

Machine Learning for Health Care

**PROJECT 1**

**Interpretable and Explainable Classification**

**for Medical Data**

**Students**

Teodora Petrovic, 23-745-797

Rohit Koonireddy, 20-622-924

Kristijan Bundaleski, 23-742-331

Zurich, April 2024

# Heart Disease Prediction Dataset

## Dataset Description

The dataset comprises the following attributes for each individual:

* Age: The age of the individual.
* Sex: The gender of the individual (Male or Female).
* Chest Pain Type: The type of chest pain experienced by the individual.
* Resting Blood Pressure: The individual's blood pressure at rest.
* Cholesterol: The cholesterol level of the individual.
* Fasting Blood Sugar: The fasting blood sugar level of the individual.
* Resting Electrocardiographic Results: Results of the resting electrocardiogram.
* Maximum Heart Rate: The maximum heart rate achieved during exercise.
* Exercise-Induced Angina: If the individual experienced angina during exercise.
* ST Depression: ST depression induced by exercise relative to rest.
* ST Slope: The slope of the peak exercise ST segment.
* Heart Disease: The presence or absence of heart disease (1 indicates presence)

## The Exploratory Data Analysis

A screenshot of a computer

Description automatically generatedExploratory Data Analysis (EDA) was conducted to understand the characteristics of the heart disease prediction dataset. Various visualizations, including violin plots and histograms, were used to explore the distribution of features and their relationships with the target variable. Sample view of the dataset is given in Figure 1.

*Figure 1. Sample view of the dataset*

A green and orange paddles

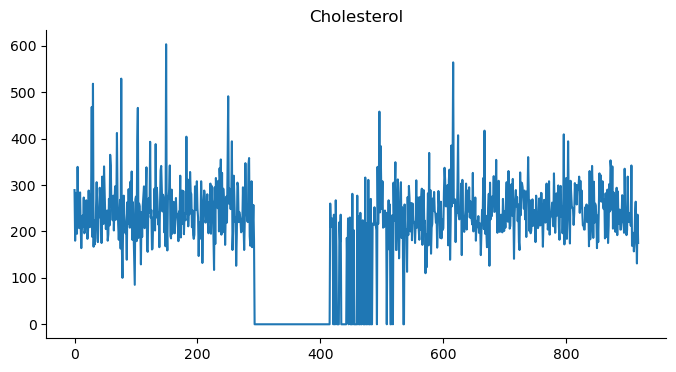
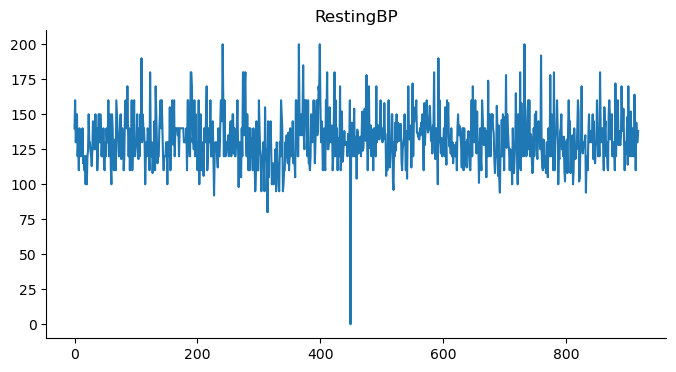
Description automatically generatedDistribution of exercise angina, resting ECG and chast pain type according to age is shown in Figure 2. Same was done for attribute sex.

*A close-up of several different colored shapes

Description automatically generatedA diagram of different colors

Description automatically generated*

*Figure 2. Distribution of attributions*

Distribution of Cholesterol and RestingBP is shown in Figure 3.

*Figure 3. Distribution of cholesterol and resting BP*

Distribution of Cholesterol and RestingBP is shown in Figure 3.

