The Application of AI in healthcare with focus on detecting diseases at early stages

Solution 1 – Sebastian Carp - 001289569

Solution 2 – Hanzla Ilyas - 001060407

Solution 3 – Marco Gizzi - 001327359

Solution 4 – Harmanpreet Singh - 001289568

Early disease detection is very important in healthcare because many conditions, like Pneumonia, and Cardiomegaly often start with small signs that are hard to spot. Chest X-rays are a common tool for diagnosing these diseases, but interpreting them can be challenging and time-consuming, especially in busy hospitals or areas with fewer medical professionals. This project uses artificial intelligence (AI) to analyze X-rays and identify conditions like Edema more quickly and accurately. By helping doctors detect diseases earlier, AI reduces their workload and makes better healthcare available to more people.

Solution 1 – SimpleCNN with implementations

This solution uses the base model of the CNN architecture, with an SE networks implementation that selectively emphasises the most important features in the given data.

```python
class SEBlock(nn.Module):
    def __init__(self, channels, reduction=16):
        super(SEBlock, self).__init__()
        self.fc1 = nn.Linear(channels, channels // reduction)
        self.fc2 = nn.Linear(channels // reduction, channels)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        # Global Average Pooling
        se_input = torch.mean(x, dim=[2, 3], keepdim=False)
        se_output = self.fc1(se_input)
        se_output = torch.relu(se_output)
        se_output = self.fc2(se_output)
        # Reshape to match the feature map size
        se_output = self.sigmoid(se_output).unsqueeze(2).unsqueeze(3)
        return x * se_output  # Scale the feature map
```

Implementation Process:

The images are loaded from the specified directory, converted to grayscale, and pre-processed with transformations such as resizing, random augmentations, and normalization. These processed images are then converted into tensors and fed into the model via a `DataLoader` for batch processing during training or evaluation. The model uses these images to learn features through convolutional layers, enhanced by a Squeeze-and-Excitation (SE) block, for classification.

Model Training:

Training was conducted on the NIH dataset, using **5000 images** for training and validation and **1000 images** for testing.

The model is trained using a K-fold cross-validation approach, where the dataset is split into 5 folds, and the model is trained and validated on each fold. For each fold, the model undergoes multiple epochs, during which it learns by optimizing the weights using the Adam optimizer and minimizing the CrossEntropy loss. The learning rate is adjusted with a scheduler, and data augmentation techniques are applied during training to increase robustness. The model's performance is evaluated after each epoch on a validation set, and the best model (based on training accuracy) is saved. This process is repeated for each fold, and the final performance is averaged across all folds to determine the model's overall effectiveness.

Solution 2 – ResNet-50

This solution uses an AI model based on ResNet-50 to analyse chest X-rays and predict diseases. It is designed for multi-label classification, meaning it can identify multiple diseases in a single image.

Implementation Process:

This model processes chest X-ray images by resizing them to 224x224 pixels to ensure all images are the same size and normalizing them to help the model work better with different types of data. It uses techniques like resizing and normalization to handle variations in the images, making it more reliable. Horizontal flipping is avoided to keep important medical details, like differences between the left and right lungs, accurate. As seen in this small snippet of code below.

```
from torchvision import transforms
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])

# transforms.RandomHorizontalFlip() not used because some disease might be more likely to the present in a specific lung (lelf/rigth)
transform = transforms.Compose([transforms.ToPILImage(),
                    transforms.Resize(224),
                    transforms.ToTensor(),
                    normalize])
```

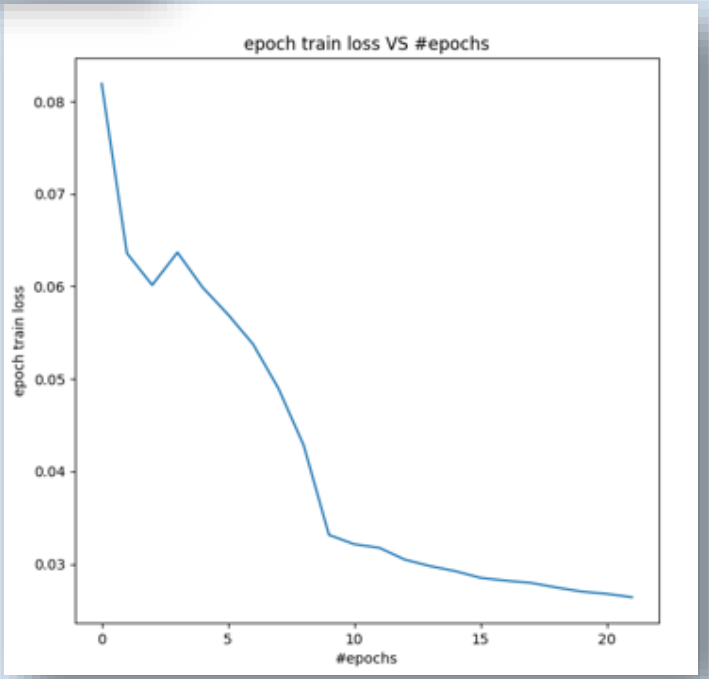
Model Training:

Training was conducted on the NIH dataset, using **86,524 images** for training and validation and **25,596 images** for testing. Focal Loss was applied to handle class imbalance effectively. The model was trained in stages:

Stage 1: Fine-tuned deeper layers with a lower learning rate for initial training stability.

Subsequent Stages: Gradually unlocked and fine-tuned additional layers for improved performance.

The graph from the training process shows that the model's loss decreases over time. This happens because of staged fine-tuning, where layers are gradually trained, optimized learning rates, and the use of Focal Loss, which helps the model handle class imbalances better.



Solution 2 – ResNet-50

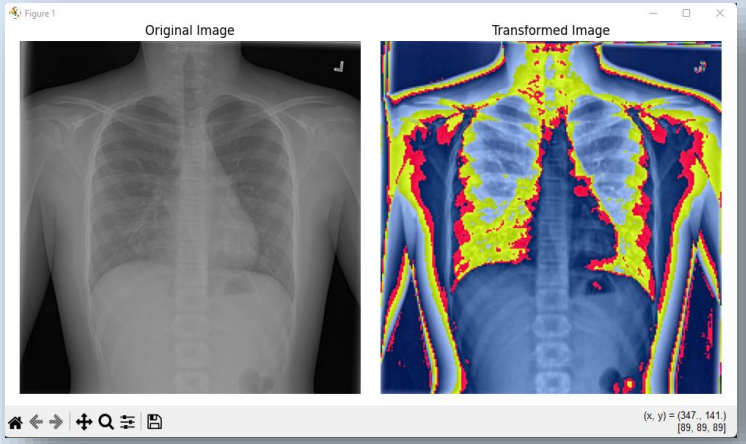
```
Class 0 (Atelectasis): ROC-AUC = 0.7214
Class 1 (Cardiomegaly): ROC-AUC = 0.8349
Class 2 (Consolidation): ROC-AUC = 0.6666
Class 3 (Edema): ROC-AUC = 0.8017
Class 4 (Effusion): ROC-AUC = 0.7824
Class 5 (Emphysema): ROC-AUC = 0.8722
Class 6 (Fibrosis): ROC-AUC = 0.7576
Class 7 (Hernia): ROC-AUC = 0.8371
Class 8 (Infiltration): ROC-AUC = 0.6296
Class 9 (Mass): ROC-AUC = 0.7468
Class 10 (No Finding): ROC-AUC = 0.7011
Class 11 (Nodule): ROC-AUC = 0.6754
Class 12 (Pleural_Thickening): ROC-AUC = 0.7159
Class 13 (Pneumonia): ROC-AUC = 0.6502
Class 14 (Pneumothorax): ROC-AUC = 0.8099

Overall Metrics:
Macro-Average ROC-AUC: 0.7469
Script completed in 0h 7m 1s!
```

Testing and Evaluation:

The model's performance is measured using **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)**, which shows how well it distinguishes between diseased and non-diseased cases. An overall ROC-AUC score of **74.7%** indicates strong accuracy, with high performance for diseases like Emphysema (87.2%), Cardiomegaly (83.5%), and Hernia (83.7%), though some conditions like Infiltration (62.9%) and Pneumonia (65.0%) had lower scores. This can be seen from the testing results on the right.

