Screening X-ray Images for COVID-19 Infections

Lorenzo Loconte
University of Bari Aldo Moro
l.loconte5@studenti.uniba.it
744328

Giuseppe Colavito
University of Bari Aldo Moro
g.colavito2@studenti.uniba.it
736047

***Abstract*—In this work we propose some approaches to solve the task of predicting COVID-19 infections using X-ray images of the chest of the patients (CXR images). We train different state-of-the-art neural networks and compare the achieved results, based on several metrics, performance measures and dataset preprocessing.**

***Index Terms*—Computer Vision, Medical Imaging, COVID-19**

I. Introduction

The goal of this work is to understand if a patient has a COVID-19 infection, given an X-ray image of the chest. It is a binary classification task, with a very unbalanced dataset. In this kind of task, it is important to don't have *false-negatives*, since an infected patient can have several health problems and can help the infection spreading. We used some state-of-the-art neural network, such as AlexNet, ResNet, VGG, DenseNet and InceptionV3, to produce the results.

II. Related work

Diagnosis or evaluation of COVID-19 from CXR (Chest X-ray) is a topic that has attracted a lot of interest from researchers. [1] made an overview of studies on this field. For example, [2] predicts the disease severity, similarly [3] predicts the disease progression by comparing an exam with the previous exams of the patient.

There are many studies using deep learning to make Image-level Predictions from CXRs. A large proportion of the studies use pre-trained standard architectures that can easily be found in deep learning libraries such as Tensorflow or PyTorch. These architectures are commonly ResNet [4], DenseNet [5], InceptionV3 [6], VGG [7], or AlexNet [8]. The choice of models depth (such as ResNet-18, ResNet-50, DenseNet-121, DenseNet-161) also varies between studies as there is no standard in this design choice. Most of those studies do not introduce methodological novelty but report or compare the performances of multiple architectures on a given task.

Just like the model depth and architecture, there are many factors that affect the performance of a deep learning model. The effect of various data augmentation and input preprocessing methods are evaluated by [9] and [10].

Various pretraining schemes are evaluated by [11], [12]. More sophisticated pre-processing steps to improve model performance include bone suppression [12] and lung cropping [13].

III. Dataset

A. Introduction

The COVIDx CXR-2 dataset is composed of 15951 training examples and 400 testing examples. In the training set, the two classes (positive and negative) are unbalanced. We have 2158 positive examples and 13793 negative examples. However, the test set is perfectly balanced. Specifically, the test set consists of 200 COVID-19 positive images and 200 COVID-19 negative images from 100 no pneumonia and 100 non-COVID-19 pneumonia patient cases. The class distributions are available in Table I. The dataset is freely available on Kaggle.

Figure 1 shows the frequency histogram of training images dimensions (in a logarithmic scale). Moreover, Figure 2 and Figure 3 shows some examples of the dataset with the corresponding cumulative histograms.

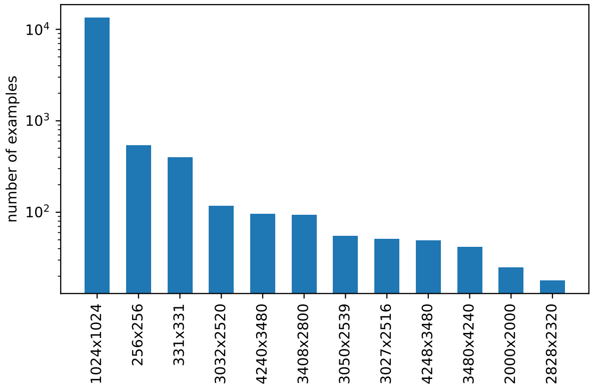


Fig. 1. Frequency histogram of training images dimensions (in a logarithmic scale). Image dimensions having frequency less than 10 are discarded in plotting this figure.

A study on this dataset has been made by Pavlova et al. [14], using ResNet50 and a custom neural network called `COVID-Net CXR-2`. We will use this study to compare the results in the next sections. Other works on the CXR-2 dataset are available on Kaggle, but they are using a questionable split of the training set to avoid the problem of unbalanced data to train their models.