## Logistic Lasso Regression

We trained a logistic regression model to predict the presence of heart disease using the provided dataset. First, we split the data into features and target variables, encoding categorical features and dividing the dataset into training and testing sets. Next, the logistic regression model was fitted to the training data. Subsequently, we used the trained model to predict heart disease presence on the test set. Finally, we evaluated the model's performance using accuracy and F1-score metrics, assessing the proportion of correct predictions and the balance between precision and recall, respectively.

A graph of blue rectangular shapes

Description automatically generated with medium confidence

## Multi-Layer Perceptrons

Multi-Layer Perceptrons (MLPs), a class of feedforward neural networks, were employed for heart disease prediction. MLPs are capable of learning complex relationships between features and the target variable. The model's hyperparameters, such as the number of hidden layers and neurons, activation functions, and optimization algorithms, were fine-tuned to achieve optimal performance. Evaluation metrics such as accuracy and F1-score were used to assess the model's predictive capability.

We use the following parameters to train the stand alone MLP model from

„from sklearn.neural\_network import MLPClassifier“

A screen shot of a computer code

Description automatically generated

Results:

**Accuracy: 0.8641304347826086, F1-score: 0.8756218905472637**

We see the following shapely values for each feature:

A graph with red and blue bars

Description automatically generated A graph with red and blue bars

Description automatically generated

*Figure x. Shapely values of each feature for figures for „no heart diease“(on left) and „heart failure“(on right)*

We see that for both the classifications ChestPainType, MaxHR, ExerciseAngina, Cholestrol, Old Peak. While the slope, Sex, FastingBS, clearly show signs of difference.

A graph with red and blue dots

Description automatically generated A graph with red and blue dots

Description automatically generated

*Figure x. Shapely values of 2 samples for „no heart diease“(on left) and „heart failure“(on right)*

## Neural Additive Models

Neural Additive Models (NAMs) offer an interpretable framework for understanding the predictions of complex neural network models. By decomposing the model predictions into additive components, NAMs provide insights into the contribution of each feature to the overall prediction. Shapely values, a popular technique for interpreting machine learning models, were employed to quantify the impact of individual features on the model predictions. Summary plots generated from Shapely values help visualize feature importance and aid in understanding the model's behavior.

A graph of different colored dots

Description automatically generated with medium confidence

Figure 2: Shapely values for each paramter for the Neural Additive Models - for heart failure

# 

# Pneumonia Prediction Dataset

## Exploratory Data Analysis

### Label distribution and qualitative description of the data

The dataset is structured into three main folders: train, val and test. Within each of these folders, there are subfolders representing the two categories of X-Ray images: Pneumonia and Normal. In total, there are 5,856 X-Ray images in JPEG format. Their distribution is shown in table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Train | Validation | Test | Total |
| Normal | 1341 | 8 | 234 | 1583 |
| Pneumonia | 3875 | 8 | 390 | 4273 |

*Table 1. Label distribution*

A close-up of x-ray images

Description automatically generatedThe normal chest X-ray typically shows clear lungs without any areas of abnormal opacification. However, in cases of bacterial pneumonia, distinct white areas may appear on the image, indicating focal lobar consolidation, as illustrated in Figure 1. These differences can be discovered even by an untrained observer.

*Figure 1. Distinct white areas are marked*

A close-up of an x-ray of a chest

Description automatically generatedIt is also easy to notice differences if pneumonia has progressed significantly, as depicted in Figure 2.

*Figure 2. Disease progressed significantly*

A x-ray of a person's chest

Description automatically generatedOn the other hand, distinguishing between healthy and diseased samples is often more challenging in cases of pneumonia in early stages. This is because viral pneumonia is often manifested with a more diffuse ‘‘interstitial’’ pattern in both lungs, which can closely resemble the appearance of a healthy chest X-ray. I was not able to see these differences. Less pronounced visual differences between healthy and disease samples are shown in Figure 3.

*Figure 3. Samples that are dificult to classify by untrained eye*

However, in most cases it is very difficult for someone who is not an expert in the field of medicine to see visual differences between healthy and disease samples as well as differences between virus and bacterial pneumonia.