Testing on New Images:

I tested the trained model on a few random X-ray images by applying the same preprocessing steps used during training, such as resizing and normalization. The model provided predictions for each class, and when I compared them to the actual labels, the results were quite accurate with only a small percentage of errors. This gave me confidence in the model's reliability and performance.




```
Predictions:
Atelectasis: Probability: 0.5794 (Positive)
Consolidation: Probability: 0.0275 (Negative)
Infiltration: Probability: 0.6299 (Positive)
Pneumothorax: Probability: 0.0303 (Negative)
Edema: Probability: 0.0816 (Negative)
Emphysema: Probability: 0.0196 (Negative)
Fibrosis: Probability: 0.9341 (Positive)
Effusion: Probability: 0.0850 (Negative)
Pneumonia: Probability: 0.3242 (Negative)
Pleural_thickening: Probability: 0.2710 (Negative)
Cardiomegaly: Probability: 0.2855 (Negative)
Nodule: Probability: 0.1925 (Negative)
Mass: Probability: 0.2417 (Negative)
Hernia: Probability: 0.1604 (Negative)
No findings: Probability: 0.0495 (Negative)

Visualizing transformations...
```

```
IMAGE_PATH = r"D:\Multi_label_NIH\sample_xrays\00025075_007.png"  # Path to your sample image
```

95377	00025075_007.png	Atelectasis|Fibrosis	7	25075	12	F	PA	2021	2020	0.194311	0.194311

Overall: This model uses advanced techniques like image augmentation, balanced loss functions, and staged training to accurately predict diseases from chest X-rays. It effectively handles class imbalance, ensures reliable predictions for rare conditions, and provides robust performance metrics, making it highly suitable for real-world medical diagnostics.

Solution 3 - DenseNet121

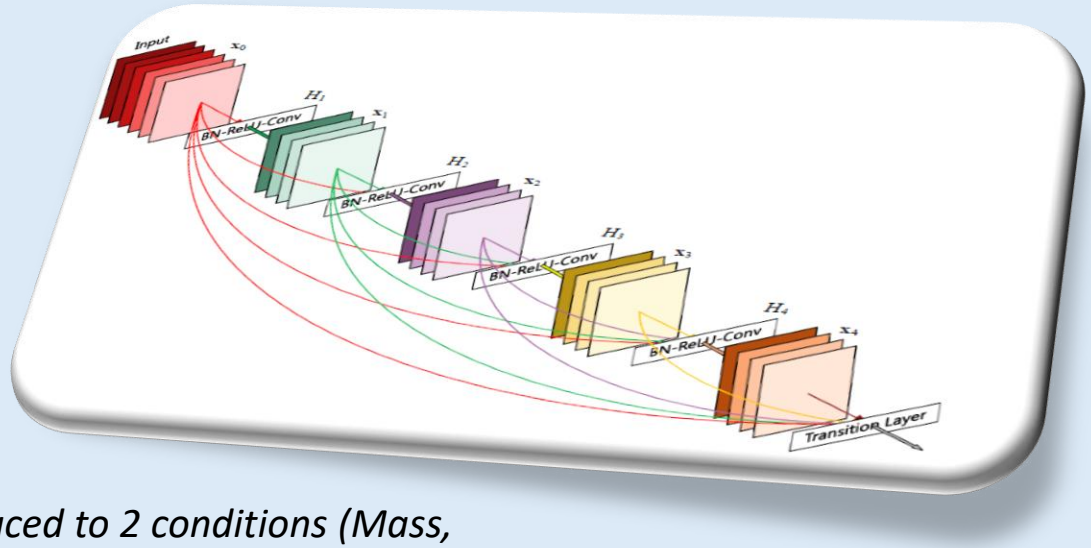
DenseNet121:

Connects each layer to every other layer, to ensure dense connectivity as in Figure 1.

Reduced parameters compared to ResNet.

Combines features via concatenation instead of summation.

Facilitates gradient flow and feature reuse.



Figuere1 (Huang et al., 2017)

Dataset and Preprocessing:

Dataset composed of Chest X-ray images with 14 conditions reduced to 2 conditions (Mass, Pneumothorax).

Images resized 224x224 avoiding different sizes, CenterCrop has been applied for robust classification.

Balanced dataset: 3:1 ratio of healthy to unhealthy images to imitate a real scenario.

Model Implementation:

Pre-trained CheXNet model modified, trained new model for 2 classes (Mass and Pneumothorax).

Image cropping using CenterCrop instead then TenCrop as the dataset are front chest X-rays.

Data script have been modified for binary classification.

Prediction, precision, recall, F1-score, and confusion matrix have been added for a better performance evaluation.

Solution 3 - DenseNet121

Challenges:

The model was working with a previous version of python, some of the features, function and libraries were outdated.

