**DETECTING PNEUMONIA BY X-RAY IMAGES (WEB, APP) AND MAKE CLINICAL DIAGNOSIS**

Phan An Dong 19146113 #1

# Student of University of Technology and Education of Ho Chi Minh city

[119146113@student.hcmute.edu.vn](mailto:119146113@student.hcmute.edu.vn)

1. ***Abstract***

Pneumonia is a respiratory infection caused by bacteria or viruses. It can affect many individuals, especially those in developing and underdeveloped nations, where high levels of pollution, unhygienic living conditions, and overcrowding are common. Early diagnosis of pneumonia is critical to ensure successful treatment and increased survival rates.The most commonly used method for diagnosing pneumonia is chest X-ray imaging. However, the examination of chest X-rays is a difficult task and can be subjective. In this study, we developed a computer-aided diagnosis system that can automatically detect pneumonia from chest X-ray images. The values ​​of the four standard assessment metrics precision, recall, f1-score and the area under the curve are merged into the weight vector, which has often been determined experimentally and error-prone in studies in the literature.

1. ***Introduction***

Pneumonia is an acute lung infection that can be caused by bacteria, viruses, or fungi. It can cause inflammation of the air sacs and pleural effusion, a condition in which the lung is filled with fluid.It accounts for more than 15% of deaths in children under the age of five, and pneumonia is more common in developing and underdeveloped countries, where overpopulation, pollution and unsanitary environmental conditions exacerbate the situation, and medical resources are scarce. Therefore, early diagnosis and treatment can play an important role in preventing the disease from becoming fatal. CT, MRI, and X-rays are often used to diagnose lung problems. X-ray imaging is an examination that is non-invasive and relatively affordable. Fig 1 shows an example of a pneumonic lung image. The white spots in the pneumonic image (indicated with red arrows), called infiltrates, distinguish a pneumonic from a healthy condition.X-rays of the chest are often subject to variability, making them less reliable for detecting pneumonia. A computer-aided diagnosis system that uses an ensemble of deep learning models for the accurate classification of chest X-ray images is needed. In this study, we developed a system that uses an approach called deep transfer learning.

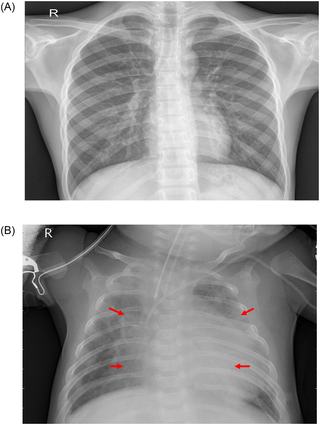


Figure 1. Examples of two X-ray plates that display (a) a healthy lung and (b) a pneumonic lung.

The red arrows in (b) indicate white infiltrates, a distinguishing feature of pneumonia.

Deep learning is a powerful artificial intelligence tool that is essential for solving many complex vision problems. CNNs are a popular type of deep learning model for image classification problems. However, such a model works best only when a large amount of data is provided.In the biomedical image classification problem, it is difficult to obtain such a large amount of labeled data because a specialist doctor needs to classify each image. Time-consuming work. Transfer learning is a way to overcome the obstacle of a small dataset. A model trained on a large dataset can be reused and the network weights determined in this model can be applied to the small dataset.CNNs are often used to classify images of biomedical objects. This is due to the large dataset of ImageNet, which includes more than 14 million images.

We are going to use CNN to analyze X-ray images of the chest. The dataset is not representative of the population. Images can have one or three color channels, which can be different on different sites.

### ***Motivation and contributions***

Pneumonia is a common disease that affects a large number of people, especially children, in developing and underdeveloped countries where risk factors such as overcrowding, poor hygiene conditions, and malnutrition are common. The early diagnosis of pneumonia is important in treating the disease completely.X-rays are the most common way to diagnosis pneumonia, but it depends on the interpretation abilities of the radiologists and often is not agreed upon. Thus, an automatic CAD system with the ability to generalize is required for disease diagnosis. The study used a new approach, ensemble learning, to improve the classification of pneumonia cases. This technique incorporates the different predictive abilities of different models, which is better than using a single model. To deal with the shortage of biomedical data, a transfer learning model was used as the base learner, and its decision scores were combined into an ensemble.

The main contributions of this study are as follows.