### Source of bias that could influence model performance

There are significantly more pneumonia samples than healthy samples, as shown in Table 1. The potential source of bias that could influence model performance would be bias towards the majority class (pneumonia class). In a classification task with imbalanced classes, a model might tend to predict the majority class more frequently due to its prevalence in the dataset. Consequently, the model may have lower sensitivity or recall for the minority class (normal images) and higher false negative rates for detecting normal cases.

### Data preprocessing

The data has undergone cropping, resizing, and normalization. This was executed with the aid of torchvision library.

* torchvision.transforms.Resize
* torchvision.transforms.CenterCrop
* torchvision.transforms.Normalize

For normalization step, we forwarded to torchvision.transforms.Normalize the previously calculated mean and std of the dataset. To address bias explained earlier, we utilized data augmentation techniques on balanced dataset. Data augmentation is creating additional training examples by applying transformations to the existing data. This can help us balance the dataset and improve model generalization. We implemented intensity transformations (sharpness adjustments, random noise) and positional transformations. We applied transformations by using functionalities from torchvision library:

* torchvision.transforms.RandomApply
* torchvision.transforms.RandomAffine
* torchvision.transforms.GaussianBlur
* torchvision.transforms.RandomHorizontalFlip
* torchvision.transforms.RandomVerticalFlip
* torchvision. transforms.RandomAdjustSharpness

A computer screen shot of code

Description automatically generatedImplementation of these transformations can be seen in Figure 4.

*Figure 4. Transforming images*

A computer code with numbers and symbols

Description automatically generatedPart of the code in which we add noise to the image is shown in Figure 5.

*Figure 5. Adding noise to the image*

A close-up of a radiograph

Description automatically generatedFigure 6 shows one image after cropping, then after resizing to (3, 222, 224), and after all other transformations.

*Figure 6. Image preprocessing and augmentations*

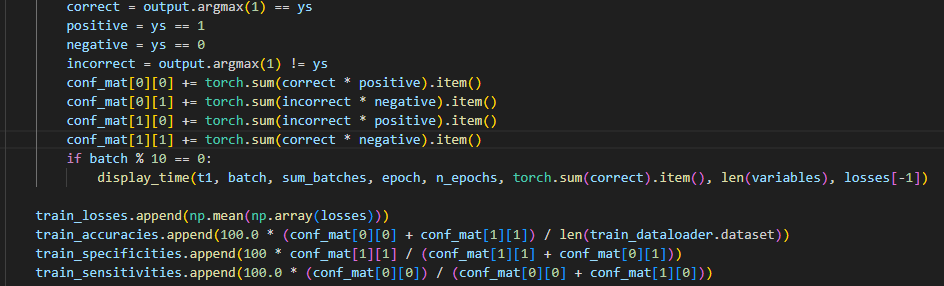
## CNN Classifier

### Network architecture and implementation

A computer screen with text

Description automatically generatedWe are using ResNet18, which was pretrained on ImageNet dataset. In python script dataset.py, in def \_\_getitem\_\_(self, index) we are processing images. We load one image, do the preprocessing and transformations and return it together with its label (0 for healty sample, 1 for pneumonia sample). File with relative path Project1\code\CNN\main.py is the heart of task 2 and used to split data (without test data) into train and validation data and call train function. Test data is unchanged. Training is done in epochs, and data is divided in batches. We are making predictions if the person has disease or not, calculate loss, back propagate it and update the weights. This is shown in Figure 7.

*Figure 7. Epoch and time tracking*

After every epoch we are appending loss, accuracy, specificity and sensitivity using the coeficiency matrix. We do this for train, validation and test. This is shown in figure 8.

*Figure 8. Calculating accuracy, specificity and sensitivity*

Then, we plot the results using code from Figure 9.

*Figure 9. Plotting the results*

A screen shot of a computer

Description automatically generatedTo keep track of time and progress, we implemented function shown in Figure 10.

