Lung disease prediction from X-ray images Project Report

Anna Badalyan[†], Mehran Farajinegarestan[‡]

***Abstract*—The outbreak of the COVID-19 pandemic has brought to the forefront the critical need for accurately and timely identifying the COVID-19 cases. Numerous solutions have been proposed for the task of detecting COVID and non-COVID pneumonia from X-ray images; however a comprehensive evaluation of these solutions have not yet been conducted. The aim of this study is to compare the performance of machine and deep learning algorithms in correctly classifying chest X-ray images of patients into three categories: COVID-19, non-COVID pneumonia, and normal health conditions. We examined the performance of the following algorithms: Logistic Regression, CNN-2, CNN-5, DenseNet-121, DenseNet-201, DenseNet with Attention, RepVGG and ResNet. The results showed that the best performing architecture on the task is the DenseNet, while ResNet was the most efficient from computational and memory point of view. Thus, we suggest that out implementation of ResNet can be used in situations where the memory constraints don't allow for larger models like DenseNet. In addition, we investigated the biasness of the trained algorithms by analysing their performance on datasets obtained using 4 various pre-processing techniques.**

***Index Terms*—Lung disease detection, Neural Networks, Convolutional Neural Networks, DenseNet, ResNet, image classification.**

I. Introduction

The current project is aimed at identifying the most suitable neural network architectures for solving image classification problem. The problem at hand is classifying X-ray images of patients with COVID-19, non-COVID pneumonia, and normal medical conditions.

Since the start of the pandemic the accurate identification of COVID-19 cases has become an acute problem for the research community. Attempts have been made by Wang [1], Cohen [2] and Hong [3] to introduce various deep learning solutions for solving the COVID-19 detection problem. However, to the best of our knowledge no attempts have been made to compare the existing machine and deep learning solution. Moreover, the existing papers compare the results achieved by different authors on different datasets. This work is aimed at comparing the performance of various machine learning models on the same balanced dataset.

To approach the classification task we used the following machine learning algorithms: Logistic Regression, neural networks with 2 and 5 convolution layers as well as popular architectures for image classification such as DenseNet, ResNet and RepVGG. The attention mechanism was used to enhance the performance of DenseNet. We investigated the fairness of the models by undertaking 3 different pre-processing techniques.

The present report is structured as follows. Section II describes the existing attempts to solve the lung disease prediction problem. Section III presents the general processing pipline of our work. Section IV provides the in detail description of the dataset and the pre-processing techniques used. Section V presents the details about the algorithms and neural network architectures used. Section VI illustrates the main results of the experiments and Section VII presents the concluding remarks.

II. Related Work

Wang (2020) [1] proposed a COVID-Net, deep convolutional neural network architecture which was specifically designed to detect COVID-19 cases in radiographic chest images. The authors contributed to research community by making the COVID-Net open source and providing an open access to the benchmark dataset consisting of 13,975 X-ray images. The proposed architecture outperforms other popular architectures for image classification such as ResNet-50 and VGG-19 and achieves the accuracy of 93.3

Cohen (2020) [2] used DenseNet architecture to identify the severity of COVID-19 disease from lung images. The authors used non-COVID-19 datasets of chest images to train the feature extraction layers. The relatively small COVID dataset was then used to classify the severity of the disease.

Liu (2019) [4] addressed the problem of identifying and localizing the chest diseases in X-ray images by introducing a Contrast Induced Attention Network (CIA-Net). The network contains an attention module that focuses on abnormalities and an alignment module which adjusts geometric deformations. The model was trained to classify 14 chest diseases and improved the AUC score from 0.80 to 0.83.

Hong (2021) [3] present a model called MGMADS-CNN which applies a multi-head attention mechanism and depthwise separable convolution for COVID detection. The network with 3 attention layers achieved 96.6% accuracy on X-ray images which outperforms popular CNNs such as LeNet-5, AlexNet, GoogLeNet, ResNet, VGGNet-16. The proposed solution is also fast at making predictions and lightweight with only 43.6 Mbytes.