1. Is there a way to find out if a file is corrupt without having to delete and re-upload it? A framework was developed that could help boost the performance of CNNs used for classifying pneumonia. For this purpose, a weighted average technique was chosen.
2. This is an incomplete paraphrase. Please add more information. The weights assigned to the classifiers were determined by combining four evaluation metrics: precision, recall, f1-score, and AUC. Instead of relying on classifiers that are accurate or based on experiment results, we used a hyperbolic tangent function.
3. ***Proposed method***

We will use CNN, Tensorflow 2.0, Keras. We will pre-process our data in a very simple and intuitive way (loading, resizing, converting to grayscale, creating labels). The model is using grayscale images because they don't make much sense to use X-Ray images in colorful form.

Our data consists of 3 volumes - Training, Test, Vol. Modeling and validation will be used to check the model's performance. The validation set is small, containing only 16 cases. We will create data frames so we can visualize the distribution of cases in each set. The large dataset may not work well with this type of load because you will run out of memory.A data generator with flow\_from\_directory may be more efficient, but it is very slow.

*What is CNN?*

A Convolution Neural Network is a multi-layered artificial neural network that is capable of detecting complex features in data, for instance extracting features in image data.

*How do CNNs work?*

Although they can be used for other tasks, CNNs are mostly used in tasks involving image data. Each image contains pixel data that can be represented in a numerical form. This numerical representation is what is passed to a CNN. As much as normal artificial neural networks can be used in processing image data, CNNs have proven to perform better, resulting in higher accuracy. Let’s now take a look at how CNNs work.

*Convolution*

Usually, you will not feed the entire image to a CNN. You will feed the features that are most important in classifying the image. The features are obtained through a process known as convolution. The convolution operation results in what is known as a feature map. It is also referred to as the convolved feature or an activation map. The feature map is obtained by applying a feature detector to the input image. The feature detector is also referred to as a kernel or a filter. The filter is usually a 3 by 3 matrix. However, other types of matrices can be used. The feature map is obtained through an element-wise multiplication of the filter with the matrix representation of the input image. The objective here is to reduce the size of the image being passed to the CNN while maintaining the important features. The filter slides step by step through each of the elements in the input image. These steps are known as strides and can be defined when creating the CNN. When building the CNN you will be able to define the number of filters you want for your network.

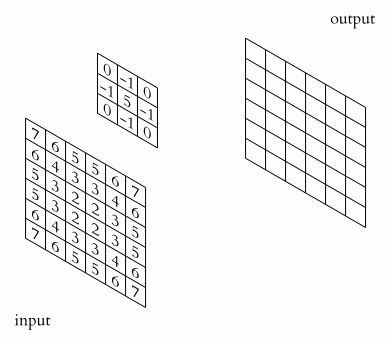


Figure 2. Convolution

Once you obtain the feature map, the Rectified Linear unit is applied in order to prevent the operation from being linear. This is because working with images is not linear.

*Pooling*

Pooling results in what is known as a pooled feature map. Pooling ensures that the neural network is able to detect features in an image irrespective of their location in an image. This is what is known as spatial invariance. There are several types of pooling, for example, max-pooling average pooling, and min pooling. For instance, in max-pooling a 2 by 2 matrix is slid over the feature map while picking the largest value in a given box.



Figure 3. Pooling example

Pooling ensures that the main features of the image are maintained while reducing the size of the image further. This reduces the amount of information passed to the neural network and hence helps to reduce overfitting.

*Flattening*

The next step is to flatten the pooled feature map. This involves transforming the entire pooled feature map into a single column that can be passed to the fully connected layer.

*Full connection*

The flattened feature map is then passed to the input layer of the neural network. The result of that is passed to a fully connected layer. After that, the result of the entire process is emitted by the output layer. An activation function is usually applied depending on the type of classification problem. For binary classifications, the sigmoid activation function will be used whereas the softmax activation function is used for multiclass problems.

*Convolutional Neural Networks (CNN) in TensorFlow*

Now that you understand how convolutional neural networks work, you can start building them using TensorFlow. However, you will first have to install TensorFlow. If you are working on a Google Colab environment, TensorFlow will already be installed.

*What are Keras and tf.keras?*

As of TensorFlow 2.0, Keras has become the official high-level API for TensorFlow. It is an open-source package that has been integrated into TensorFlow in order to quicken the process of building deep learning models. It is accessible via `tf.keras`. That is what you will be using in this article.