B. Preprocessing

The first preprocessing steps applied to the whole dataset are center-cropping, resizing and converting to greyscale.

TABLE I
THE DATASETS DISTRIBUTION.

	Train Set	**Test Set**
Positive	2158 (14%)	200 (50%)
Negative	13793 (86%)	200 (50%)
Total	15951	400

Greyscale conversion is necessary since there are some CXR images that are not in greyscale, e.g. in "blueish" color. However, since there can be various text information on the top border of the images, the 8% of top most pixels are cropped as well, as done by Pavlova et al. [14]. This is done in order to make the training process more fair.

Usually the images have an empty border and, sometimes, they have other parts of the body that are not the chest (e.g. the second example in Figure 2). Generally, the chest is present on the center part of the image. Therefore we decided to center crop the images. A resizing step is necessary to have all the images with the same size and to fit the input size requirements of the neural networks we will use for the task. So, all the images are resized to 224x224 using bicubic interpolation.

Figure 2 and Figure 3 show negative and positive examples respectively. Such figures also include a visualization of the cumulative histogram and the corresponding *histogram equalized* image. In this work we are going to explore if equalizing the histogram of an image improves performance on the classification task. Other preprocessing techniques, such as *sharpening*, are not used because of possible introduction of noise in the input images.

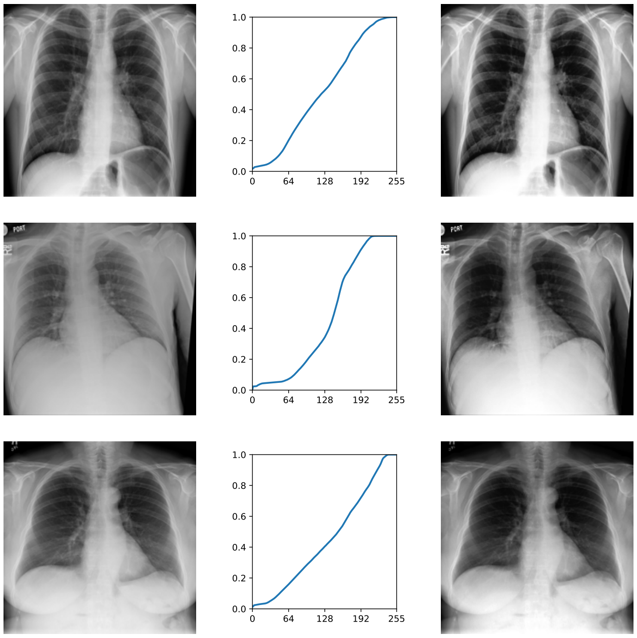


Fig. 2. COVID-19 negative chest X-ray images (on the left), the cumulative histograms (at the center) and histogram equalized images (on the right).

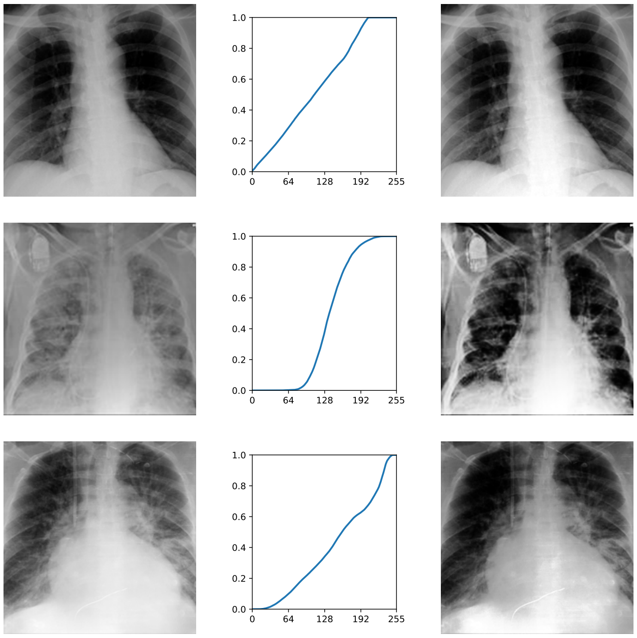


Fig. 3. COVID-19 positive chest X-ray images (on the left), the cumulative histograms (at the center) and histogram equalized images (on the right).

