Report
Pneumonia detection using X-ray images

Group: BDA-2102
Students: Zhauymbayeva Madina, Serikkazy Ulpan
Link to GitHub: https://github.com/zhvmdn/Pneumonia_Detection
Link to Video: https://www.youtube.com/watch?v=JOVYkFSyomk

## Introduction:

- Problem:

Pneumonia is one of the very contagious diseases, the so-called "Lung Infection" that has affected the lives of millions of people around the world. You're more likely to get pneumonia if you have asthma, chronic obstructive pulmonary disease (COPD) or heart disease. Most often, this disease causes respiratory symptoms that can be very similar to shortness of breath, acute respiratory viral infections or tachycardia. Pneumonia can range in seriousness from mild to life-threatening. Pneumonia can be quickly and accurately diagnosed using computed tomography (CT) and chest X-ray (CXR). However, since it takes a long time and is very prone to human error, identifying an infection manually using a radio image is quite difficult. Our goal is to create an image recognition model that will allow us to determine the presence of Pneumonia from an X-ray image of the patient's lungs.

- Literature review with links (another solutions):

Author of this model used a custom deep convolutional neural network and retraining a pre-trained "InceptionV3" model to identify pneumonia from x-ray images. For retraining, he removed the output layers, freezed the first few layers, and fine-tuned the model for two Pneumonia and Normal classes.
https://github.com/anjanatiha/Pneumonia-Detection-from-Chest-X-Ray-Images-with-Deep-Learning

Author of this project divided the train set into training and validation data. He solved the problem of data imbalance using stratified class. Training and validation set split changed from 99:1 ratio to 90:10 ratio.
https://www.kaggle.com/code/wirachleelakiatiwong/pneumania-classification-transfered-cnn/notebook

In this project, the author compared pre-trained models such as VGG16, CNN_2, ResNet, InceptionNet, DenseNet with their mean absolute error. Also, the author built his own CNN model that showed 91% of accuracy.
https://github.com/0xpranjal/Pneumonia-Detection-using-Deep-Learning

- Current work (description of the work)

We used Google Colab as IDE and installed kaggle to upload the dataset into Colab files. The Pneumonia detection model gets a dataset that is divided into three folders (train, validation and test) from the content of colab. Validation set images are moved to the train set and images from the train set are used for training and validation. Train set has a data imbalance problem, and assigning weights to classes solves it. Model is built using CNN architecture, and it trains to identify normal and pneumonia patients in chest x-ray images. In training, the model finds patterns in the passed train data and makes predictions. Training and validation accuracy and loss are visualized after completion of training to see how the learning performance changes over epochs. For evaluation of model, classification report and roc curve are used. In the end, we built the model's demo with Gradio.

## Data and Methods

- Information about the data

The model is built from a dataset from Kaggle and this dataset is divided into train, test and validation data. Overall, there are 5856 X ray images (JPEG) and each of the 3 folders are breached into 2 subfolders (Normal / Pneumonia), containing first chest x-ray images of a healthy person and second is for the x-rays of a person with pneumonia. Train set has 3875 x-ray images in class Pneumonia and 1341 x-ray images in class Normal. Test set has 390 x-ray images in class Pneumonia and 234 x-ray images in class Normal. Validation set has 8 x-ray images in class Pneumonia and 8 x-ray images in class Normal. As the validation data set has only 16 images, validation images were moved to the train set. Then the train set was splitted into training and validation (90% - training, 10% - validation) in the ImageDataGenerator.

Figure 1 depicts what x-rays of a person with pneumonia and a healthy person look like. It can be noticed that normal x-rays have less white shading in the lung area and are more clear, while Pneumonia x-rays are more opaque.