*Figure 10. Printing progress and time*

### Results

Best CNN model (Run\_20240329\_215317/model\_15.pth) which performed best with parameters shown in table 2 and achieved results presented in table 3.

|  |  |
| --- | --- |
|  | Balanced dataset |
| number of epochs | 15 |
| batch size | 64 |
| learning rate | 0.0001 |
| scheduler step size | 5 |
| scheduler gama | 0.33 |
| optimization function | sgd |
| loss function | cross entropy |
| weight decay | 0.0005 |

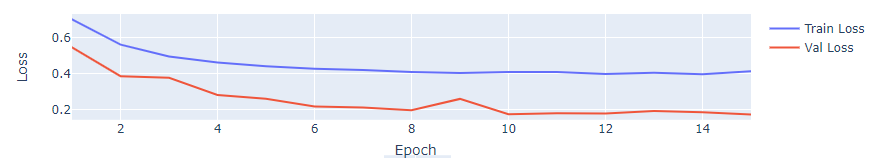
*Table 2. Best parameters for training the CNN*

|  |  |
| --- | --- |
|  | Balanced dataset |
| Accuracy | 86.69 |
| Specificity | 79.91 |
| Sensitivity | 90.77 |

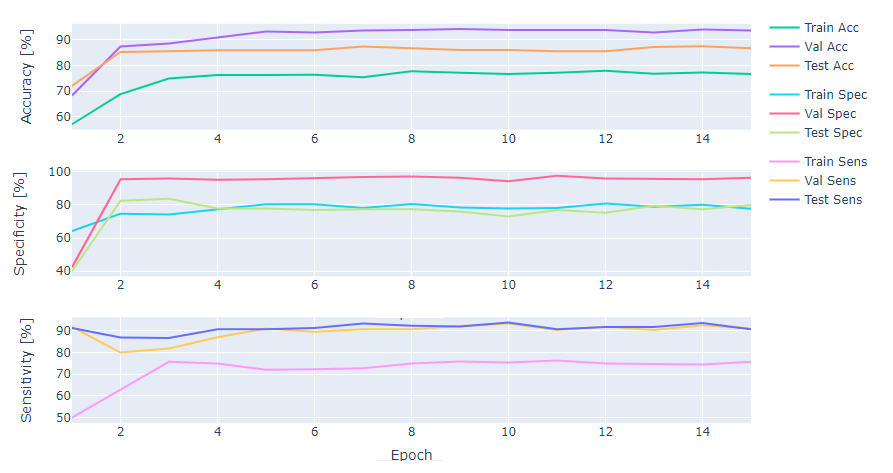
*Table 3. Best results*

We have calculated the performance using formulas (1), (2) and (3).

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |
|  | (3) |

Loss function on train and validation data is given in Figure 7.

