Installation

Install all python packages by running `pip install -r requirements.txt`. Also install jupyter-notebook if you want to prepare the dataset for training

Dataset

Download the dataset from here You can also download more datasets and extract them into the folder "COVID-19_Radiography_Dataset", just like the other dataset.

Data Preprocessing

After extracting the files in the same format as the first dataset, you can also get a list of duplicates images by running the script `python compare_images.py`. This will create a .json file that keeps a list of images that are similar to the compared image. See 'similar-covid-images.json' for an example.

You can visualize how many unique/similar images every class has, by running `python stacked_bar_plot.py`.

If you want to delete the similar images, you can remove them by running `python remove_duplicate_images.py`.

Next step is to balance the data, run the script `python balance_datasets.py`. This will create a uniform distribution of the dataset, by removing the additional images from the class with superior amount of data.

When you've merged multiple datasets, it's important to remove all images that are not frontal x-rays. If these images are taken from the side, or from the back, they can either be removed manual, or by inspecting the supplied metadata and writing an script according to the metadata.

When the data processing is finished, the final step is to split the data into training- and validation set. This is done by running the jupyter notebook 'COVID-19 data einlesen.ipynp'. The notebook will create two additonal folders for each class, the training folder and the testing folder. After the data has been split into two sets, we can use the function from 'load_data()' from the script 'data_process.py'.

Training the Network

Open the file train.py. There are a lot of hyperparameters to play around with. You can use different models, different learning rates, apply data augmentation and more by looking at the code and changing the corresponding values. When everything has been set-up to your liking, you can run the script with `python train.py`. This will train the whole network for x epochs (100 by default). All training process(such as parameters) will be stored in the folder 'weights' and 'tensorboard'. You can use the tensorboard for advanced visualization and debugging. See the tensorflow documentations for details. Because the training images take a lot of memory (because of large resolutions), it is essential to use a training_generator, which loads the images in batches. Feel free to look at the file 'training_generator.py' to get an idea of how the images are loaded, and preprocessed further.

When the network has finished training, it will evaluate the network and save the information into the folders 'metrics' (for precision,recall and f1-score), 'cmatrix' (for the confusion matrix of the evaluated model), 'history' (for the complete training- and validation accuracy for every epoch) and 'model' (for the complete model, so we later reconstruct the trained model with the same configuration).

Evaluation

After the training process you can take a look at the folders mentioned above, they contain images such as the confusion matrix, the precision and recall, weights for the model(which we can use for fine-tuning, transfer-learning or something simlar) and the model itself (which we can reconstruct and use for inference). The complete training history can also be plotted and analyzed further by running `python plot.py`. The path to the correct file has to be changed in the .py file.