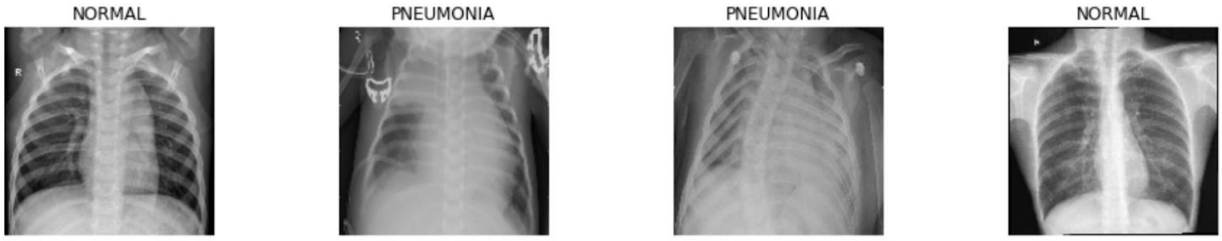


Figure 1. X-ray images

It was found that the images of pneumonia are much more than normal during familiarization with the data set. Bar graph (Figure 2) below proves that the number of normal x-ray images in the train set is less, or the data in that set is unbalanced. Also, Class 0 (Normal) weight is greater than class 1 weight (Pneumonia), meaning that the number of the normal x-ray images is less. To make the training data balanced, each normal image was weighted more to balance the data. Also, data augmentation was used for better accuracy and to avoid overfitting.

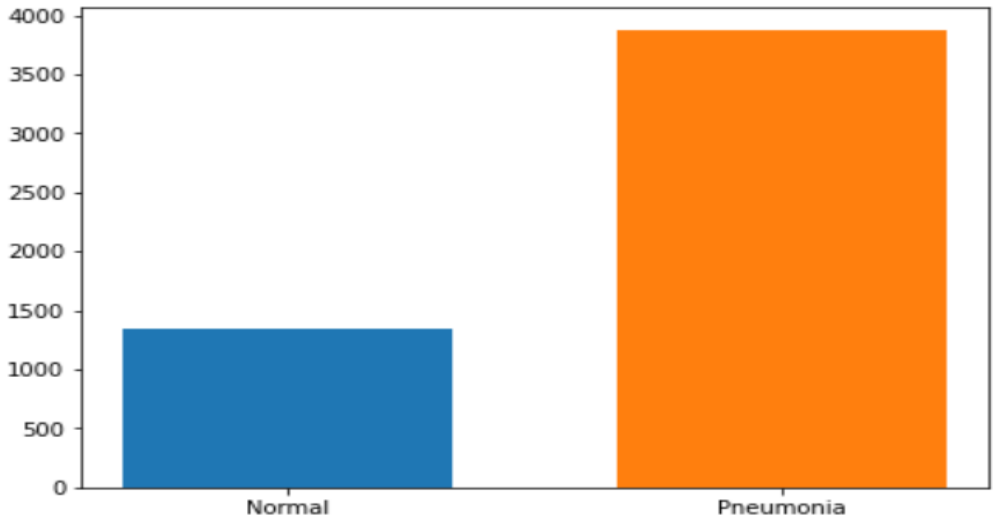


Figure 2. Unbalanced train set

keras.preprocessing.image class Image Data Generator generates batches of tensor image data with real-time data augmentation. We do not have enough diverse sets of images, so for data cleaning and preprocessing data augmentation creates more training samples. Parameters such as rescale (img pixels between 0 and 1), rotation_range (range for rotation), shear_range (shear intensity), zoom_range (the image is enlarged) and horizontal_flip (randomly flips the images horizontally) were used for augmentation on the train set. All images were rescaled by 1/255 and were resized to 150x150. Validation split is set to 0.1, so training data (90%) and validation data (10%) will be created from the data of the train set. flow_from_directory reads the images from folders. Target size was set to 150x150 and batch size is 32. Image batch shape: (32, 150, 150, 3).

