘plot’, the training and val accuracy and loss curves are approaching to the same points quite nicely, but due to the big imbalance between the classes sizes we can see that the model had a difficulty to reach better results than 72% accuracy.



```
Classification Report:
                     precision    recall  f1-score   support

             Normal       0.97      0.60      0.74       234
Pneumonia_Bacterial       0.76      0.96      0.85       242
    Pneumonia_Viral       0.61      0.72      0.66       148

           accuracy                           0.77       624
          macro avg       0.78      0.76      0.75       624
       weighted avg       0.80      0.77      0.76       624

Accuracy: 77%
```



5.2 Section B - Embedding Vector and KNN Classifier

5.2.1 Classifying New Images

5.2.1.1 Embedding Vectors

- After successfully training a neural network for the multiclass classification of X-ray images, the next step was to explore how new, unseen images could be classified.
- We utilized the trained CNN model and extracted the embedding vectors from the 'embedding_layer', which served as a compact representation of the image features.
- The 'embedding_layer' captured essential features of the X-ray images, enabling us to create a numerical representation of each image.

5.2.1.2 Loading and Preprocessing New Images

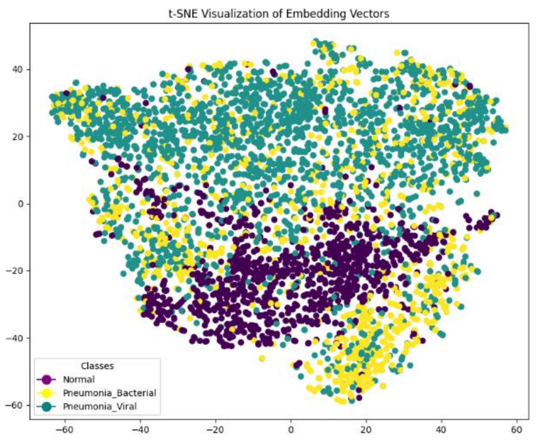
- To demonstrate the classification process, we loaded a new image from the 'test' folder. Additionally, users had the option to upload their own image for classification.
- The new image was preprocessed to match the dimensions and normalization used during training. Grayscale conversion, resizing to 150x150 pixels, and normalization to [0, 1] were performed.

5.2.1.3 Classification Using k-NN

- Once the new image was preprocessed, we obtained its embedding vector using the trained model.
- Simultaneously, we collected the flattened embedding vectors from the training set to build a reference dataset for the k-NN classifier.
- A k-Nearest Neighbors (k-NN) classifier was trained on the flattened embedding vectors from the training set.
- The k-NN classifier utilized the Euclidean distance metric to find the k nearest neighbors to the new image's embedding vector in the reference dataset.
- The predicted class label for the new image was determined by a majority vote from the k nearest neighbors.

5.2.1.4 Visualization of Classification

- To visualize the effectiveness of the embedding vectors in capturing image similarities, we employed the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm.
- t-SNE is a technique for dimensionality reduction, particularly well-suited for visualizing high-dimensional data in lower dimensions.
- The flattened embedding vectors from the training set were transformed using t-SNE to create a 2D representation of the data.
- Each point in the t-SNE plot represents an embedding vector, colored according to its corresponding class label.
- This visualization allowed us to observe clusters formed by the embedding vectors of different classes, providing insights into the model's ability to separate and classify images.



5.2.1.5 **Results and Discussion**

- Upon uploading and preprocessing the new image, the k-NN classifier successfully predicted its class.
- The predicted class label provided by the k-NN classifier was based on the nearest neighbors in the embedding space.
- This approach demonstrates how a trained CNN model can be used to extract meaningful features (embedding vectors) from images, enabling downstream tasks like classification with k-NN.
- The t-SNE visualization further validated the effectiveness of the embedding vectors. Clusters corresponding to different classes were clearly visible, indicating that the embedding vectors captured important characteristics for classification.
- Overall, this methodology showcased a pipeline for classifying new images using a pretrained CNN model, extracting embedding vectors, and employing a k-NN classifier for prediction.