IV. NEURAL NETWORKS

The networks are trained from scratch, i.e. no pretraining is used when instancing the models. We decided not to use pretraining such as the widespread *ImageNet* because there exists an obvious domain gap between natural images and medical images. However, the use or not of the ImageNet pretraining on the medical domain is still an open research question. As shown in [15], pretraining with ImageNet on the medical domain offers limited performance gains and much smaller architectures can perform comparably to the standard ImageNet models. The neural networks we trained for this task are reported below and summarized in the next sections:

- AlexNet
- VGG16
- ResNet50
- DenseNet121
- InceptionV3

A. AlexNet

AlexNet [8] is a convolutional neural network (CNN) without skip connections consisting of several convolutional layers in sequence. The non-linearity applied after each convolutional layer is ReLU, in order to solve the *vanishing gradient* problem that occurs in the presence of other activation functions (e.g. sigmoid). For this reason, the units of this network are also called *non-saturating* neurons. There are a total of five convolutional layers. In order to reduce the dimensionality of feature maps at the hidden convolutional layers, max-pooling is used after the first two convolutional layers and after the last convolutional layer.

After the features extractor model consisting of convolutional layers, a feed-forward classifier of three densely-connected layers is placed. However, in order to reduce overfitting, *dropout* is used. Dropout consists of setting to zero the output of each hidden neuron with a certain probability. In the original work, a probability of 0.5 is used. It's important to notice that no batch normalization layers were used in the original work.

B. VGG

VGG [7] is a class of CNNs consisting of five stacks of convolutional layers. Between every stack, max-pooling is applied in order to reduce the dimensionality of feature maps. Every stack has at least two convolutional layers. The network ends with two 4096 fully connected layers, a 1000 fully connected layer and softmax activation (since the network was originally introduced in order to classify ImageNet images).

VGG16 uses several convolutional layers with kernel size 3×3. In some configurations, also 1×1 kernels are used to increase non-linearity. The stacks are used instead of single but bigger sized convolutional layers. This means that we have more rectification layers, which will make the decision function more discriminative. This setting also lowers the parameters to learn.

Figure 4 shows an illustration of the VGG16 neural network architecture.

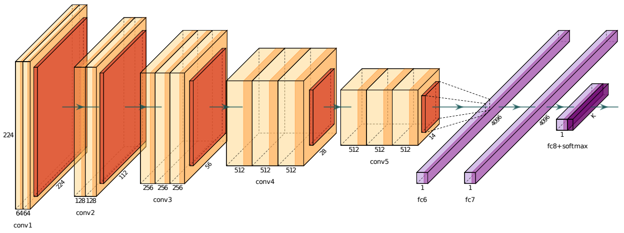


Fig. 4. Illustration of the VGG16 neural network architecture (better view when zoomed in).

C. ResNet

ResNet [4] is a class of CNNs that is a special case of *residual networks*. A residual network is a network in which, instead of approximating a mapping $H(x)$, we let the network approximate a residual function $F(x) := H(x) - x$. This formulation is used to address the *degradation problem*. With the network depth increasing, the accuracy gets saturated and then degrades rapidly. The ResNet architecture uses 3×3 convolutional layers with batch normalization [16] and ends with a global average pooling layer and a fully-connected layer with softmax activation function. Shortcut connections are inserted. The identity shortcuts can be directly used when the input and output are of the same dimensions. When the dimension increase, an identity shortcut with 0-padding or a projection shortcut can be used.

Figure 5 shows two kinds of residual convolutional blocks based on residual connections: *basic* and *bottleneck*. Bottleneck residual blocks differ from basic residual blocks for the fact that it is composed by 1×1 convolutions. For this reason, bottleneck residual blocks are more efficient and therefore permitting to better scale deeper residual architectures.