- Description of the ML models you used with some theory

For our project, we have created Convolutional Neural Networks architecture. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. CNNs are especially useful for finding patterns in images for recognizing objects, classes, and categories (Figure 3). The model is built with 4 convolutional blocks consisting of convolutional layers, max-pooling. The depth of the feature maps increased from 32 to 64 and size of the feature maps decreased from 150x150 to 10x10. In the end, we used the Dense layer of size 3 and sigmoid function as this is a binary classification problem. For compilation, Adam is the optimizer and binary cross-entropy is the loss. In between, Dropout is used to avoid overfitting. As this is the imbalanced data set, for the metric, Precision and Recall are included and they show a more accurate view of how effective the model is. Additionally, precision measures the accuracy of positive predictions, while recall measures the completeness of positive predictions. Labels were not defined as 0 and 1 since ImageDataGenerator already made that (rescale = 1/255). Early Stopping callback is used to avoid overfitting by stopping the epochs depending on a metric and conditions. restore_best_weights is set to true, so the returned model is the model with the best weights. Reduce learning rate when a metric has stopped improving.

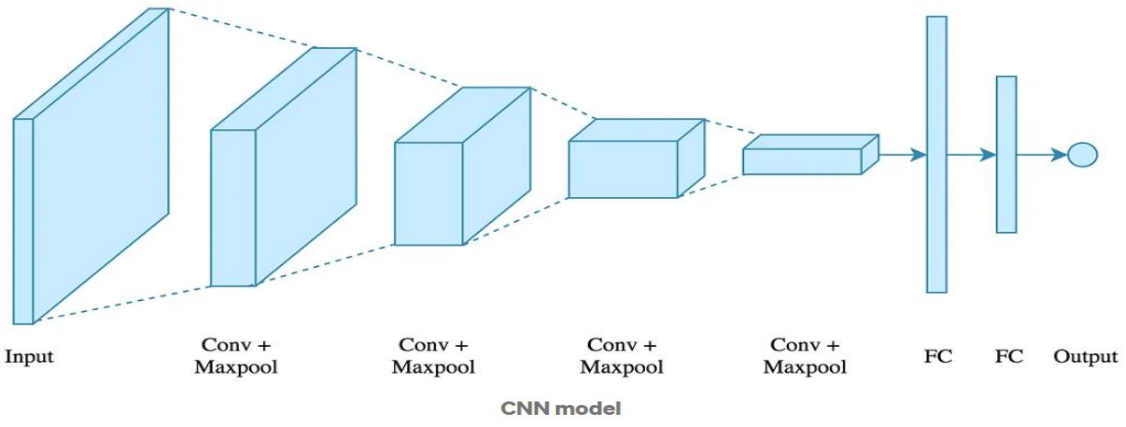


Figure 3. CNN model structure
https://medium.com/@mayankverma05032001/binary-classification-using-convolution-neural-network-cnn-model-6e35cdf5bdbb

## Results

After training, we obtained the plots of accuracy and loss of training and validation to identify any learning issues that can cause a model to be either underfit or overfit. As can be seen in Figure 4, training accuracy or the number of correct predictions increases, and its loss or summation of errors decreases. When it comes to validation, its accuracy rises and loss decreases, but it fluctuates a little. In 10 epochs, the model achieved 94% of training and 91% of validation accuracy.