*Develop multilayer CNN models*

Loading the dataset can be done directly by using Keras utilities. Other datasets that ship with TensorFlow can be loaded in a similar manner.

*Visualize distribution of cases*

Now let's draw how are our datasets (im)balanced. You will find out, that training dataset is highly imbalanced, testing dataset is slightly imbalanced and validation dataset is balanced:

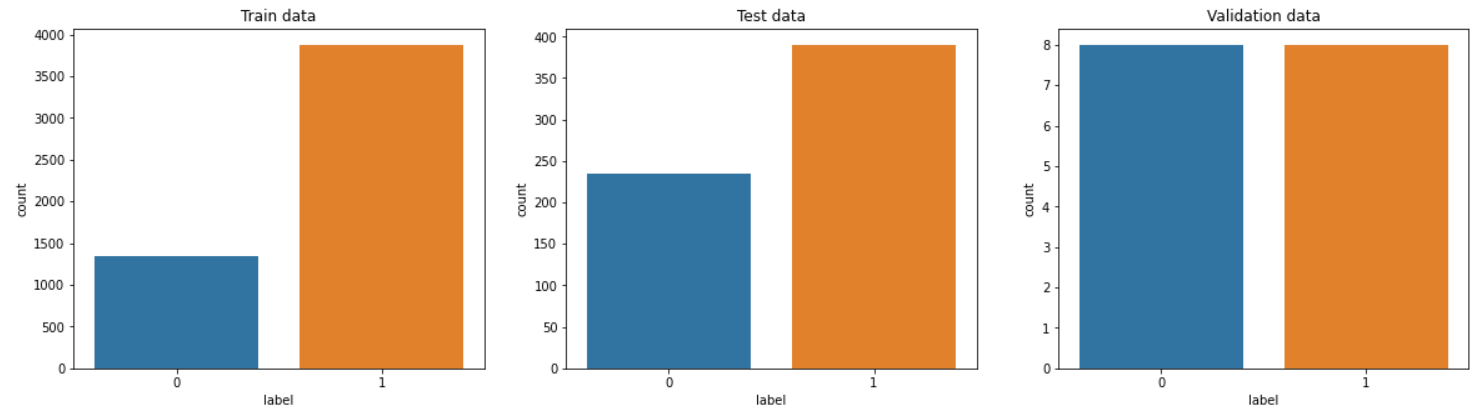


Figure 4. Visualize distribution of cases

*Draw few samples of each case*

In this step we simply want to see few cases of pneumonia and few cases of healthy people. For uninterested person it may not be easy to identify pneumonia on image. You will also notice that images does not have same dimension and must be rescaled to same width & height.