*Figure 7. Loss function*

**Performance parameters, such as accuracy, specificity and sensitivity during epochs is shown in Figure 8.

*Figure 8. accuracy, specificity and sensitivity during epochs*

## Integrated Gradients

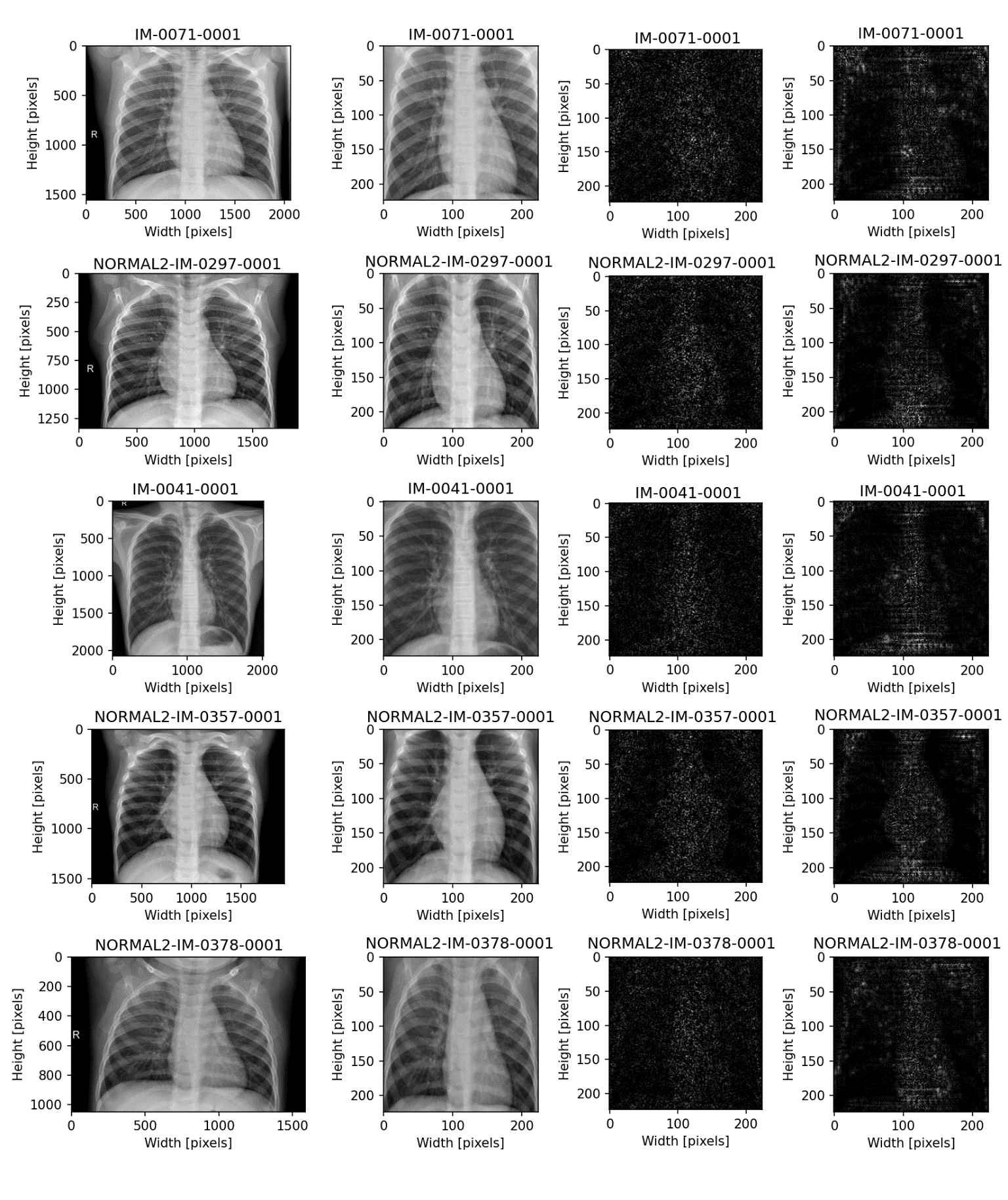
### Implementation

A screen shot of a computer program

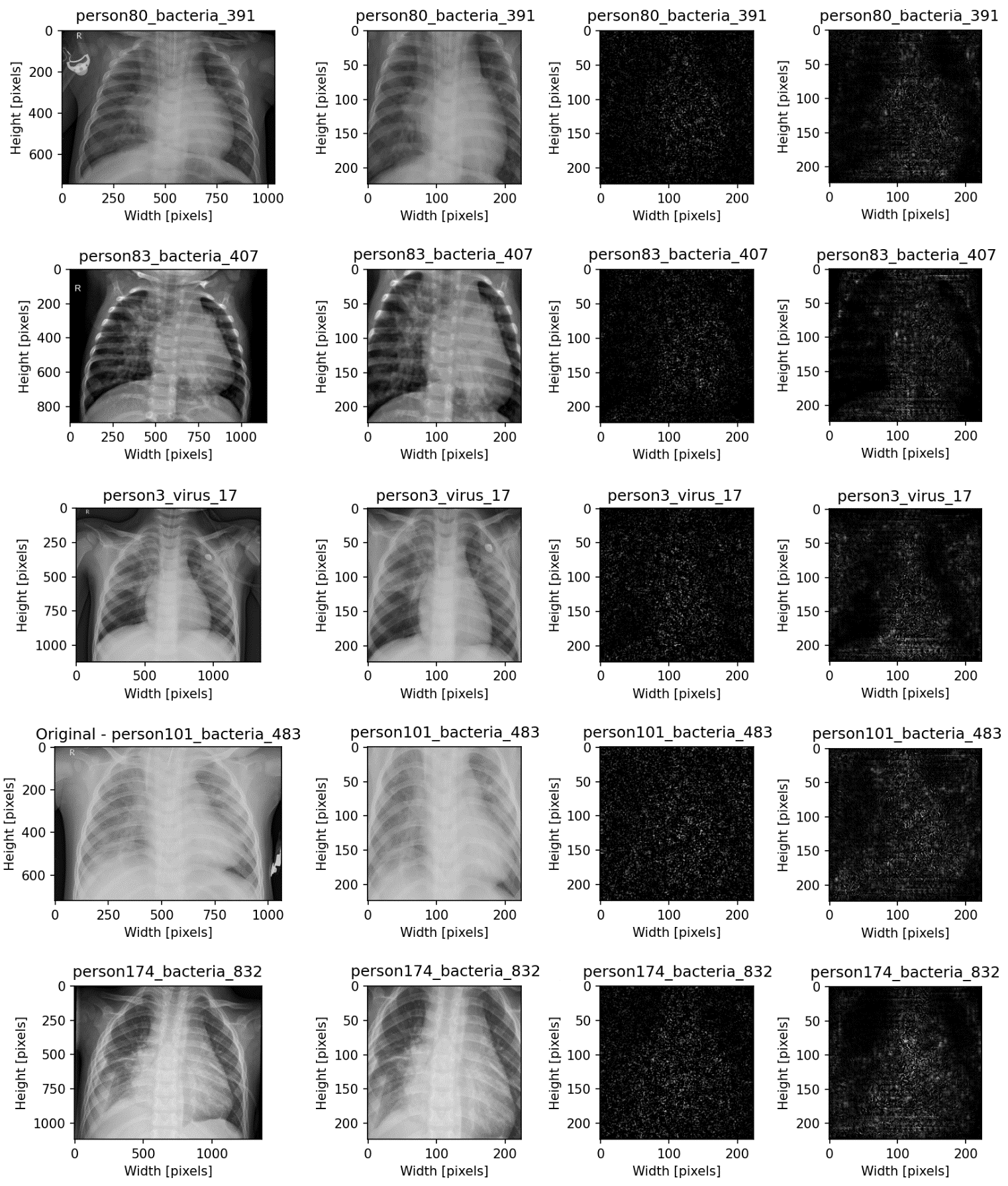
Description automatically generatedFor the implementation of integrated gradients we have modified the publicly available code [1]. The implementation is pytorch implementation of the paper Axiomatic Attribution for Deep Networks [2]. Most important part of the code are functions get\_grads and integrated\_gradients in file integrated\_gradients.py. Screenshot of this part of the code is given in Figure 9.