5.2.1.6 **Future Considerations**

- While the k-NN approach provided successful classification, there are avenues for further exploration.
- Future work could involve fine-tuning the k-NN classifier hyperparameters for optimal performance.
- Additionally, investigating other dimensionality reduction techniques or deep learning-based approaches for feature extraction could enhance classification accuracy.
- The t-SNE visualization highlighted the separability of classes in the embedding space. This insight could guide improvements in the model architecture or data preprocessing to better distinguish between classes.

5.2.2 **Conclusion**

- In conclusion, we demonstrated a comprehensive pipeline for classifying new X-ray images into multiple classes using a combination of a CNN model, embedding vectors, and a k-NN classifier.
- The process involved preprocessing new images, extracting embedding vectors from the CNN model, training a k-NN classifier on the reference dataset, and visualizing the embedding space using t-SNE.
- Through this approach, we achieved successful classification of new images while gaining insights into the underlying structure of the embedding space.
- This methodology has practical applications in medical image analysis, allowing for efficient and accurate classification of X-ray images into various disease categories.

5.3 **Anomaly Detection**

- In this section we were required to detect ‘anomalies’ in our data set, where ‘Normal’ is labeled ‘normal’ and Pneumonia is labeled ‘abnormal’.

- **Data processing:**
  For the data processing we chose a similar approach to section 5.1.1 but added another split to distinguish between ‘normal’ and ‘abnormal’ data.

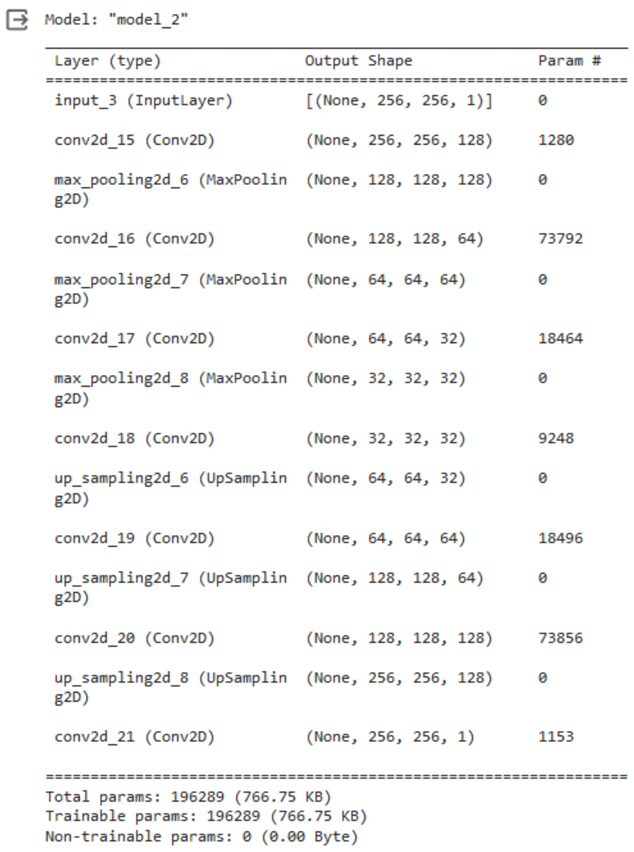
- **Autoencoder Architecture**:

  - The encoder compresses the input data into a latent space representation, while the decoder reconstructs the input from this representation.
  - In this architecture, the encoder consists of convolutional layers followed by max-pooling layers, reducing the spatial dimensions of the input.
  - The decoder consists of upsampling layers followed by convolutional layers, aiming to reconstruct the original input size.
  - The model is compiled with the Adam optimizer and binary cross-entropy loss function.We chose Binary cross-entropy because we thought it is more suitable for binary classification problems.

Training Data Selection:

- Only 'NORMAL' class images are used for training the autoencoder. This ensures that the model focuses on learning the normal data distribution without being influenced by anomalies present in other classes.
- Similarly, 'NORMAL' class images are also used for validation during training.