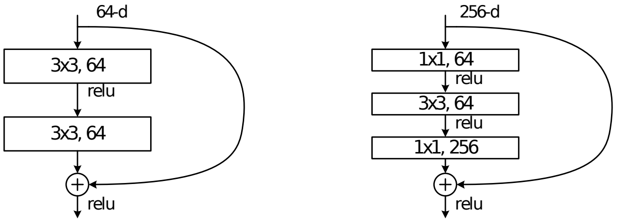


Fig. 5. Illustration of both a basic residual block (on the left) and a bottleneck residual block (on the right).

D. DenseNet

DenseNet [5] is a class of CNNs in which every layer has direct connections to all subsequent layers. So, every layer receives the feature maps of all preceding layers as input. The network is divided into multiple densely connected dense blocks, interleaved by transition layers, which do convolution and pooling. The convolution kernels size used in the DenseNet architecture are 1×1 and 3×3. Generally, a DenseNet architecture is composed by three or four dense blocks with their transition layers, and ends with average pooling, a 1000 fully connected layer and softmax activation function (since the network was originally introduced in order to classify ImageNet images).

Figure 6 shows a densely connected block that is used in DenseNet neural network architectures.

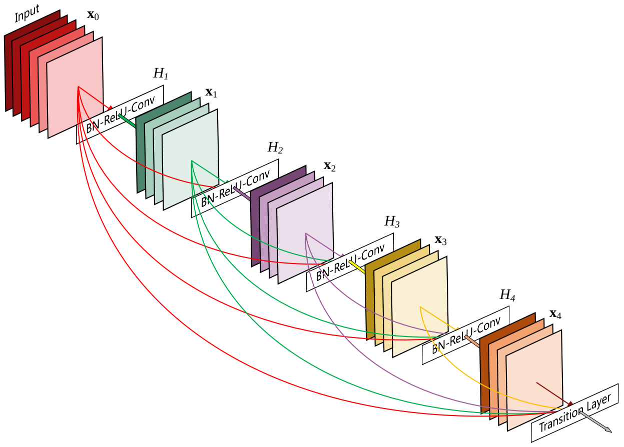


Fig. 6. Illustration of a densely connected block.

E. InceptionV3

InceptionV3 [6] is a convolutional neural network which has some Inception modules. An inception module is a layer that performs multiple convolutions at the same level and returns a concatenation of all of them.

The InceptionV3 architecture from the Inception family makes several improvements including using *label smoothing*, factorized 5×5 and 7×7 convolutions, and the use of auxiliary classifiers. *Auxiliary Classifiers* are type of architectural component that seek to improve the convergence of very deep networks. They are classifier heads we attach to layers before the end of the network. The motivation is to push useful gradients to the lower layers to make them immediately useful and improve the convergence during training by combatting the vanishing gradient problem.

Figure 7 shows both the original Inception module by Szegedy et al. [17] and the improved InceptionV3 module by Szegedy et al. [6]. As one can notice, the InceptionV3 module differs from the InceptionV1 module from the fact that it decomposes 5×5 convolutions into two 3×3 convolutions. Moreover it decomposes 3×3 convolutions into two asymmetric 3×1 and 1×3 convolutions. This method significantly decrease the number of parameters of the model.

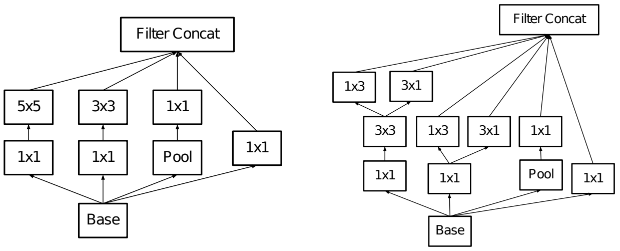


Fig. 7. The original InceptionV1 module (on the left) and the improved InceptionV3 module (on the right). Note that 5×5 convolutions are replaced by chaining two 3×3 convolutions. Moreover 3×3 convolutions are replaced by concatenating the results of 3×1 and 1×3 asymmetric convolutions.

V. Methodology

A. Experimental Setting

Data augmentation has been used because, in general, having augmented data can help to achieve a lower generalization error and higher classification accuracy, as shown in [18], [9]. The dataset is augmented using random scaling, with a factor sampled uniformly from $[0.9, 1.1]$, and random translation on both the X and Y axes, with two factors sampled uniformly from $[-0.1, 0.1]$. Moreover, random rotations with a small degrees value sampled uniformly from $[-10, 10]$ are introduced.