The model was detecting 14 diseases, modification to identify 2 conditions led to compatibility issues.

Balancing the validation, training and test list. With no appropriate ratio and labels the model result inaccurate.

Computational challenges when training the model, working with the CPU had major effect on time complexity.

Results:

From the AUROC score (Area Under Receiver Operating Curve) the model has a good ability to distinguish between the positive and negative classes, 82% for Mass and 81% for Pneumothorax .

Single images was taken online and tested for prediction; both cases have been tested, and the model is able to distinguish the conditions.

Precision measures the corrected prediction, the result shows 0.59% positive Mass, and 0.76% Pneumothorax in the given data. The model has higher results with Pneumothorax.

Recall measures the correct positive prediction.

F1-Score is an harmonic mean between precision and recall.

Support is the number of given data.

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

        Mass       0.59      0.86      0.70       413
Pneumothorax       0.76      0.70      0.73       438

   micro avg       0.65      0.78      0.71       851
   macro avg       0.67      0.78      0.71       851
weighted avg       0.67      0.78      0.71       851
 samples avg       0.65      0.66      0.65       851

Confusion Matrix:
 [[478  84]
 [143 295]]
Macro-Averaged AUROC: 0.8240
AUROC for Mass: 0.8057
AUROC for Pneumothorax: 0.8424
```

```
Predicted probabilities for single image:
Mass: 72.87%
Pneumothorax: 50.64%
```

```
Predicted probabilities for single image:
Mass: 53.82%
Pneumothorax: 67.24%
```

Using Autoencoder in My Solution 4

For my solution, I am utilizing an autoencoder, a specialized type of artificial neural network designed for unsupervised learning. Autoencoders are particularly effective for learning efficient coding's of unlabelled data.