Overall, there have been many attempts by the research community to solve the problem of lung disease detection from X-ray images. It is however, difficult to compare these results and identify the best architecture for the problem as the researchers used different techniques and datasets for the

[†]Department of Mathematics, University of Padova, email: {anna.badalyan}@studenti.unipd.it

[‡]Department of Mathematics, University of Padova, email: {mehran.farajinegarestan}@studenti.unipd.it

problem. While some researches used pretrained model others trained the models from scratch. This paper aims at bridging this gap by testing popular neural network architectures on lung disease prediction task and identifying the most suitable one.

III. Processing Pipeline

In this work, we present a technical approach to analyze chest X-ray images in order to categorize a patient's medical condition as either normal, COVID-19, or non-COVID pneumonia. Our approach consists of a pre-processing pipeline that resizes the images and adjusts their shape to prepare them for analysis.

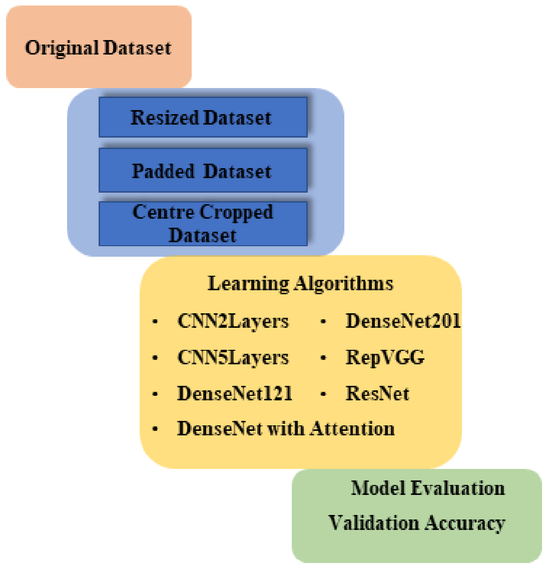


Fig. 1: High Level Description of Processing Pipeline

Figure 1 illustrates our processing pipeline. We utilized three different pre-processing strategies, including resizing the image to a square format, padding rectangular images to achieve a square shape, and center cropping rectangular images. After this step, all the images were resized to 224x224.

The pre-processed images were then used to train various models. Training and validation accuracy of each model as well as their training time were recorded. Finally, the best model was selected based on its validation accuracy and evaluated on the test set.

We tried a machine learning approach by building Logistic Regression and then switched to deep learning solutions starting with shallow networks moving to more advanced deep neural network architectures. The performance of the implemented algorithms on datasets processed in different ways was used to choose the best pre-processing technique. The attempts were made to improve the accuracy of the best performing model by applying the attention mechanism.

The algorithms were trained on Kaggle platform. Logistic Regression model was trained on Intel Xeon 2.20 GHz CPU with 30GB RAM and neural networks were trained with NVIDIA Tesla P100 GPU accelerator with 16GB of memory.

IV. Signals and Features

The dataset contains 4575 posteroanterior chest X-ray images belonging to 3 classes: covid, normal and pneumonia. The dataset is balanced with 1525 images per class. Figure 2 shows the sample images for each class. The authors of the dataset report that 912 images of COVID cases were takem from an already augmented dataset [5].

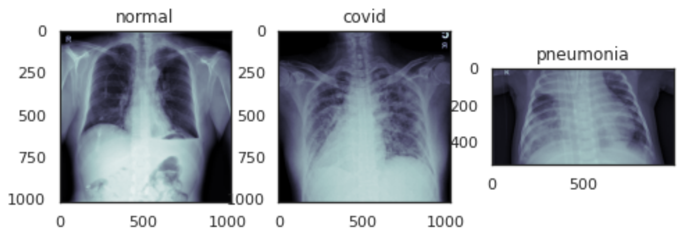


Fig. 2: Sample images by label

The main problem with the dataset are the image sizes and their width to height ratio. The average width to height ratio for pneumonia images is 1.45 compared to 1.04 for COVID and 1.0 for normal images as can be seen in figure 3. Another issue with the dataset is that some of the COVID images are rotated or flipped which may make it harder to for the neural network to learn the relevant dependencies.

