BAM-NET, Recognizing COVID-19 Symptoms In Lung Images Using Convolutional Neural Networks

Alexander Murtaza **Maksims Svjatoha** **Behnam Yosef Nezhad Arya**

Abstract

The purpose of this report is to experiment with training a convolutional neural network to recognize COVID-19 symptoms in X-ray images. Starting with code structure and with a dataset of 25 Covid-positive x-ray images and 25 Covid-negative x-ray images, a series of convolutional neural networks were designed and tested with various hyperparameters such as learning rate, batch size and epochs, where the best combination of the architecture and hyperparameters is presented as the final result. Furthermore, the dataset of 50 x-ray images was expanded to roughly 1600 images due to good access to x-ray images of healthy lungs, but it became unbalanced, containing 4 non-Covid images for every Covid image. A generative adversarial network, or GAN, was used to generate Covid images and reduced this imbalance to a 2:1 ratio. A total of 2000 images was used for the network. With a batch size of 40, a final network accuracy of 99.5% was achieved after 50 epochs.

1 Introduction

This report describes a project performed as part of the course DD2424 Deep Learning in Data Science at KTH Royal Institutes of Technology. This project consists of creating a deep learning model that classifies X-ray images of lungs as COVID-19 positive or COVID-19 negative. This is done using convolutional neural networks (CNNs or ConvNets) and is implemented in Python using TensorFlow, Keras, and OpenCV. Generative adversarial networks (GAN) are used to generate more images to classify with the CNNs.

2 Related Work

It has been reported that bilateral and peripheral ground glass opacification (GGO) are predominant CT findings in COVID-19 patients and therefore computed tomography imaging is crucial for diagnosis [1]. Deep learning methods is an important tool to extract COVID-19's specific graphical features and provide a clinical diagnosis ahead of the test, thus saving critical time for disease control [2].

Artificial data generated by a Generative Adversarial Network (GAN), can augment real datasets and provide a greater quantity of data for training of large neural networks. It can even balance the dataset, resulting in substantial improvement in classification performance [3]. With a combination of real and artificial data good results can be obtained. Data augmentation using synthesized images increases the diversity of the training dataset and therefore improves generalization performance of deep learning for classification of unseen data.

3 Data

The initial dataset is curated by Dr. Joseph Cohen, a postdoctoral fellow at the University of Montreal [4]. The dataset contained 25 Covid-positive and 25 Covid-negative chest images. The X-ray data was loaded and preprocessed by extracting the class label (either Covid-positive or normal) [5], preprocessing it by converting to RGB channel ordering, and resizing it to 224x224 pixels to be used in the convolutional neural network.

The largest dataset that could be found as of May 2020 was part of the COVID-Net open source project [6]. The images were difficult to re-use in the above mentioned algorithm and an attempt was made to write a secondary convnet for classifying these. The image set contains almost 15000 x-ray images of lungs, with only approximately 150 being images of covid-positive lungs.

In this project, Dr. Joseph Cohen's dataset of 50 images was successfully expanded to 1366 covid-negative lungs and 255 covid-positive lungs using the COVID-19 radiography database from Kaggle [7] and University of California San Diego [8]. The algorithm and data structure were adjusted so all of these could be examined using the same code as for the previous 50 images. The imbalance in the dataset necessitated the use of a GAN in an attempt to balance the dataset.