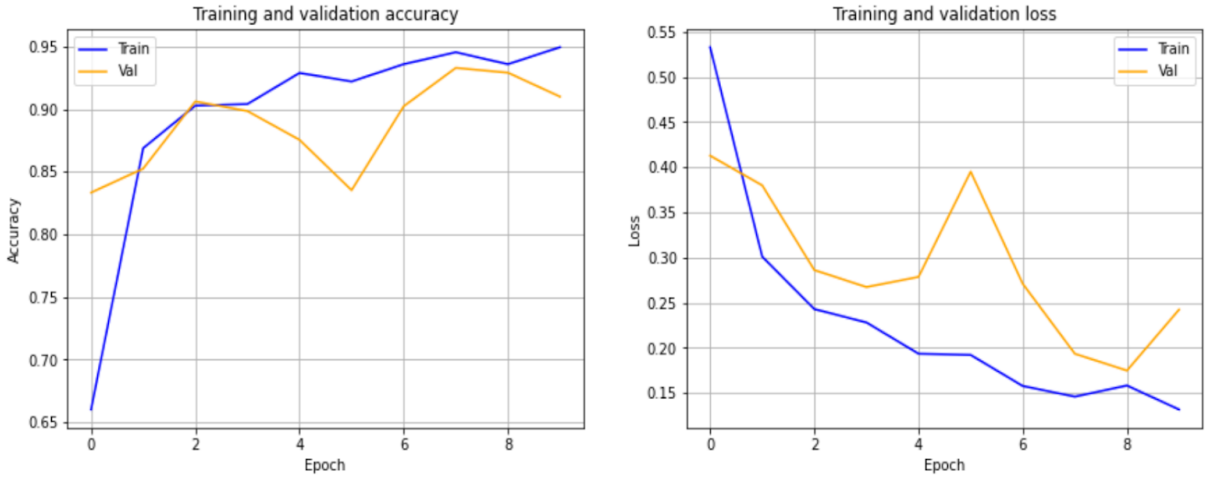


Figure 4. Accuracy and Loss

We made an evaluation on the test set. Results showed that we got test accuracy of 92.15%, test loss of 0.22%, test precision of 93.38%, and test recall of 94.10%.

```
Test Loss      : 0.22
Test Accuracy  : 92.15%
Test Precision : 93.38%
Test Recall    : 94.10%
```

Figure 5. Evaluation on the test set

Classification report details on the normal and pneumonia class to measure the quality of predictions are shown in Figure 6. To predict the metrics of the classification report. True Positives, False Positives, True negatives and False Negatives are used. We obtained a recall of 94% (positive cases), Precision of 93% (correct predictions) and F1 score of 94% (correct positive predictions) for Pneumonia class, which are pretty high.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Normal | 0.90 | 0.89 | 0.89 | 234 |
| Pneumonia | 0.93 | 0.94 | 0.94 | 390 |
| accuracy | | | 0.92 | 624 |
| macro avg | 0.92 | 0.91 | 0.92 | 624 |
| weighted avg | 0.92 | 0.92 | 0.92 | 624 |

Figure 6. Classification Report

False Negatives (how many times model predicted pneumonia images as normal) for Model are just 23 and False Positives (how many times model predicted normal images as pneumonia) are 26. Our model is able to classify 575 x ray images correctly out of 624 test images.