```python
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
```

It consists of two parts :

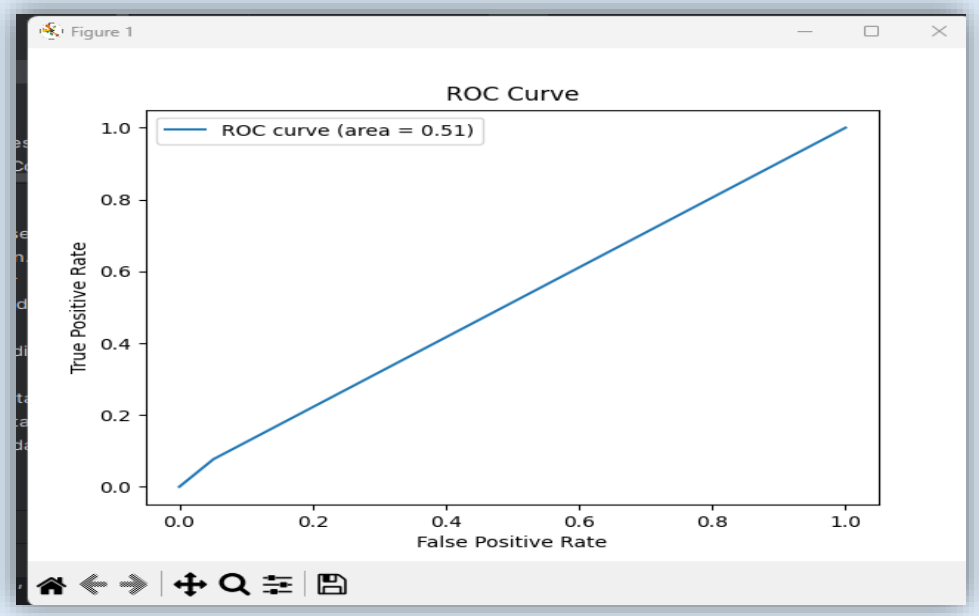
- **Encoder**: Compresses the input into a smaller latent representation.
- **Decoder**: Reconstructs the original input from the latent representation.

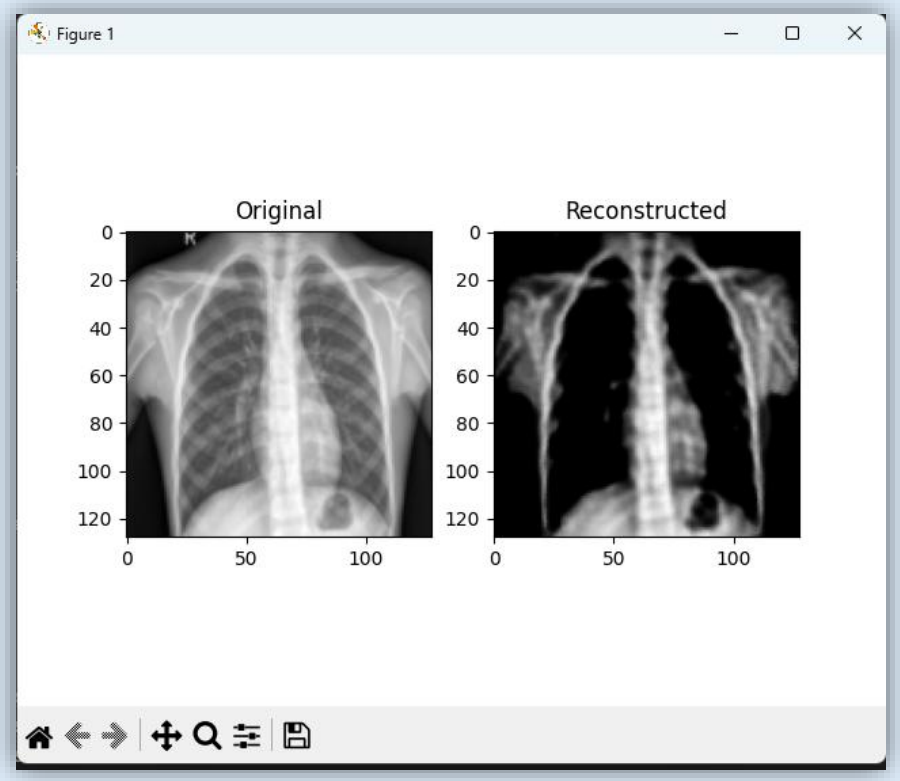
```python
self.decoder = nn.Sequential(
    nn.ConvTranspose2d( in_channels: 128,  out_channels: 64, kernel_size=3, stride=2, padding=1, output_padding=1),  # 64 x 32 x 32
    nn.ReLU(),
    nn.ConvTranspose2d( in_channels: 64,  out_channels: 32, kernel_size=3, stride=2, padding=1, output_padding=1),  # 32 x 64 x 64
    nn.ReLU(),
    nn.ConvTranspose2d( in_channels: 32,  out_channels: 1, kernel_size=3, stride=2, padding=1, output_padding=1),  # 1 x 128 x 128
    nn.Sigmoid()
)
```

The encoder uses **convolutional layers** to compress the input image.

Solution 4

The model's True Positive Rate (Sensitivity) and False Positive Rate (1-Specificity) are traded off in this Receiver Operating Characteristic (ROC) curve.
The model's performance is indicated by the Area Under the Curve (AUC) value. The AUC in this case is 0.51, which is really near to the 0.5 for random guessing.





The reconstruction closely matches the original for normal X-rays when the autoencoder learns patterns typical of normal chest X-rays.Any discrepancy between the original and recreated image indicates anomalies, such as pneumonia symptoms.

Conclusion

These methods allow us to pick the right approach for different needs, from simple setups to advanced diagnostics. Combining them could further improve reliability and accuracy in disease detection.

Looking ahead, there’s potential to integrate these algorithms into real-world clinical systems, like combining them with electronic health records for personalized diagnostics. Future work could focus on creating hybrid models or ensembles that bring together the best of these techniques. There’s also room to explore how these methods perform with larger and more diverse datasets, improving their generalization and making them even more effective in healthcare.

These advancements can make early disease detection faster, more accurate, and accessible, ultimately improving patient care and saving lives.

References

National Institutes of Health Clinical Center (2017). NIH Chest X-ray Dataset [Data set]. Kaggle. Available at: https://www.kaggle.com/datasets/nih-chest-xrays/data

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. Available at: https://arxiv.org/abs/1512.03385

Weng, A. (2017) *CheXNet: Model Implementation*. GitHub. Available at: https://github.com/arnoweng/CheXNet/blob/master/model.py

Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K.Q., 2017. Densely connected convolutional networks. *arXiv preprint*, arXiv:1608.06993. Available at: https://arxiv.org/abs/1608.06993

Hu, J., Shen, L. and Sun, G. (2018). Squeeze-and-Excitation Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp.7132–7141. doi:https://doi.org/10.1109/cvpr.2018.00745.