For ResNet50, DenseNet121 and InceptionV3, the Adam optimizer [19] with default hyperparameters has been used. However, for AlexNet and VGG16 the Adam optimizer with learning rate $1 \cdot 10^{-4}$ has been used in order to make them converge. This is due to the fact that, for example, VGG16 is a very complex model without *skip connections*, in contrast with ResNet50 and DenseNet121.

It's important to say that InceptionV3 was trained without using auxiliary classifiers, in order to make the training process more fair. Moreover, images are upsampled using bilinear interpolation from 224×224 to 299×299 in order to match the input size requirements of InceptionV3.

For all the specified models, a batch size of 32 and no L_2 regularization has been used. Furthermore, in order to mitigate the problem of unbalanced training data, a *weighted binary cross-entropy* is used as loss function. The weights of the loss function are inversely proportional with respect to the class frequencies in the training data. The training process is stopped either after 100 epochs or if no improvements on the validation loss happen after 20 consecutive epochs.

B. Tools & Hardware

To train the networks we decided to use the Python programming language with the PyTorch library. We used Google Colab to train the networks. The code is fully available on GitHub.

The average inference time of the models on a single image are evaluated on a single GPU and on a CPU. The GPU used is a NVidia GTX-1660 and the CPU is a Intel 4460 quad-core at 3.4GHz.

VI. Evaluation

Each model has been evaluated on the test set according to performance and complexity:

1) Precision, recall and F_1 metrics on both the negative and positive class.
2) Inference time on both a single GPU and CPU and number of parameters of the model.

Table II and III show the performance metrics on negative samples and positive samples respectively. Table IV shows macro-averaged performance metrics among the two classes. Best results are showed in bold. Finally, Table V show the complexity metrics (number of parameters and inference time) of the models. Specifically, average inference time per image on both CPU and GPU are reported.

TABLE II
Performance Metrics on the Negative Class.

	With Hist. Equalization			**Without Hist. Equalization**		
Architecture	Precision	Recall	F_1	Precision	Recall	F_1
AlexNet	0.843	0.995	0.913	0.866	1.000	0.928
VGG16	0.787	0.995	0.879	0.752	1.000	0.858
ResNet50	**0.964**	**0.940**	**0.952**	**0.905**	**0.995**	**0.948**
DenseNet121	**0.955**	**0.955**	**0.955**	0.843	0.995	0.913
InceptionV3	0.873	0.995	0.930	**0.896**	**0.995**	**0.943**

TABLE III
Performance Metrics on the Positive Class.

	With Hist. Equalization			**Without Hist. Equalization**		
Architecture	Precision	Recall	F_1	Precision	Recall	F_1
AlexNet	0.994	0.815	0.896	1.000	0.845	0.916
VGG16	0.993	0.730	0.841	1.000	0.845	0.916
ResNet50	**0.941**	**0.965**	**0.953**	**0.994**	**0.895**	**0.942**
DenseNet121	**0.955**	**0.955**	**0.955**	0.994	0.815	0.896
InceptionV3	0.994	0.855	0.919	**0.994**	**0.885**	**0.937**

TABLE IV
MACRO AVERAGED PERFORMANCE METRICS.

	With Hist. Equalization			Without Hist. Equalization		
Architecture	Precision	Recall	F_1	Precision	Recall	F_1
AlexNet	0.918	0.905	0.904	0.932	0.923	0.922
VGG16	0.890	0.863	0.860	0.876	0.835	0.830
ResNet50	**0.953**	**0.952**	**0.952**	**0.949**	**0.945**	**0.945**
DenseNet121	**0.955**	**0.955**	**0.955**	0.919	0.905	0.904
InceptionV3	0.933	0.925	0.924	**0.945**	**0.940**	**0.940**

TABLE V
MODELS COMPLEXITY METRICS.