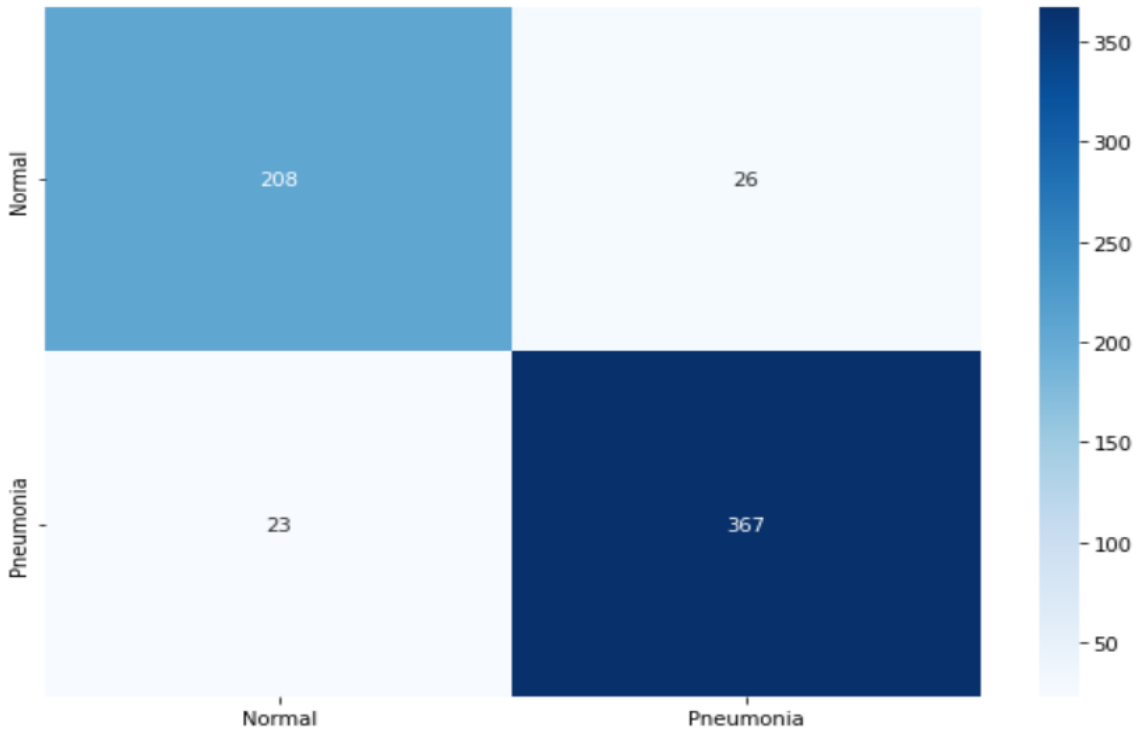


Figure 7. Confusion Matrix

ROC AUC score tells that the model is 97% efficient (Figure 8). The model's performance at distinguishing between the positive and negative classes is almost perfect.

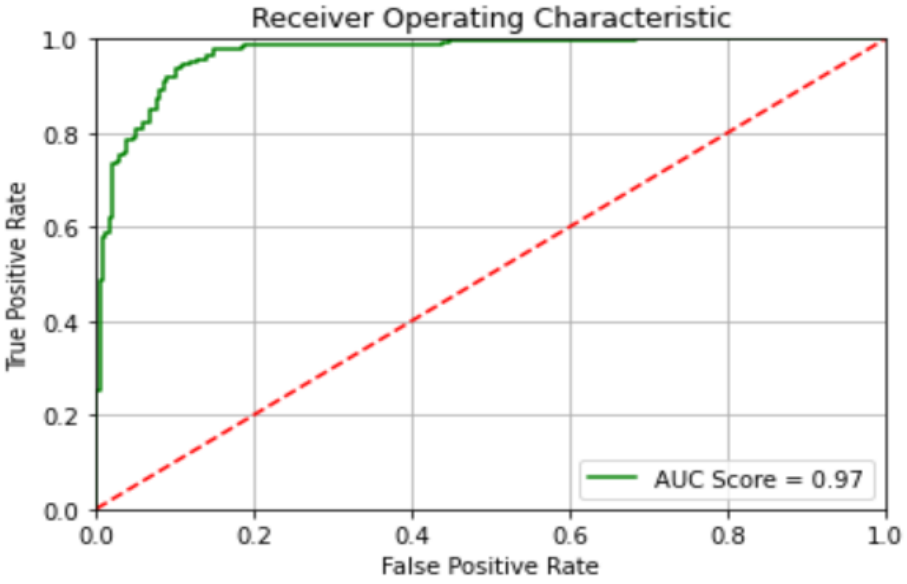


Figure 8. ROC Curve

Figure 9 shows the model predictions on the test set and their actual value. The first x-ray image was predicted correctly, while the second was not.