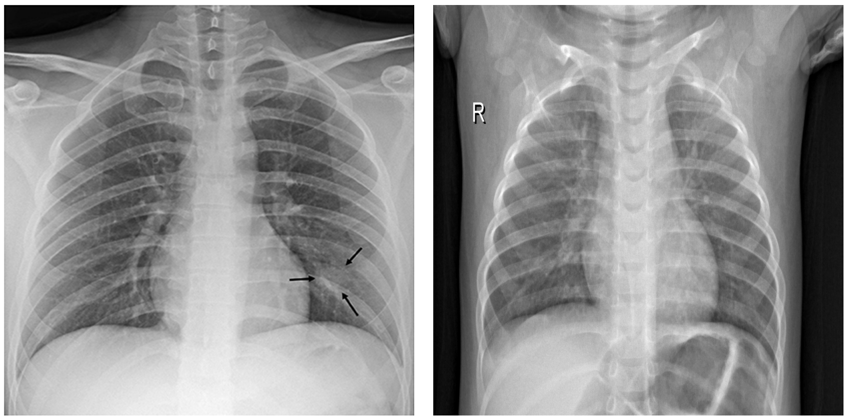

Figure 1: An exempel of a Covid-positive chest image (left), and a Covid-negative one (right). [7]

4 Methods

This project consists of two main parts - training a CNN to recognize X-ray images of covid-positive lungs and training a GAN to generate new images of covid-19 lungs for dataset balancing.

The project code is written in Python and uses, among others, the TensorFlow, keras, scikit-learn, Imutils, Matplotlib and OpenCV libraries. Several datasets were tested.

4.1 Convolutional Neural Network

A convolutional neural network is a class of deep neural networks which has seen success when used in image recognition, generation and classification. It can take in an input image, assign learnable weights and biases to various objects in the image and be able to differentiate one from the other. A convolutional layer has the attributes of having kernels defining width, height and depth of the images and input and output channels.

In this project TensorFlow and Keras were used when working with CNN. Scikit-learn was used to work with data preprocessing and model evaluating. OpenCV was used to load and process images, and Matplotlib was used to plot the results of the model.

4.1.1 Dr. Joseph Cohens dataset

In an initial experiment, Dr Joseph Cohens dataset was used. Command line arguments were parsed and relevant hyperparameters initialized to build the CNN. The initial learning rate was 0.001 and the batch size was set to 8, and the code ran for 25 epochs, and differens values for each parameter were tested.

The extracted (class label with either Covid-positive or normal) X-ray data was pre-processed by converting to RGB channel and resized to 224x224 pixels and then was used in the CNN. Data and label lists were updated accordingly and converted to NumPy arrays. One-hot encoding was applied to the labels and they were split into training and validation set. 20% of the dataset was reserved for validation and the remaining 80% was reserved for training.

For building the model, the VGG16 model [9] was initialized. It instantiates the VGG16 network with weights pre-trained on ImageNet, constructing the head layers, and appending them to the model. Then every layer in the model was frozen so they do not update during the first training process.

For training the CNN, the codes were compiled with the ADAM optimizer. A two-class problem binary_crossentropy loss was used rather than categorical cross-entropy because there are either COVID-19 positive or COVID-19 negative categories. Keras' fit_generator method [10] was called, while passing the chest X-ray data via data augmentation object.

After training the model, the model was evaluated by making predictions on the testing set. scikit-learn's helper utility was used to generate and print out a classification report. Next, a confusion matrix for further statistical evaluation was computed. The confusion matrix was used to derive the accuracy, sensitivity, and specificity. And finally the training history was plotted.

4.1.2 COVID-Net

COVID-Net is an open-source project available through COVID-Net team member Linda Wangs github profile [6]. The database used here includes close to 15000 chest x-ray images, with only around 1% being covid-positive pictures. The network mainly uses Tensorflow for classification. The attempted implementation of this network was perhaps the most experimental one, requiring a lot of code to be modified before usage, the clashes being mainly between different versions of Tensorflow as well as problems encountered when attempting to run the code on a GPU.

Ultimately attempting to modify COVID-Net for the purposes of this project was abandoned as a concept when Dr. J. Cohens algorithm was successfully expanded to work on the larger COVID-19 sets from Kaggle. The COVID-Net approach took roughly half a day to run 10 epochs on a CPU and encountered memory serious issues when running on a GPU and was hence vastly inefficient despite best attempts at making it work.