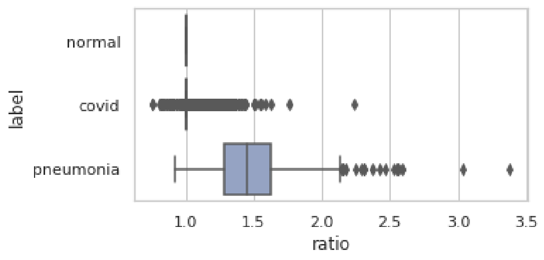


Fig. 3: Average width to height ratio by label

To solve the issue with the image aspect ratio, the following pre-processing techniques were applied (Figure 4):

1) Resizing the image to the squared format. The disadvantage of this technique is that wide images get elongated and so instead of learning the disease representation, the models will associate a long rib cage with pneumonia.
2) Padding the rectangular images to achieve the squared shape. This technique preserves the original ratio of the image, however, similarly to the first case, bias could be introduced to the model with padded regions.
3) Center cropping the rectangular images. With this technique the important regions could be cropped.

After the first pre-processing step, all images were resized to 224x224 and saved as one channel gray scale images.

In addition to the above mentioned pre-processing techniques we also tried to augment the dataset. The augmentation techniques include random shifting of the height and width of the images, zooming and horizontal flip. These transformation were randomly applied to the dataset with padded images.

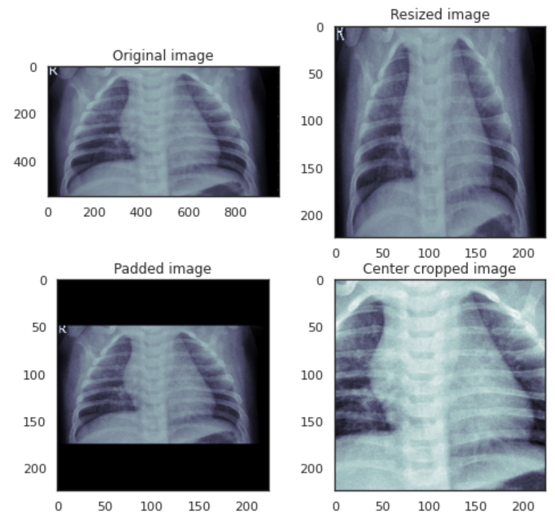


Fig. 4: Pre-processing techniques applied to a wide image

The dataset was split into train and test in proportions 80% and 20% respectively. The train dataset was then split into train (80%) and validation (20%) using ImageDataGenerator from Keras library.

V. Learning Framework

The following algorithms were applied to solve the classification problem:

Logistic Regression. The Logistic Regression algorithms from `sklearn` library was used for the image classification problem. The images were flattened to a vector of length 50176. The type of penalty ($l1$ and $l2$) as well as regularization parameter C were chosen on the validation set. Then the models were retrained using the train and validation sets.

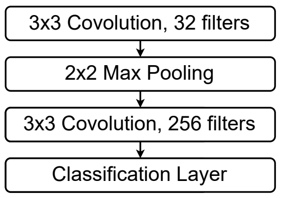
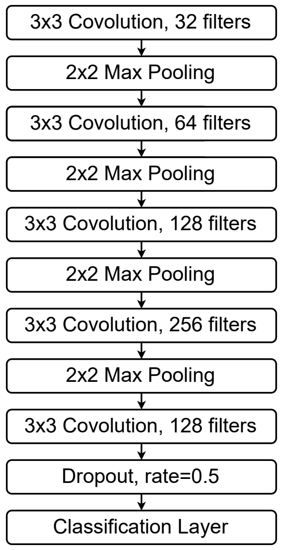


Fig. 5: CNN-2 Architecture

Shallow CNN. We tested 2 shallow CNN architectures:

1) CNN with 2 layers (CNN-2). The input is the image of size (224x224x1) followed by a convolution layer with 32 filters and ReLU activation. Then max pooling operation is applied followed by another convolution layer with 256 filters and ReLU activation. The output is then flattened and passed to the dense layer with softmax activation.
2) CNN with 5 layers (CNN-5). The input image of size (224x224x1) is passed to 5 convolution layers with ReLU activation and 32, 64, 128, 256 and 256 filters respectively. Max pooling operation is used between the convolution layers. After the output is flattened, dropout layer is used in order to prevent over-fitting.