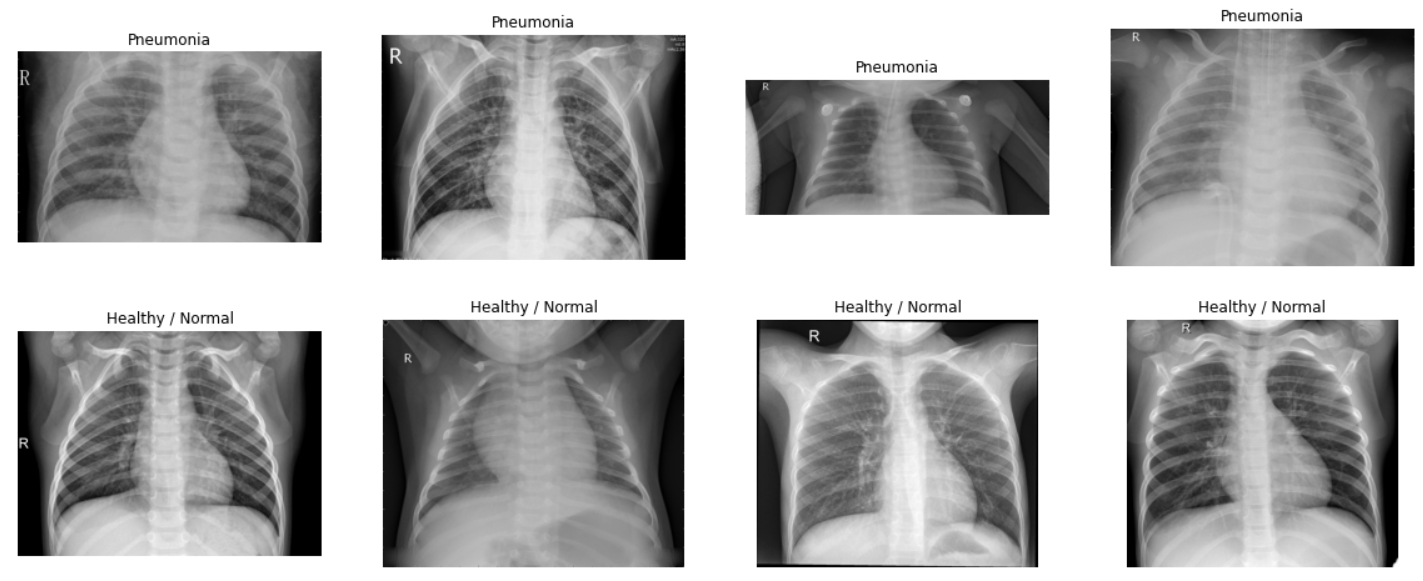


Figure 5. Few samples of each case

*Data preprocessing*

In following section we are going to prepare our data for modeling. Down-size of this technique is that you may run out of memory very quickly... but it's really fast!

To make it simple, define 2 functions.

**Process\_data** - load image, resize it, convert to grayscale, normalize and reshape to dimension required for tensorflow.

**Compose\_dataset** - loop through images, generating 2 numpy arrays. First contains image itself as matrix, second contains label.

Using our functions let's prepare our train, test and validation arrays from dataframes:

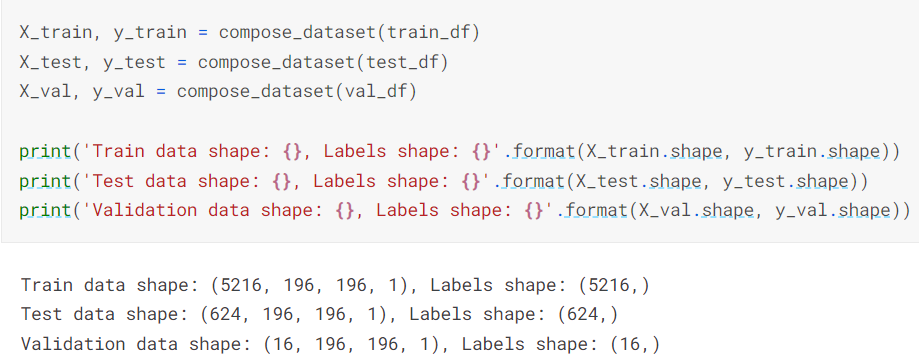


Figure 6. Data preprocessing

***Image augmentation*** is very important to make our model robust to unseen data. It takes each image and modify it slightly so simply said in each epoch, different image (generated from same) is sent to model for training.

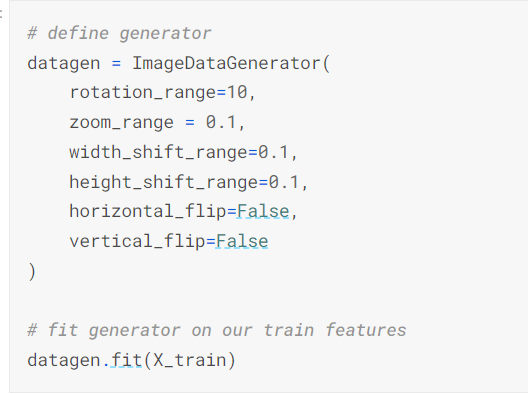


Figure 7. Generator on train features

It may be redundant step, but we are going to convert our 1D array of target labels into 2D array, changing classification from binary to categorical. During my tests, binary classification was slower and less accurate then categorical.

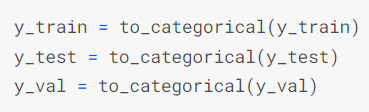


Figure 8. Convert 1D array of target labels into 2D array

1. **RESULT**

*Modelling*

Simple sequential model is used, starting with 2 convolutional networks of kernel size (7,7) and max pooling with pool size (3,3), followed by 2 convolutional networks of kernel size (7,7) and same pool size and finalized by several repeating sets of 2 convolutional networks of kernel size (3,3) with max pooling and pool size (2,2)