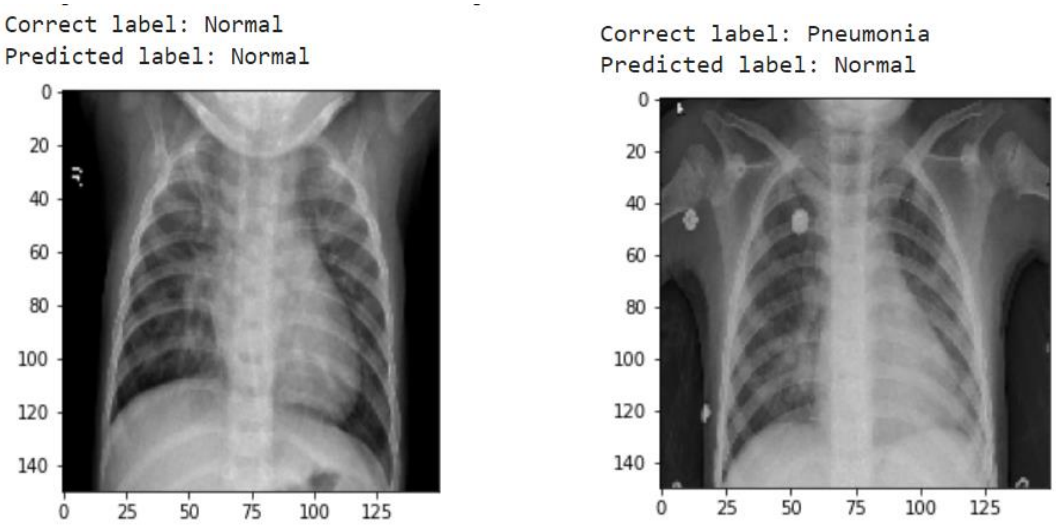


Figure 9. Predictions on the test set

To test the model with images we used Gradio, which is a free and open-source Python library that allowed us to develop an easy-to-use customizable component demo for a machine learning model. Figure 10 and Figure 11 depicts how the model classified correctly normal and pneumonia x-ray images which were taken from the test set's normal and pneumonia folders.