Architecture	**# params (M)**	**CPU time (ms)**	**GPU time (ms)**
AlexNet	57.0	21.5	2.6
VGG16	134.3	171.0	12.7
ResNet50	23.5	95.1	11.0
DenseNet121	7.0	84.1	21.5
InceptionV3	21.8	116.4	18.3

As we can see, the histogram equalization can be useful for this task, but some models have better performance on the raw images. The best models for the task are DenseNet121 and ResNet50 with the equalized images, which achieved very similar performance. In general, the performance for the positive and the negative class are quite similar for every model, which is really good considering the unbalanced setting of the training data. Furthermore, even if ResNet50 has more parameters, it has similar inference time to DenseNet121 on both CPU and GPU.

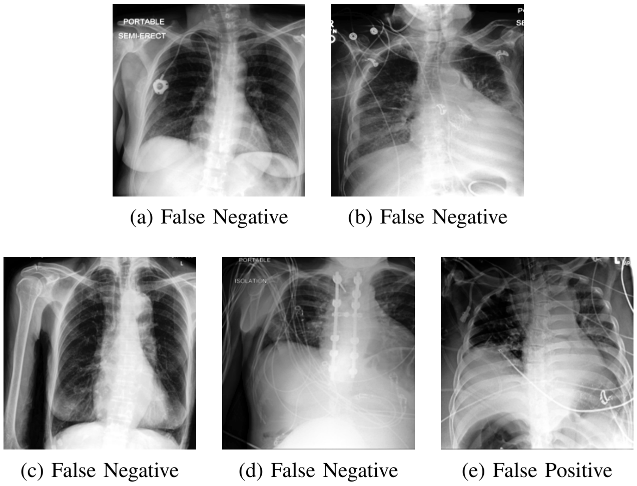
We analyzed the prediction errors to understand if there were some particular images harder to classify than the majority. We found out that images in which the patient has some medical devices on his body are harder to classify, probably because they hide some parts of the chest. Other images where not well centered, or the patient's chest was a bit rotated.

Figure 8 shows four CXR images that have been misclassified either as false negatives or false positives by both ResNet50 and DenseNet121.

Table VI shows a comparison between our best models and the COVID-Net CXR-2 [14]. As we can see, the performance are very similar even if the model used by Pavlova et al. is more complex and has a bigger input size, hence a higher inference time, as shown in Table VII.

TABLE VI
COMPARISON OF THE BEST PERFORMING MODELS USING METRICS ON THE POSITIVE CLASS.

Architecture	**Input size**	**Precision**	**Recall**	**Accuracy**
ResNet50	224 × 224	0.941	0.965	0.952
DenseNet121	224 × 224	0.955	0.955	0.955
COVID-Net CXR-2	480 × 480	0.970	0.955	0.963



(a) False Negative (b) False Negative

(c) False Negative (d) False Negative (e) False Positive

Fig. 8. Five CXR images (already histogram equalized) that have been misclassified by the best performing models. Four of them are false negatives and one of them is false positive. (a), (b) and (d) give the idea that CXR images on a patient having medical devices on their body make the classification task harder. Moreover, (c) and (d) give the idea that CXR images that are not centered very well make the classification task more difficult. Finally, (e) is a particularly rotated CXR image of a patient with several medical devices attached.

TABLE VII
COMPARISON OF THE BEST PERFORMING MODELS ON NUMBER OF PARAMETERS AND INFERENCE TIME ON GPU.

Architecture	**# params (M)**	**Inference Time (ms)**
ResNet50	23.5	11.0
DenseNet121	**7.0**	**21.5**
COVID-Net CXR-2	8.8	71.1

VII. CONCLUSIONS AND FUTURE WORKS

The presence of medical annotations on the chest x-ray images might introduce a bias in the classifications given by the models. Analyzing the dataset, we discovered that some data sources which provided images belonging to only one class have the same medical annotations on the x-ray images. For example, the data source `ricord` (RSNA International COVID-19 Open Radiology Database) provides only positive examples in both the training and the test set. The images provided by `ricord` contains medical annotations on the top area. So, cropping the 8% top pixels of images is necessary in order to reduce unwanted biases.