Model bit slower, but getting above 92% accuracy that's great result**!**

****

Figure 9. Modelling

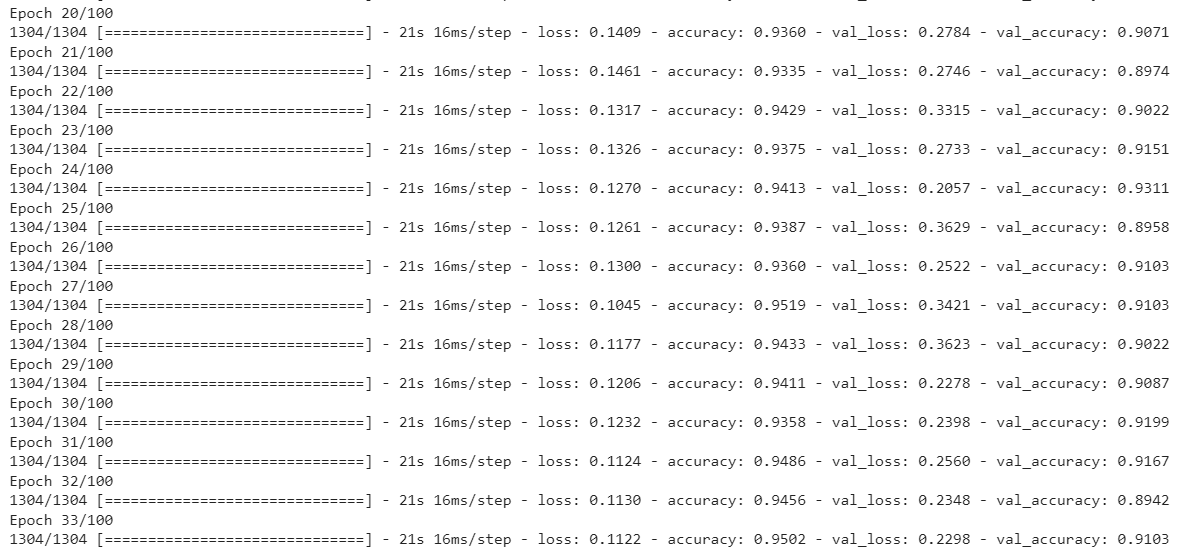
****

Figure 10. Result

*Evaluation*

First, we will quickly check evolution of loss and accuracy over epochs and then draw confusion matrix on test data. Then how our validation set (16 cases) will work with trained model and compare real vs predicted label.