4.2 Generative Adversarial Network

To increase the amount of Covid-19-positive chest x-ray images and make the CNN more accurate, a Deep Convolutional Generative Adversarial Network (DCGAN) was used. A DCGAN consists of two CNNs, the generator, that generates new data and the discriminator that will feed with both real and generated picture. The output of the discriminator is a probability label if the picture comes from the real dataset or a picture that was generated by the generator. The aim with a GAN is to make it difficult for the discriminator to distinguish between real and fake picture, i.e. maximize the probability that the discriminator classify a fake picture as real i.e the generator should learn to generate picture that the discriminator interpret as real. [11]

Many GAN models suffer problems, e.g. the model parameters oscillate, become unstable and never converge to a final value. Another problem that can occurs when the training of the discriminators is successful which can lead to that the generator gradient vanishes and it learns nothing. [12]

The discriminator was fed 161 pictures from a database that was created by researcher from Qatar the University of Dhaka. [7]

Since the larger dataset used with final CNN was unbalanced, using a GAN to even out the difference was considered a sound approach.

5 Experiments

5.1 Dr J. Cohen's dataset, initial experiments

Initially, Dr. Joseph Cohen dataset and code were tested as-is, with only minor alterations to different hyperparameters, to make sure the code is working as intended. Figures 2 and 3 show a training history for 25 Covid-19 and 25 normal images with the default hyperparameters with batch sizes 8, 4 and 2 (left to right). The default code puts loss and accuracy in the same plot, but they are separated in later experiments. The learning rates are 0.001 and 0.01 and the code ran for 100 epochs. As seen, the very low batch sizes give a spikier, more erratic behavior, but seemingly quicker convergence. Surprisingly, the validation accuracy reaches over 95% within this time. As expected, the higher learning rate also leads a more erratic behavior.

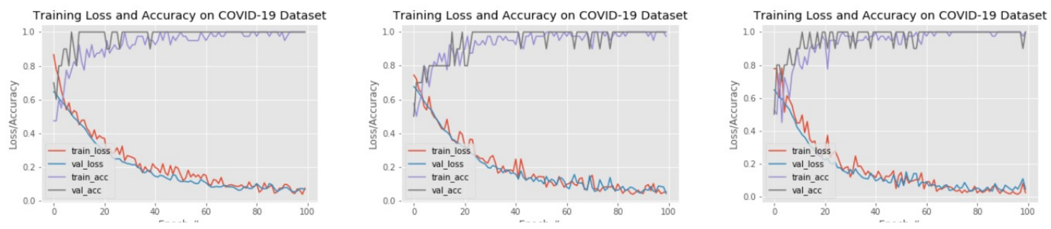

Figure 2: The CNN model for the 50 original images. Learning rate:0.001; batch sizes:8, 4, 2; epochs:100. As can be seen in train_acc, smaller batch sizes show earlier and better convergence.

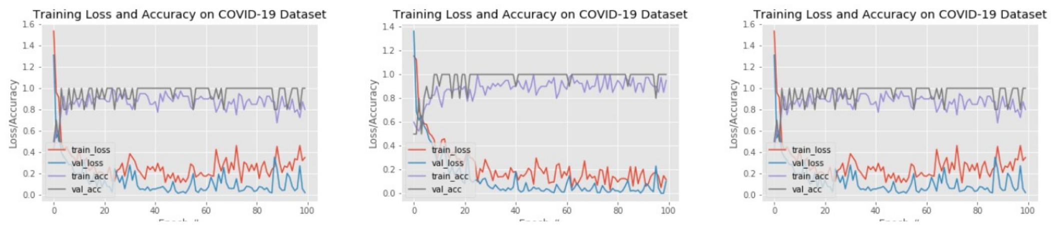

Figure 3: The CNN model for the 50 original images. Learning rate:0.01; batch sizes:8, 4, 2; epochs:100.