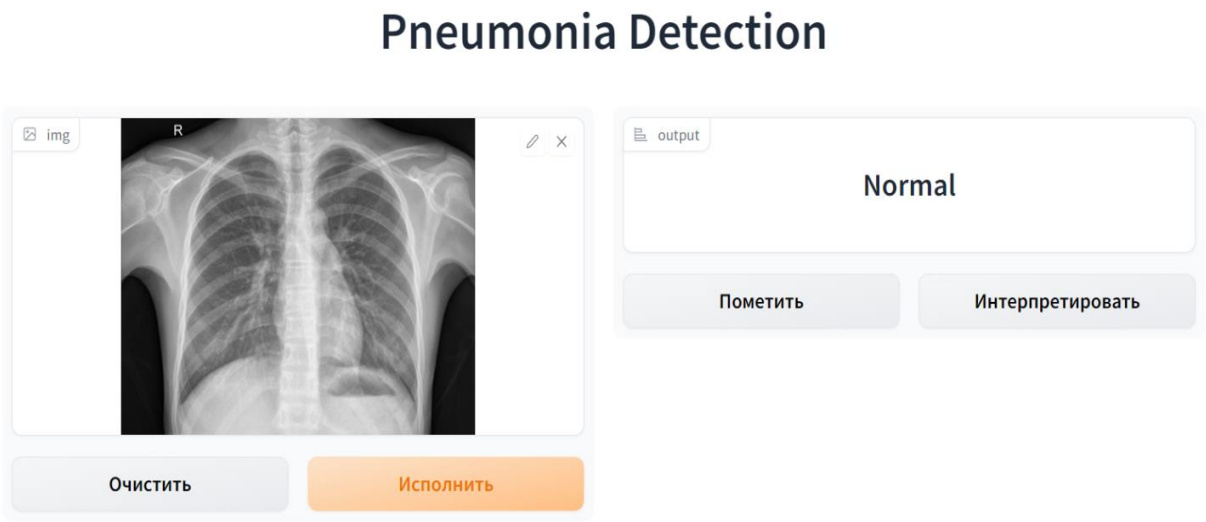


Figure 10. Prediction made on normal image with Gradio

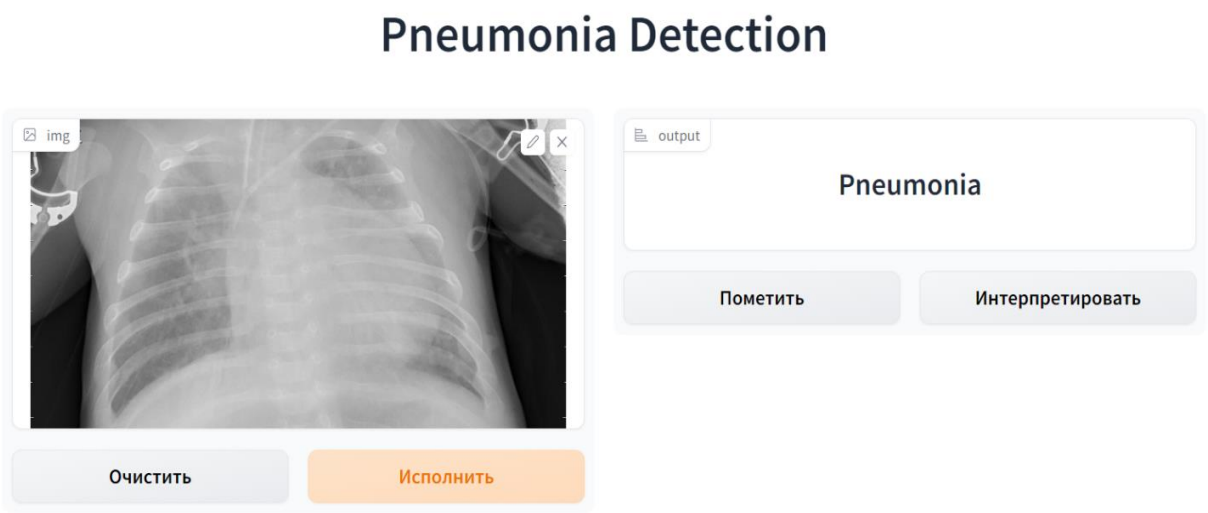


Figure 11. Prediction made on pneumonia image with Gradio

## Discussion

- Critical review of results

We think that our model makes good predictions as test accuracy was 92%. As mentioned, our project is based on the method of training using CNN to identify normal patients and patients with pneumonia on chest X-rays. The project was executed well, but during the simulation it was not without some difficulties.

1) According to the results, the model works very well, it is worth mentioning that the model could not correctly classify 49 images out of 624. In the future, we can improve the results with higher accuracy by resolving this error.

2) In the ImageDataGenerator, we encountered problems with the choice of parameters. It took time and effort to find the right parameters and eventually chose rescale, rotation_range, shear_range, zoom_range and horizontal_flip.

3) We faced an uneven distribution of observations since our dataset is not balanced. Having solved this problem with assigning weights to classes, we came to the conclusion that we need more x-ray images for a more accurate result and preferably with equal data.

- Next steps

In the further development of the project, we would like to use transfer learning from a pre-trained model (Figure 12). Transfer learning makes use of the knowledge gained while solving one problem and applying it to a different but related problem. Therefore, we can achieve a high level of performance, even with a small amount of data. During the training of this model, it will distinguish our data sets with accurate characteristics, and, naturally, less errors will be detected proportionally to this. In addition, we believe that using other datasets related to our project will be advantageous in training. Moreover, we want to demonstrate our model to medical specialists in order to check the work of it and get advice to understand the differences between normal and pneumonia x-ray images from a medical point of view. Their recommendations will help to build more accurate model.

