

Machine Learning for Health Care

**PROJECT 1**

**Interpretable and Explainable Classification**

**for Medical Data**

**Students**

Teodora Petrovic, 23-745-797

Zurich, March 2024

# Heart Disease Prediction Dataset

## Exploratory Data Analysis

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

## Logistic Lasso Regression

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

## Multi-Layer Perceptrons

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

## Neural Additive Models

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

# 

# 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

To address bias explained earlier, we utilized data augmentation techniques. 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*

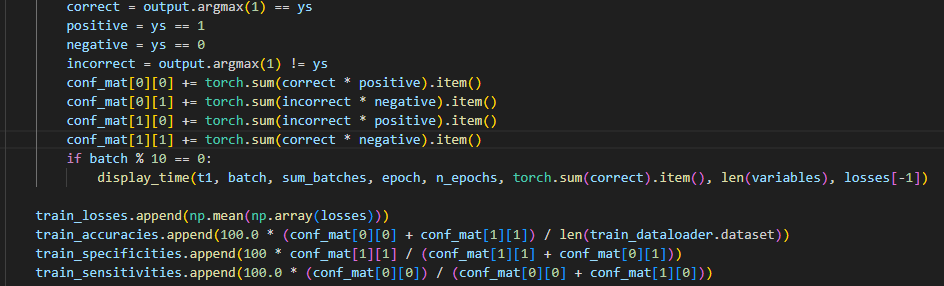
## CNN Classifier

### 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 2d\_test.py is the heart of task 2 and used to split data into train and validation data and call main 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.

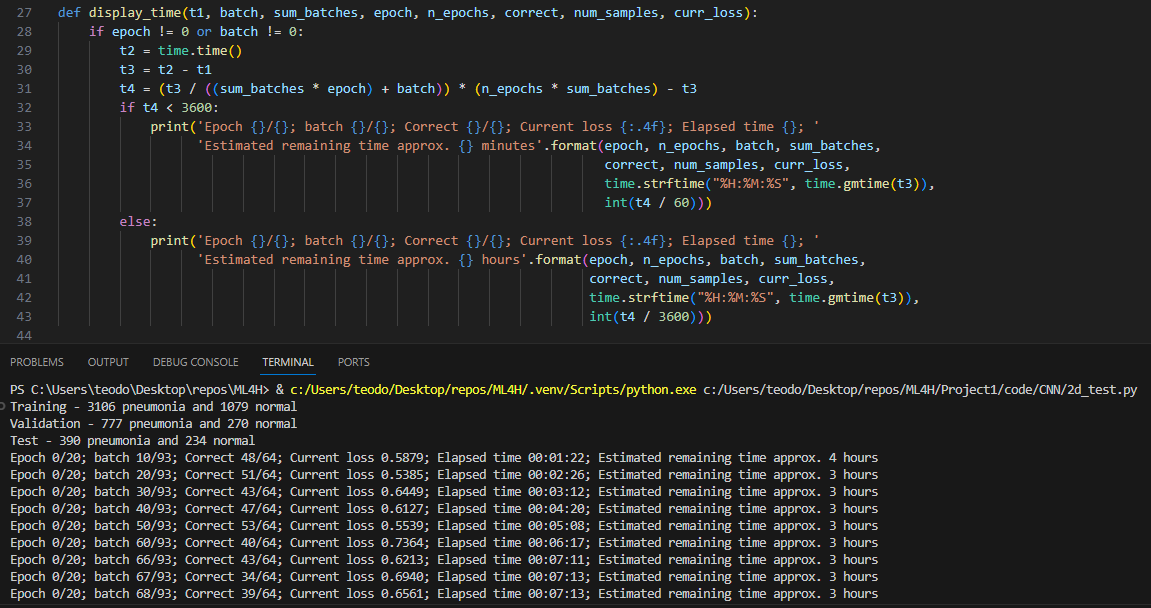
*Figure 7. Epoch and time tracking*

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*

To keep track of time and progress, we implemented function shown in Figure 10.

*Figure 10. Printing progress and time*

### Results

CNN performed best with parameters shown in table 2 and achieved results presented in table 3.

|  |  |  |
| --- | --- | --- |
|  | Unbalanced dataset | Balanced dataset |
| number of epochs | 20 | 20 |
| batch size | 64 | 64 |
| learning rate | 0.0001 | 0.0001 |
| scheduler step size | 10 | 10 |
| scheduler gama | 0.33 | 0.33 |
| optimization function | sgd | sgd |
| loss function | cross entropy | cross entropy |
| weight decay | 0.0005 | 0.0005 |

*Table 2. Best parameters for training the CNN*

|  |  |  |
| --- | --- | --- |
|  | Unbalanced dataset | Balanced dataset |
| Accuracy |  |  |
| Specificity |  |  |
| Sensitivity |  |  |

*Table 3. Best results*

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

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

A graph showing the growth of the number of countries/regions

Description automatically generated with medium confidenceLoss function on train and validation data is given in Figure 7.

*Figure 7. Loss function*

A graph of different types of graphs

Description automatically generated with medium confidencePerformance parameters, such as accuracy, specificity and sensitivity during epochs is shown in Figure 8.

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

## Integrated Gradients

## Grad-CAM

## Data Randomization Test

# References