Having made sure that the code works and produces different and expected results, further modifications were made, now taking into account the momentum, the decay of the learning rate, batch normalization and data distribution between training and testing sets. The dataset was also successfully expanded upon to roughly 1600 images using the COVID-19 radiography database[7] and increased even further up to exactly 2000 images, using GAN-generated images of Covid-positive lungs. This somewhat balanced the set from 1366 images of healthy lungs and 295 of covid-positive lungs, to a ratio of 1366/634, or almost a 2:1 ratio.

5.2 Dr J. Cohen's dataset, expanding the dataset

Training the model on the larger dataset without GAN images was attempted but ultimately abandoned in the interest of time. Trying to plot the dataset that is balanced using GANs was deemed more interesting to study. Further plots will have the accuracy and loss data separated for clarity. The results for the model that was trained with the larger dataset is presented in the following graphs, regrettably with the worse learning rate of 0.01:

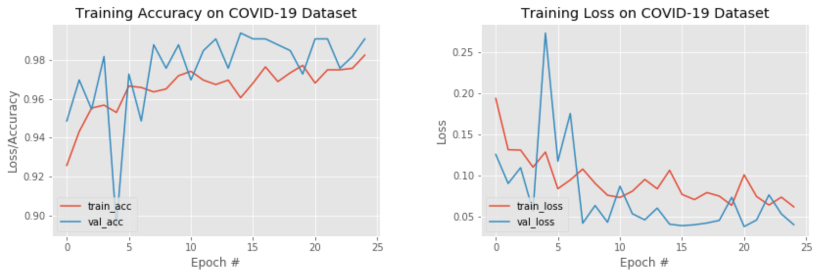

Figure 4: Accuracy and loss plots of the larger dataset without GAN images. Learning rate = 0.01, batch size = 8

5.3 GAN implementation

A GAN algorithm was also implemented to balance the amount of images of covid-19 positive versus negative lungs. The final algorithm ran for 1000 epochs, the batch size was 32 and the learning rate for both the generator and the discriminator was 0.0002 with Adam optimizer. For each epoch 8 images were produced.

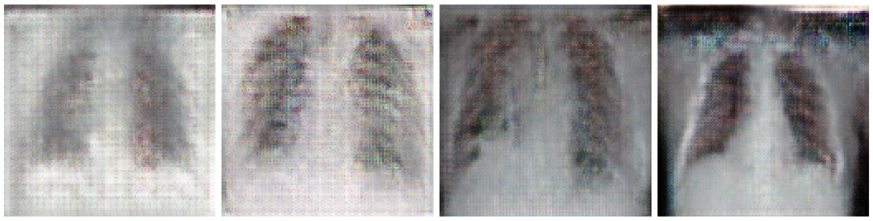

Figure 5: The progress of images of covid-19 lungs, after epoch 298, 525, 721 and 1000, from left to right

As mentioned earlier, it can be difficult to train a GAN. As can be seen in the Figure 6 the discriminator can distinguish between real and fake images, but the generator is not good enough to generate "real" images.

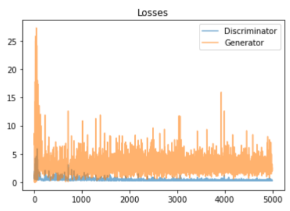

Figure 6: The loss plots for the GAN

5.4 Dr J. Cohen's dataset, adding GAN images

These experiments are more thorough and examine a much wider variety of parameters. Batch normalization was added and the decay parameter was decoupled from the amount of epochs and the learning rate itself, so it can be chosen directly. A few different network architectures were tested. Experiments were conducted from a "base case" shown in figure 7.