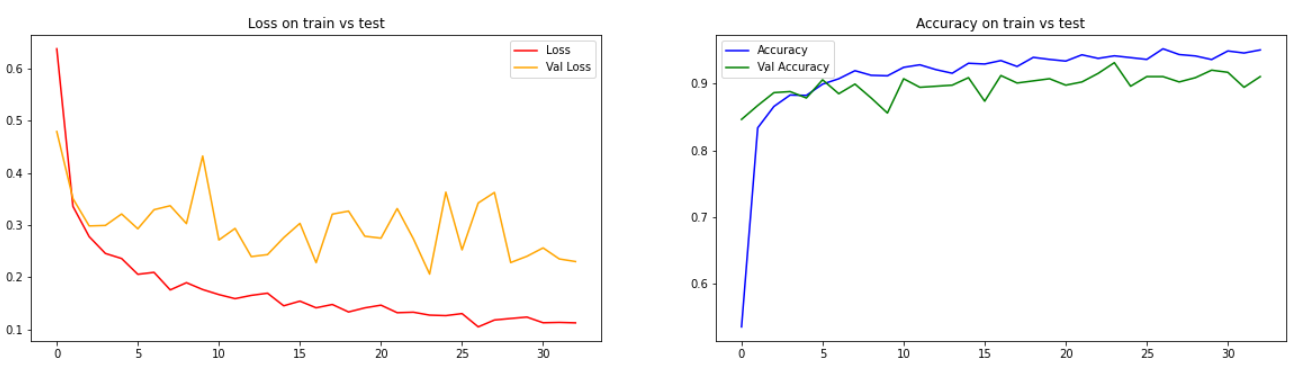


Figure 11. Draw loss on train vs test data evolution

*Confusion matrix on test data*

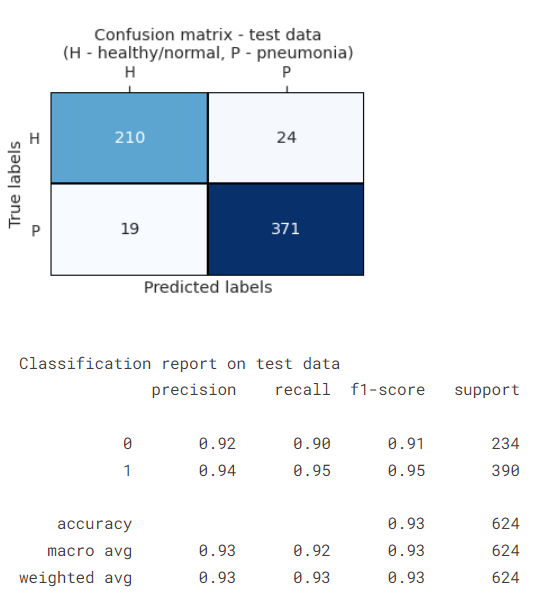
**

Figure 12. Plot confusion matrix as heatmap

*Validation data*

Now let's show images, real & predicted labels:

We use command:

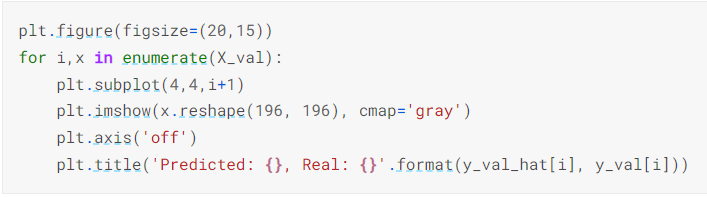
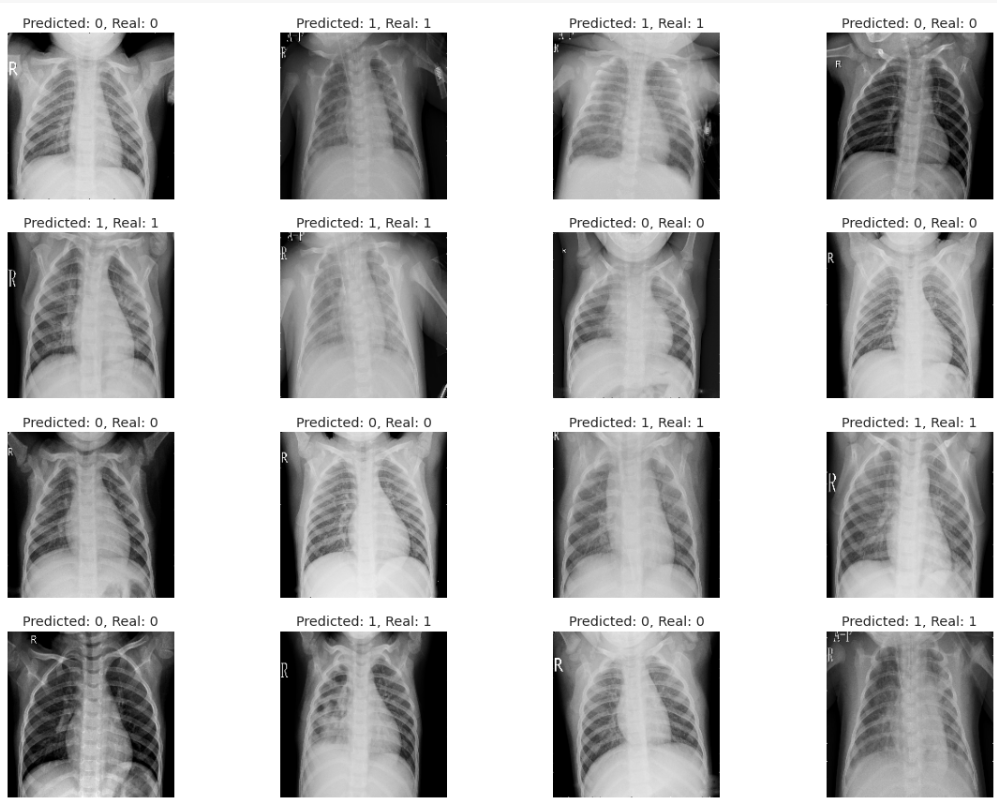


Figure 13. Result of my project

Wow, not bad! Actually great! Even doctor would have problems to classify it as correct but we can do it.

1. References

[1] Github.com [2] Kaggle.com [3] cnvrg.io