The proposed models achieve good performance on the CXR-2 Dataset. In particular, ResNet50 and DenseNet121 are the best models for the task, in line with the state of the art shown in [14]. Furthermore, we discovered that histogram equalization is useful for this dataset. We saw that the majority of the errors the models did, were caused by either poor quality of the CXR images or by the presence of medical devices.

Future works include experimenting with a broader range of hyperparameters and both data augmentation and preprocessing algorithms. Particularly, the use of more sophisticated preprocessing algorithms can be relevant in cases of low quality CXR images, as showed in the previous section. Moreover, several explainability techniques for deep learning can be introduced in order to visualize the predictions, hence resulting the models in being more reliable and consistent with their predictions (especially in a very important domain like medical imaging).

References

[1] E. Sogancioglu, E. Çalli, B. Ginneken, K. G. V. Leeuwen, and K. Murphy, "Deep learning for chest x-ray analysis: A survey," *ArXiv*, vol. abs/2103.08700, 2021.

[2] J. P. Cohen, L. Dao, P. Morrison, K. Roth, Y. Bengio, B. Shen, A. Abbasi, M. Hoshmand-Kochi, M. Ghassemi, H. Li, and T. Q. Duong, "Predicting covid-19 pneumonia severity on chest x-ray with deep learning," 2020.

[3] M. Li, N. Arun, M. Gidwani, K. Chang, F. Deng, B. Little, D. Mendoza, M. Lang, O. Vtc Lee, A. O'Shea, A. Parakh, P. Singh, and J. Kalpathy-Cramer, "Automated assessment and tracking of covid-19 pulmonary disease severity on chest radiographs using convolutional siamese neural networks," *Radiology: Artificial Intelligence*, vol. 2, p. e200079, 07 2020.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[8] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[9] I. Sirazitdinov, M. Kholiavchenko, R. Kuleev, and B. Ibragimov, "Data augmentation for chest pathologies classification," *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pp. 1216–1219, 2019.

[10] M. S. Vidya, V. Manikanda Krishnan, G. Anirudh, K. Srinivasa Rao, and J. Vijayananda, "Local and global transformations to improve learning of medical images applied to chest radiographs ," in *Medical Imaging 2019: Image Processing*, E. D. Angelini and B. A. Landman, Eds., vol. 10949, International Society for Optics and Photonics. SPIE, 2019, pp. 813 – 821.

[11] O. Gozes and H. Greenspan, "Deep feature learning from a hospital-scale chest x-ray dataset with application to TB detection on a small-scale dataset," in *41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2019, Berlin, Germany, July 23-27, 2019*. IEEE, 2019, pp. 4076–4079. [Online]. Available: https://doi.org/10.1109/EMBC.2019.8856729

[12] I. M. Baltruschat, H. Nickisch, M. Grass, T. Knopp, and A. Saalbach, "Comparison of deep learning approaches for multi-label chest x-ray classification," *Scientific Reports*, vol. 9, 2019.

[13] H. Liu, L. Wang, Y. Nan, F. Jin, Q. Wang, and J. Pu, "Sdfn: Segmentation-based deep fusion network for thoracic disease classification in chest x-ray images," *Computerized Medical Imaging and Graphics*, vol. 75, p. 66–73, Jul 2019. [Online]. Available: http://dx.doi.org/10.1016/j.compmedimag.2019.05.005

[14] M. Pavlova, N. Terhljan, A. G. Chung, A. Zhao, S. Surana, H. Aboutalebi, H. Gunraj, A. Sabri, A. Alaref, and A. Wong, "Covid-net cxr-2: An enhanced deep convolutional neural network design for detection of covid-19 cases from chest x-ray images," 2021.

[15] M. Raghu, C. Zhang, J. M. Kleinberg, and S. Bengio, "Transfusion: Understanding transfer learning with applications to medical imaging," *CoRR*, vol. abs/1902.07208, 2019. [Online]. Available: http://arxiv.org/abs/1902.07208

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 448–456. [Online]. Available: http://proceedings.mlr.press/v37/ioffe15.html

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[18] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *ArXiv*, vol. abs/1712.04621, 2017.

[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.