*Figure 9. Integrated gradients – core code*

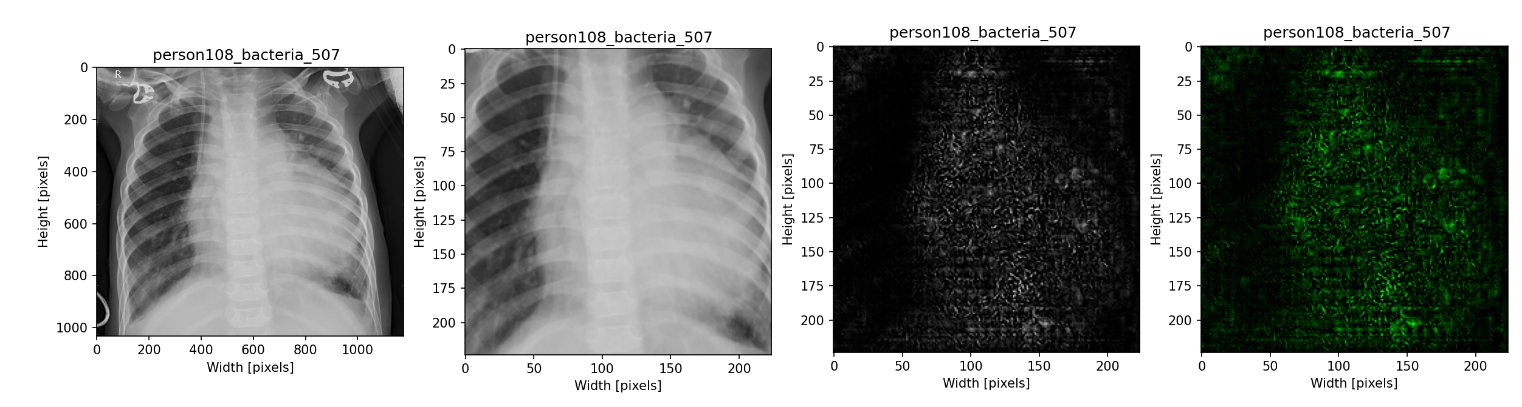
### Visualization of healthy and disease test samples

Five healthy test samples and results of using integrated gradients are shown in Figure 10. Column 3 are results when baseline is 255.0\*np.random.random(inputs.shape). Column 4 are results when using black image as a baseline.

*Figure 10. Healthy samples*

Five disease test samples and results of using integrated gradients are shown in Figure 11. Column 3 are results when baseline is 255.0\*np.random.random(inputs.shape). Column 4 are results when using black image as a baseline.

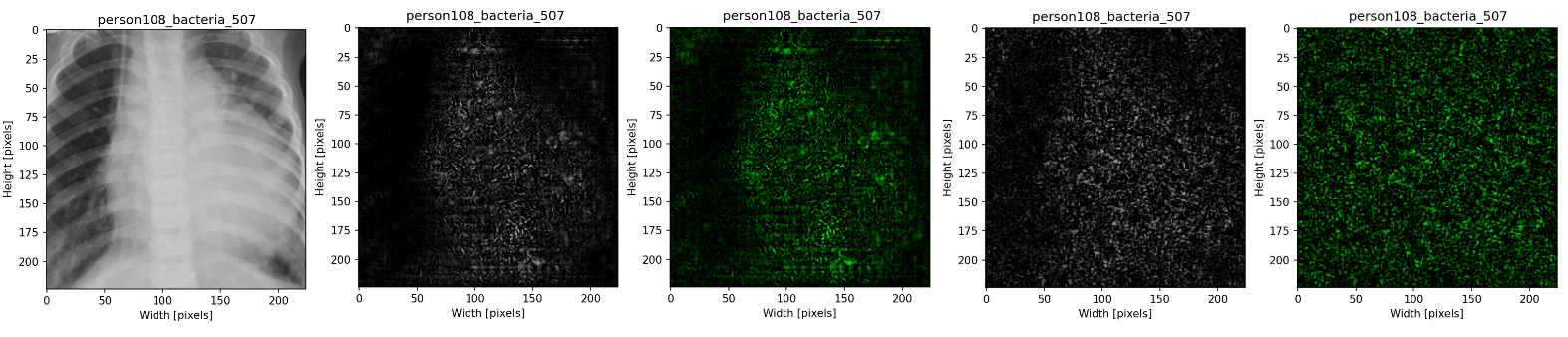
*Figure 11. Disease samples*

The attribution maps do highlight sensible regions. The model detects the white regions on the images. When the input is a pneumonia image, pneumonia is highlighted, as in Figure 12. First image is the original image, then we added cropped and resized image, followed by integrated gradient overlay and gradient overlay.