Model Architecture:



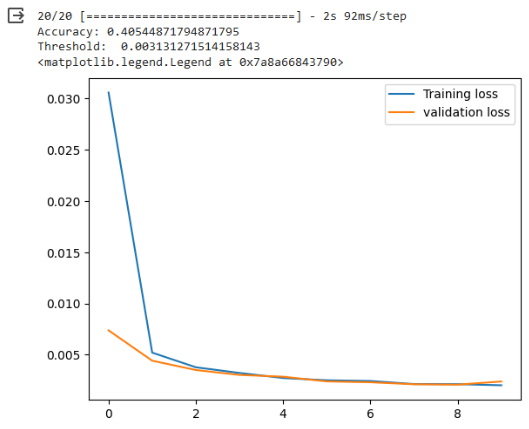
```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 256, 256, 1)]     0

 conv2d_15 (Conv2D)          (None, 256, 256, 128)     1280

 max_pooling2d_6 (MaxPoolin  (None, 128, 128, 128)     0
 g2D)

 conv2d_16 (Conv2D)          (None, 128, 128, 64)      73792

 max_pooling2d_7 (MaxPoolin  (None, 64, 64, 64)        0
 g2D)

 conv2d_17 (Conv2D)          (None, 64, 64, 32)        18464

 max_pooling2d_8 (MaxPoolin  (None, 32, 32, 32)        0
 g2D)

 conv2d_18 (Conv2D)          (None, 32, 32, 32)        9248

 up_sampling2d_6 (UpSamplin  (None, 64, 64, 32)        0
 g2D)

 conv2d_19 (Conv2D)          (None, 64, 64, 64)        18496

 up_sampling2d_7 (UpSamplin  (None, 128, 128, 64)      0
 g2D)

 conv2d_20 (Conv2D)          (None, 128, 128, 128)     73856

 up_sampling2d_8 (UpSamplin  (None, 256, 256, 128)     0
 g2D)

 conv2d_21 (Conv2D)          (None, 256, 256, 1)       1153

=================================================================
Total params: 196289 (766.75 KB)
Trainable params: 196289 (766.75 KB)
Non-trainable params: 0 (0.00 Byte)
```

Results: Despite training the autoencoder model, its performance in distinguishing between pneumonia and normal chest X-ray images on the test set was suboptimal. The classification report indicates that while the model achieved a relatively high recall for pneumonia cases (0.96), its precision for normal cases was moderate (0.38), leading to an overall moderate F1-score for pneumonia detection (0.54).

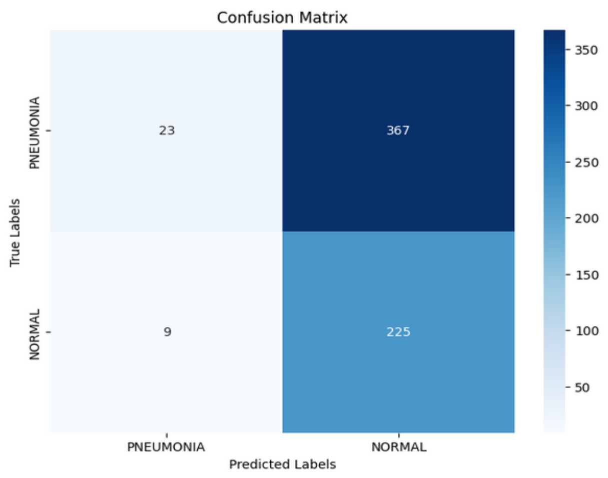
These results suggest that the autoencoder might not have learned sufficiently discriminative features to distinguish between images with pneumonia and normal images effectively. This could be due to various factors such as model complexity, dataset size, or the inherent difficulty of the task.

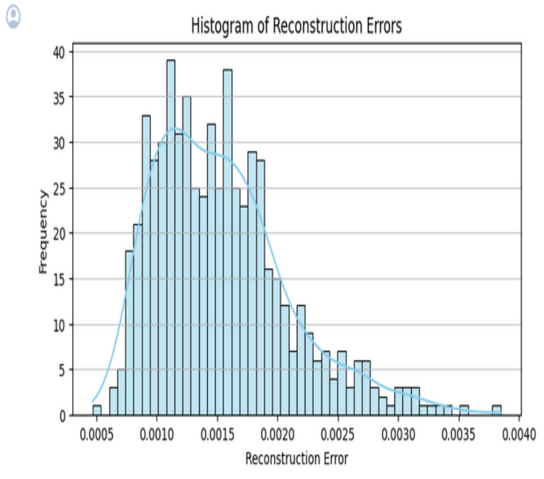
It is important to note that while the autoencoder approach has its limitations in this context, it still provides valuable insights into the challenges of anomaly detection in medical imaging. Further improvements to the model architecture, training process, or dataset curation may be necessary to enhance its performance for this task.