3x3 Covolution, 32 filters
2x2 Max Pooling
3x3 Covolution, 64 filters
2x2 Max Pooling
3x3 Covolution, 128 filters
2x2 Max Pooling
3x3 Covolution, 256 filters
2x2 Max Pooling
3x3 Covolution, 128 filters
Dropout, rate=0.5
Classification Layer
Fig. 6: CNN-5 Architecture

ResNet. Residual Network [6] is a type of Convolutional Neural Network (CNN) that is designed to overcome the vanishing gradient problem in deep neural networks. ResNet architecture consists of multiple building blocks, called residual blocks, which contain multiple convolutional layers and are designed to allow the network to learn residual representations of the input data. It has become one of the most widely used deep learning models for computer vision tasks. The success of ResNet has inspired further developments in deep learning and has led to the creation of deeper and more sophisticated models that continue to advance the field of computer vision.

Our implementation has residual blocks containing convolutional layers with a 3x3 kernel followed by batch normalization and relu activation repeated two times, after the third convolution and batch normalization the input of the residual block is added and after a relu activation is applied, the output of the residual block is passed to the next block. Similar to the original ResNet implementation, out ResNet model design has a convolution layer with a 7x7 kernel and stride of 3 followed by batch normalization, relu activation and maxpooling of size 3x3. However to avoid overfitting it has only 4 residual block with 128 and 256 channels. Average pooling and a fully connected layer are the last layers of the model. **RepVGG**. Multi-branch topologies e.g., ResNet, are suitable at training time since they avoid gradient vanishing problem, on the other hand plain architectures like VGG [7] have less computational cost and high inference time for predictions. RepVGG [8] is designed to exploit the advantages of both strategies. This model has a multi-branch topology at the training time and a VGG-like inference-time architecture which is nothing but a stack of 3x3 convolution and ReLU layers. This decoupling of the training-time and inference-time architecture is done by structural re-parameterization, hence it is called RepVGG.

The model used in this work consists of 7 blocks. Each block at the training-time has 3 branches. First branch has a 3x3 convolution, The second one has a 1x1 convolution and the third branch contains the input (skip-connection) called identity branch. The identity branch can be viewed as a 1x1 convolution with an identity matrix as the kernel. After all the convolutions a batch normalization is applied. The final output is the summation of all three branches. In order for the outputs have the same dimensions padding is applied to the branch with 3x3 convolution. After training each block is converted into a single 3x3 convolution layer for inference.

The weights of the new convolution layer are calculated according the original paper which makes the output of the inference-time model exactly the same as output of the training-time model.

To the best of our knowledge this project was the only Tensorflow implementation of RepVGG. At inference-time the number of trainable parameters were 702,147 and 0 non-trainable parameters since the models doesn't have any batch normalization layer anymore. This means more than 10% reduction of the number of parameters after the architecture of the model is changed for inference. This makes the model suitable for real-world applications in both academia and industry.

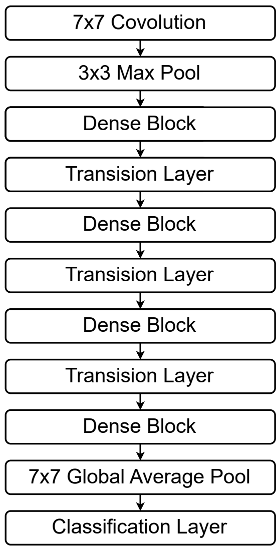


Fig. 7: DenseNet architecture

DenseNet. Introduced by [9] the DenseNet architecture exploits dense connections between each of the convolution layers in a network in feed-forward fashion. We have implemented the DenseNet-121 and DenseNet-201 architectures proposed by [9].

Our implementation controls the size of the model by setting the parameter the growth rate k (filters) and the number of repetitions inside the block as well as the number of blocks. The architecture consists of the following building blocks:

- *The convolutional block* consists of batch normalization, ReLU activation and 3×3 convolution layer. Although other popular deep CNN architectures apply batch normalization after convolution, the reverse design proved to work better for DenseNet.
- *The dense block* contains 2 convolutional blocks with $4 \cdot k$ and k filters sized 1×1 and 3×3 respectively. The 2 convolutional blocks are repeated for a specified number of iterations and the outputs at each iteration are concatenated. This structure makes all the layers within the dense block connected to each other.
- *The transition block* is applied between the dense blocks. It consists of a 1×1 convolution layer with the number of filters equal to the half of the number of channels and a 2×2 average pooling with stride 2.