*Figure 12. Highlighted pneumonia*

If the input is a healthy sample, only the spinal cord and chest bones are highlighted as shown in Figure 10. Attributions are consistently highlighting sensible regions, but as every sample is unique, attributions are different across samples.

### Baseline choice

As stated in Axiomatic Attribution for Deep Networks [2] the choice of baseline image has a big effect on the attribution maps. The baseline should represent a complete absence of signal, so that the features that are apparent from the attributions are properties of input only, and not of the baseline. Good choice of baseline image is black image. Black image as a baseline gave us the best results. To compare results with black image as a baseline and 255.0\*np.random.random(inputs.shape) as the baseline we plotted the results in Figure 13. The comparison is also made in Figures 10 and 11.

*Figure 13. Comparison of baselines*

A computer screen with text

Description automatically generatedTo get better results, we are doing intergrated gradients multiple times and take the average of the results. Baseline can be changed in line 44. Part of the code where we implement this is given in Figure 14.

*Figure 14. Multiple integrated gradients*

## Grad-CAM

## Data Randomization Test

A close-up of x-ray images

Description automatically generated

*Figure 9. View for Integrated Gradients approach with 70% target randomization*

# General Questions

**Q1: How consistent were the different interpretable/explainable methods? Did they find similar patterns?**

The interpretable/explainable methods varied in their consistency. Traditional methods like feature visualization or activation maximization relied on directly visualizing learned features within the model, leading to inconsistent patterns depending on the specific architecture and training data. In contrast, methods like Integrated Gradients and Grad-CAM offered more consistent results by highlighting important features or regions contributing to model predictions. These methods found similar patterns across different instances, providing more reliable insights into the model's decision-making process.

**Q2: Given the “interpretable” or “explainable” results of one of the models, how would you convince a doctor to trust them? Pick one example.**

To convince a doctor to trust the interpretable/explainable results, we would provide transparency and clarity regarding the methods used and the rationale behind the results. For example, if using Grad-CAM to interpret a CNN model's predictions on chest X-ray images, we would highlight specific regions in the images that Grad-CAM identifies as indicative of pneumonia. We would explain how these regions align with known characteristics of the disease, such as opacities or infiltrates in the lungs, backed by relevant medical literature. Additionally, we would emphasize the method's consistency and its ability to provide actionable insights into the model's decision-making process, ultimately helping clinicians understand and trust the model's predictions.

**Q3: Elaborate whether the feature importances from the interpretability/explainability methods intuitively make sense to find the respective disease.**

The feature importances obtained from interpretability/explainability methods generally align with the expected characteristics of the respective disease. For example, in medical imaging, methods like Grad-CAM highlight regions of abnormality or pathology that are indicative of the diagnosed condition. These regions often correspond to known anatomical or pathological features associated with the disease, such as tumors, lesions, or inflammatory changes. By focusing on these relevant features, the interpretability/explainability methods provide intuitive insights into how the model makes predictions and help clinicians understand the underlying reasoning behind the model's decisions.

**Q4: If you had to deploy one of the methods in practice, which one would you choose and why?**

If we had to deploy one method in practice, we would choose Integrated Gradients or Grad-CAM due to their interpretability and transparency. These methods offer clear and actionable insights into the model's decision-making process by highlighting important features or regions in the input data that contribute to the predictions. This transparency is crucial in healthcare settings, where trust and understanding are paramount. Additionally, Integrated Gradients and Grad-CAM provide consistent and interpretable results, making them valuable tools for clinical decision support and aiding in the interpretation of model predictions by healthcare professionals.

# References

[1] Tianhong Dai, Weikai Yang, “integrated-gradient-pytorch” GitHub repository, 2018. Available: <https://github.com/TianhongDai/integrated-gradient-pytorch>

[2] Sundararajan, Taly, and Yan, “Axiomatic Attribution for Deep Networks”, 2017. Available: <https://doi.org/10.48550/arXiv.1703.01365>