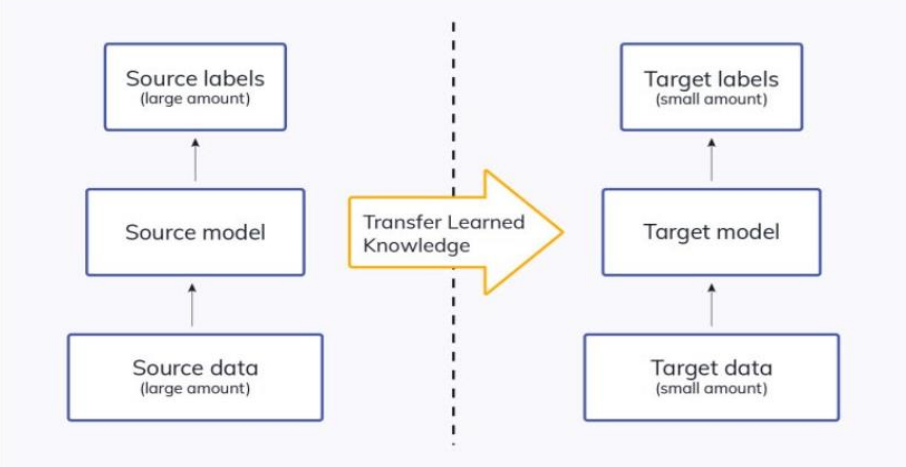


Figure 12. Transfer learning scheme

## References:


1. *Chest X-Ray Images (Pneumonia)*. (2018, March 24). Kaggle. https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia
2. Shriram Vasudevan. (2021, January 23). *How to Import (use) Kaggle datasets in Google Colab?* [Video]. YouTube. https://www.youtube.com/watch?v=57N1g8k2Hwc
3. GeeksforGeeks. (2022, September 1). *How to move all files from one directory to another using Python*. https://www.geeksforgeeks.org/how-to-move-all-files-from-one-directory-to-another-using-python/
4. Saha, S. (2022, November 16). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Medium. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
5. Kundu, R., Das, R., Geem, Z. W., Han, G., & Sarkar, R. (2021). Pneumonia detection in chest X-ray images using an ensemble of deep learning models. *PLOS ONE, 16*(9), e0256630. https://doi.org/10.1371/journal.pone.0256630
6. Singh, K. (2022, December 2). *How to Improve Class Imbalance using Class Weights in Machine Learning*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/
7. A. (2020, July 20). *TensorFlow Pneumonia Classification on X-rays*. Kaggle. https://www.kaggle.com/code/amyjang/tensorflow-pneumonia-classification-on-x-rays
8. F. (2021, June 6). *deep-learning-with-python-notebooks/5.2-using-convnets-with-small-datasets.ipynb at master · fchollet/deep-learning-with-python-notebooks*. GitHub. https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/first_edition/5.2-using-convnets-with-small-datasets.ipynb

9. Çakır, R. (2021, December 12). *Create Tensorflow Image Classification Model with Your Own Dataset in Google Colab*. Medium. https://medium.com/analytics-vidhya/create-tensorflow-image-classification-model-with-your-own-dataset-in-google-colab-63e9d7853a3e
10. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
11. Amit, D. (2021, February 19). *How can Tensorflow be used to create a pair using a file path for the flower dataset?* https://www.tutorialspoint.com/how-can-tensorflow-be-used-to-create-a-pair-using-a-file-path-for-the-flower-dataset
12. Vanshika, S. (2022, March 6). *Pneumonia Detection using Convolutional Neural Network.* [Video]. YouTube. https://youtu.be/5AeKIAAOzC0
13. Damenian, P. (2017, September 19). *Image with directories as labels for Tensorflow data.* https://pontifex.dev/posts/images-with-directories-as-labels-for-tensorflow-data/
14. Deshmukh, H. (2021c, December 16). *Medical X-ray ⚡Image Classification using Convolutional Neural Network*. Medium. https://towardsdatascience.com/medical-x-ray-%EF%B8%8F-image-classification-using-convolutional-neural-network-9a6d33b1c2a
15. Ogunbiyi, I. A. (2022, June 2). *How to Deploy a Machine Learning Model as a Web App Using Gradio*. freeCodeCamp.org. https://www.freecodecamp.org/news/how-to-deploy-your-machine-learning-model-as-a-web-app-using-gradio