The blocks are then stacked in the following manner: 7×7 convolution with stirde 2, 3×3 max pooling with stride 2, dense blocks and transitions layers between them, 7×7 global average pooling and a dense classifier with softmax activation function.

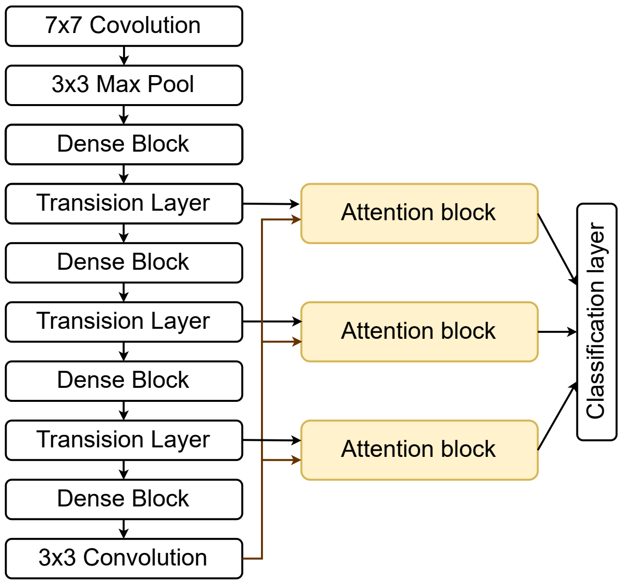


Fig. 8: DenseNet with Attention architecture

Attention mechanism. We applied the attention mechanism proposed by [10] which is based on the idea of compatibility between the outputs of intermediate layers (l) and the final output (g). The attention block takes the tensors l and g as input and computes a compatibility score. Authors propose the dot product between g and l to compute the compatibility score. In our implementation, the convolution later was applied to the $g + l$. The score was then normalized using softmax operation. The normalized score was then multiplied element wise with with the layer representation l. The result then replaces g as a global representation for the image.

The described attention mechanism was them applied to the DenseNet model. The attention block was applied to the first three outputs of the dense blocks. To get the global image representations the transition layer was applied to the output of the last dense block and 3×3 convolution layer with 512 filters was applied to bring the images to the necessary shape.

Training procedure. We used Adam optimizer in order to train the neural networks. The learning rate was adjusted to the size of the model: CNN with 5 layers was trained with a learning rate of 1e-3, CNN with 2 layers and RepvGG with a learning rate of 1e-4, ResNet and DenseNet with 1e-5 and 1e-6 respectively.

Early stopping technique with patience equal to 3 was adopted as a termination criterion. For deep models that are not expected to start learning during the first 20 epochs a starting point for the patience was set to 20.

VI. Results

pre-processing technique. We tested 3 different pre-processing techniques to bring the X-ray images to the same size namely resizing, padding and center cropping. Logistic Regression, CNN-2, CNN-5, DenseNet-121 and DenseNet-201 models were trained with all three datasets. The accuracy on the validation set was used to determine the best pre-processing technique.

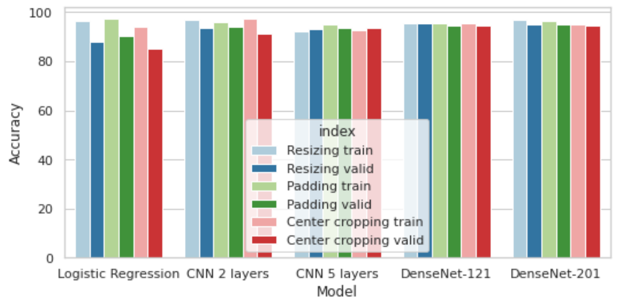


Fig. 9: Train and validation accuracy with 3 different types of pre-processing on CNN-2, CNN-5 and DenseNet-121.