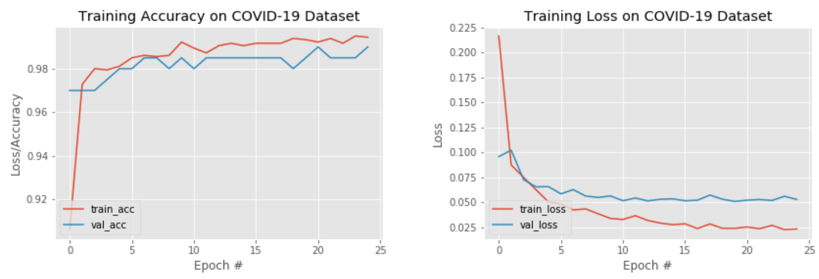

Figure 7: Accuracy and loss plots of the larger dataset with GAN images. Learning rate = 0.001, momentum = 0.833 or 5/6, decay rate = 0.01, dropout rate = 0.25, batch size = 40, 25 epochs. 10% of the data is used for validation.

After documenting the base case, various hyperparameters were tested, including decay rate, learning rate, validation data percentage and dropout percentage and ran for 25 epochs. The best parameters from each case were taken and combined. Note the significant improvement over the base case.

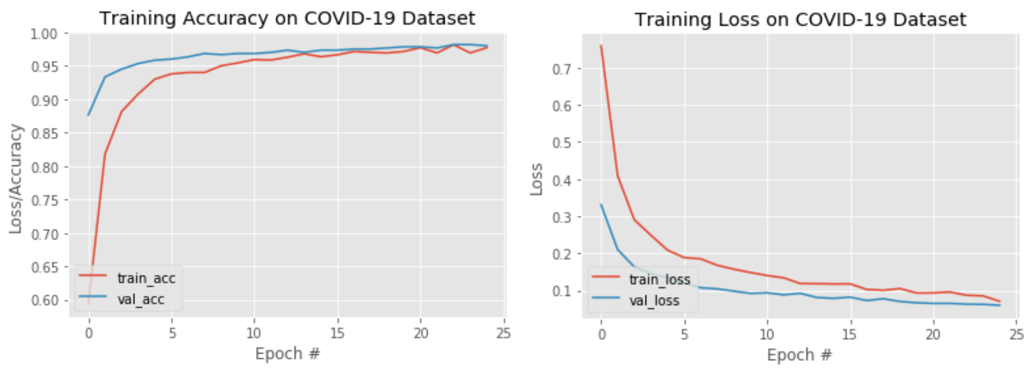

Figure 8: Accuracy and loss plots of the best hypermeters from testing. Learning rate = 1e-4, momentum = 0.833 or 5/6, decay rate = 1e-5, dropout rate 0.5, batch size = 40, 25 epochs. 30% of the data is used for validation.

The next step was to experiment with the network architecture by adding various layers. After roughly 10 different architectures, the best architecture featured two additional convolutional layers (originally only 1 was used) and two additional regular densely connected neural layers ((originally only 1 was used here as well). Batch normalization was added and a mix och relu and leaky relu activation functions was used. Figure 9 shows the resulting accuracy and loss graphs when using the best network architecture with the best hyperparameters from before. The highest validation accuracy so far was achieved here - 99.5%. This means that 1 out of 200 validation images was missclassified.

5.5 Final experiment

The final experiment entails using the best architecture with the best hyperparameters and using the code for 50 epochs - once with GAN images and once without. The dataset without GAN images performed slightly worse (99.50% vs 99.19%). The results are presented in figures 10 and 11:

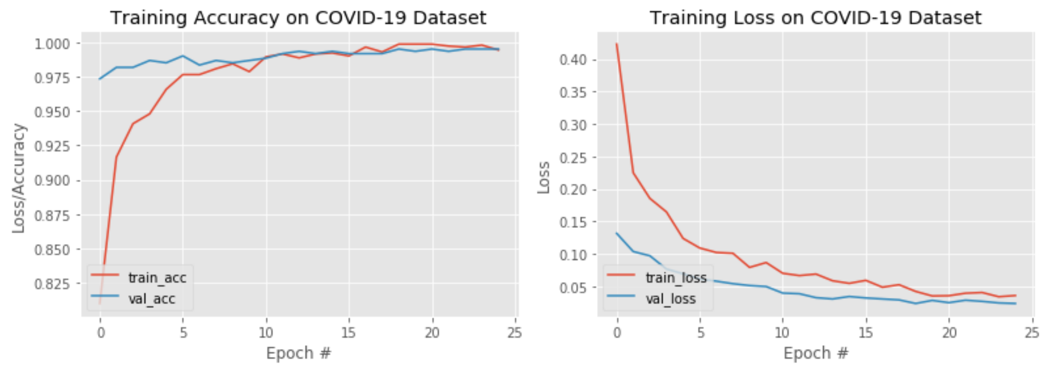

Figure 9: Accuracy and loss plots of the best hypermeters with the best network architecture. Learning rate = 1e-4, momentum = 0.833 or 5/6, decay rate = 1e-5, dropout rate 0.5, batch size = 40, 25 epochs. 30% of the data is used for validation. 3 convolutional layers, 3 regular "dense" layers.

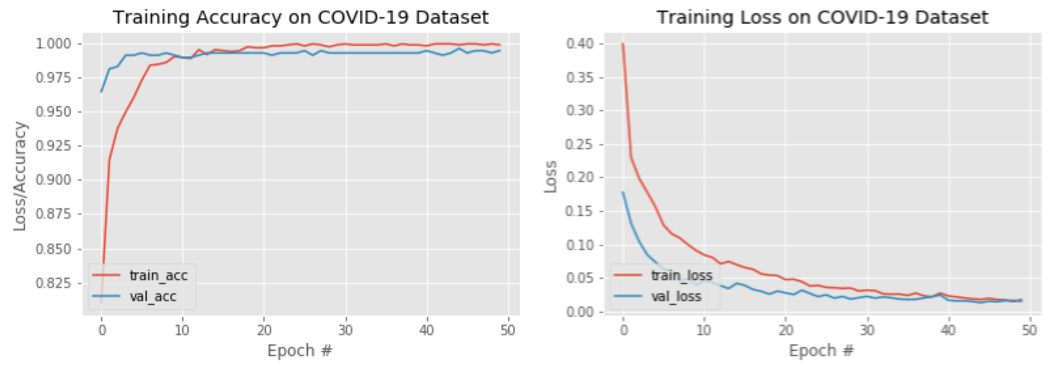

Figure 10: Accuracy and loss plots of the best hypermeters with the best network architecture with GAN images. Learning rate = 1e-4, momentum = 0.833 or 5/6, decay rate = 1e-5, dropout rate 0.5, batch size = 40, 50 epochs. 30% of the data is used for validation. 3 convolutional layers, 3 regular "dense" layers.

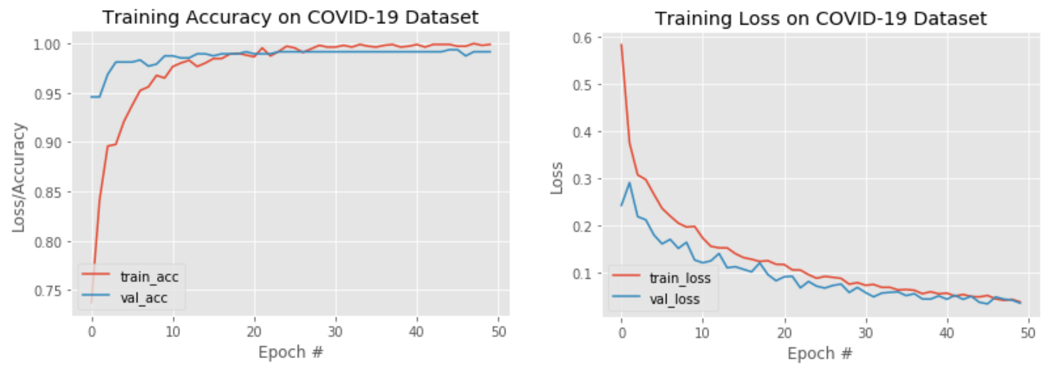

Figure 11: Accuracy and loss plots of the best hypermeters with the best network architecture without GAN images. Learning rate = 1e-4, momentum = 0.833 or 5/6, decay rate = 1e-5, dropout rate 0.5, batch size = 40, 50 epochs. 30% of the data is used for validation. 3 convolutional layers, 3 regular "dense" layers.