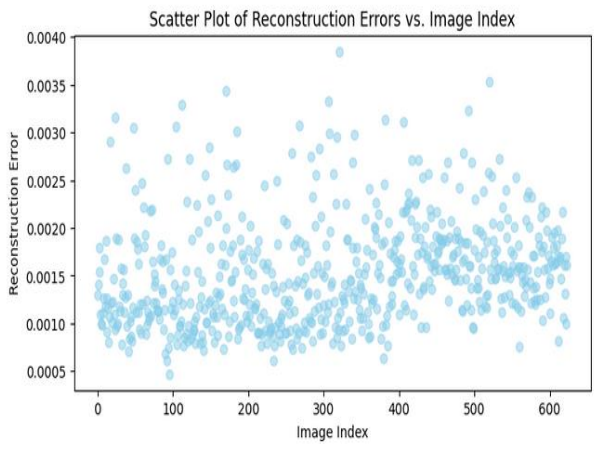
In the project, various strategies were explored to improve the performance of the anomaly detection autoencoder. However, despite implementing data augmentation techniques and simplifying the model architecture, the results did not show significant improvement. This suggests that the autoencoder may not have been able to effectively distinguish between normal and pneumonia images in the dataset.

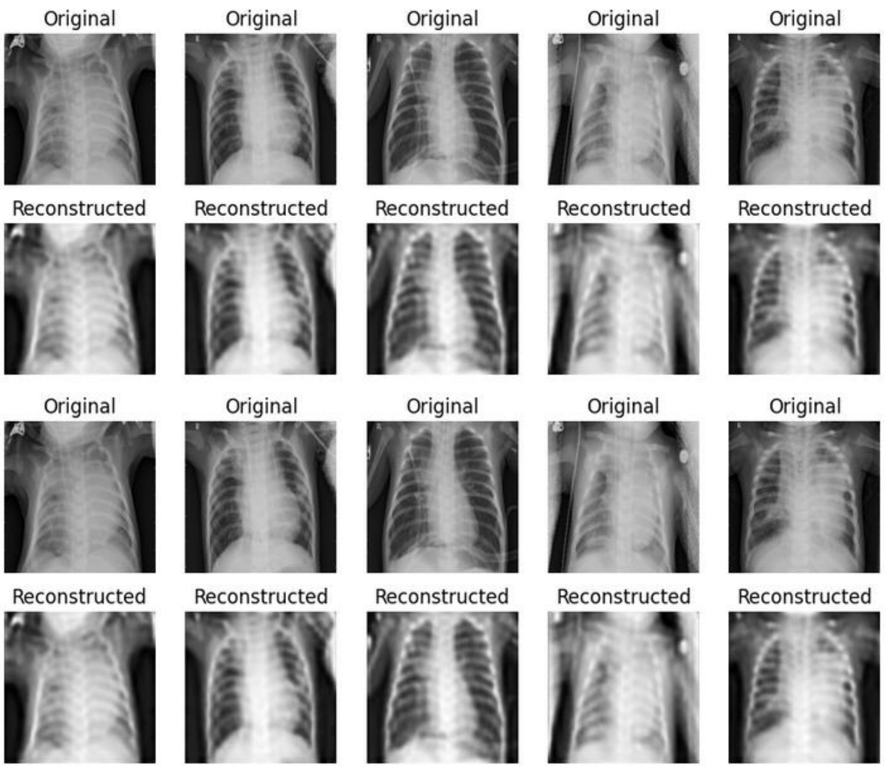


```
20/20 [==============================] - 2s 92ms/step
Accuracy: 0.40544871794871795
Threshold:  0.003131271514158143
<matplotlib.legend.Legend at 0x7a8a66843790>
```








Original
Original
Original
Original
Original
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Original
Original
Original
Original
Original
Reconstructed
Reconstructed
Reconstructed
Reconstructed
Reconstructed