6 Conclusion

The expansion of the original database of 50 images, final CNN network architecture and choice of hyperparameters is a solid start for further work. Incorporating more hyperparameters, trying different activation functions and further experimenting with the network architecture might yield even better performance. However, expanding the dataset to include more confirmed Covid cases is likely a priority. The high final accuracy is possibly a cause for concern, or at least skepticism, since it might suggest that the data is getting overfit.

Having enough computing power to run both the CNN and GAN codes for longer, and train the model on the larger dataset with and without GAN images would likely also yield more interesting results, as an increased amount of epochs tends to give a higher accuracy for both networks. It would be preferable to involve people with medical expertise as well to assert the quality of the process.

In summary: for future work, more expertise in medicine, advanced deep learning or various other algorithms is needed to make the results more valid. Better GAN images where human eyes do not recognize the difference between them and real images are required. A confirmation is needed by outside expertise that the CNN is indeed highly accurate.

References

[1] Lung Infection Quantification of COVID-19 in CT Images with Deep Learning. Department of Radiology, Shanghai Public Health Clinical Center
url: $https : //arxiv.org/ftp/arxiv/papers/2003/2003.04655.pdf$ (visited on 2020-05-15).

[2] A deep learning algorithm using CT images to screen for Corona Virus Disease (COVID-19). Tianjin Medical University Cancer Institute and Hospital and National Supercomputer Center in Tianjin
url: $https : //www.medrxiv.org/content/medrxiv/early/2020/02/17/2020.02.14.20023028.full.pdf$ (visited on 2020-05-15).

[3] GENERALIZATION OF DEEP NEURAL NETWORKS FOR CHEST PATHOLOGY CLASSIFICATION IN X-RAYS USING GENERATIVE ADVERSARIAL NETWORKS
url: $https : //arxiv.org/pdf/1712.01636.pdf?fbclid = IwAR3WG2Hkb - R2v - YesBP2u2xVuU0KX1gzAkfhm5cbdsme18jA9ftYoi - DCfU$ (visited on 2020-05-15).

[4] Dr. Joseph Cohen
url: $https : //www.dropbox.com/s/j364809im78rqqv/compressed_dataset.zip?dl = 0$ (visited on 2020-05-13).

[5] Medium
url: $https : //medium.com/better - programming/how - to - detect - coronavirus - using - deep - learning - 1568332c728$ (visited on 2020-05-15).

[6] Github
url: https://github.com/lindawangg/COVID-Net/ (visited on 2020-05-17).

[7] Kaggle, COVID-19 Radiography Database
url: $https : //www.kaggle.com/tawsifurrahman/covid19 - radiography - database?$ (visited on 2020-05-09).

[8] Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification
url: $https : //data.mendeley.com/datasets/rscbjbr9sj/2?fbclid = IwAR0crdl49LtzCGwUkfFOensXEiigKM6vWTxw3jXHQH8ZgevopSwCKAi8bWg$ (visited on 2020-05-17).

[9] Keras Applications.
url: $https : //keras.io/api/applications/$ (visited on 2020-05-13).

[10] The Sequential class
url: $https : //keras.io/api/models/sequential/$ (visited on 2020-05-13).

[11] Generative Adversarial Nets
url: $https : //papers.nips.cc/paper/5423 - generative - adversarial - nets.pdf$ (visited on 2020-05-15).

[12] Medium
url: $https : //medium.com/@jonathan_hui/gan - why - it - is - so - hard - to - train - generative - advisory - networks - 819a86b3750b$ (visited on 2020-05-15).