The neural networks performed similarly well with resizing and padding pre-processing techniques. While CNN-2 performed slightly better on a padded dataset than on a resized one (93.99% to 93.58%), the DenseNet-121 performed in the opposite way (94.81% to 95.49%). On the other hand, Logistic regression showed better performance on the padded dataset with 90.13% compared to only 87.98% on resized dataset.

All models performed worse on the center cropped dataset except the CNN-5 model. The difference is particularly high for shallow models such as Logistic Regression and CNN-2.

Performance drop on the center crop dataset can be explained by the fact that some useful features were cut during cropping. Another possible explanation of the phenomenon is that given that the wide images mostly belonged to the pneumonia class, by adding padding or resizing them we introduced biased to the dataset and the model was detecting the black edges instead of the actual disease. This can also be explained by the fact that it was easier to identify padding than resizing for Logistic Regression, which led to better performance on the padded dataset; while for the convolution operation it was also able to identify the stretched images.

The experiments on data augmentation were carried out with the CNN-2 and DenseNet-121 architectures. Both of them showed significant drops in validation accuracy compared to the dataset without augmentation (the padded dataset)

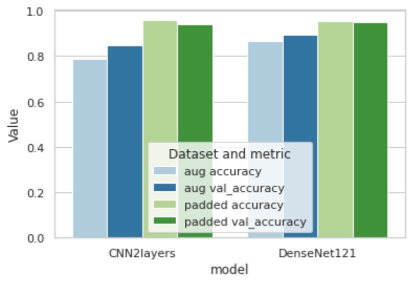


Fig. 10: Train and validation accuracy on augmented and padded data on CNN-2 and DenseNet-121.

as illustrated in Figure 10. The validation accuracy for the CNN-2 dropped from 93.98% to 84.84% and the accuracy for the DenseNet-121 dropped from 95.34% to 86.65%. Given that the original COVID images were also augmented, such a drop in performance with the augmentation techniques may prove the hypothesis of the original biasness of the dataset.

As the resizing and padding pre-processing techniques showed better performance on validation set, we used them for the classification task.

Classification performance The classification accuracy of 7 neural network architectures was compared on the validation set. As it can be seen from (fig), all models performed well, surpassing the accuracy of 93% on validation set. Figure 11 shows the loss curves for 4 main architecture types that were trained on the padded dataset. We can see that the shallow network (CNN-2) trains fast, reaches the lowest value of the loss function in less than 10 epochs and starts overfitting quickly. On the other hand, deeper networks learn poorly during the first 10 epochs, but later they converge to better results than shallow ones.

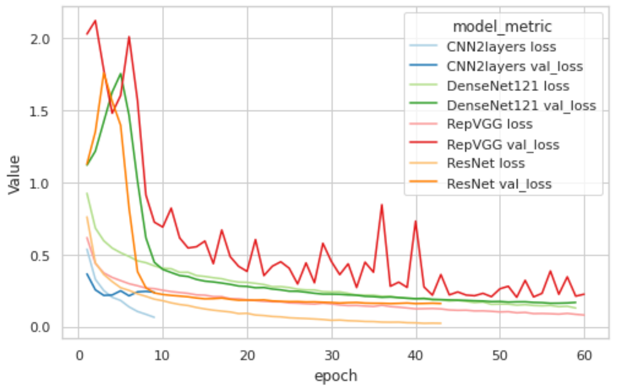


Fig. 11: Training and validation loss curves for the CNN-2, DenseNet-121, RepVGG and ResNet.

RepVGG shows fluctuating behaviour of the loss on the validation set during training. This could be explained by the fact that our implementation didn't use any dropout layer between convolution layers as proposed in the original paper.

DenseNet architecture exhibits a smaller gap between the training and validation losses during the whole training process than other trained models. Attention mechanism applied to the DenseNet architecture didn't improve the performance with the validation accuracy dropping from 95.49% to 93.17%. As illustrated in Figure 12, the training of the model with attention took more epochs that DenseNet architectures without attention. This may mean that the output vector of the original DenseNet architecture already contains useful information from the previous dense blocks, that's why applying the attention on the intermediate outputs didn't produce good results.

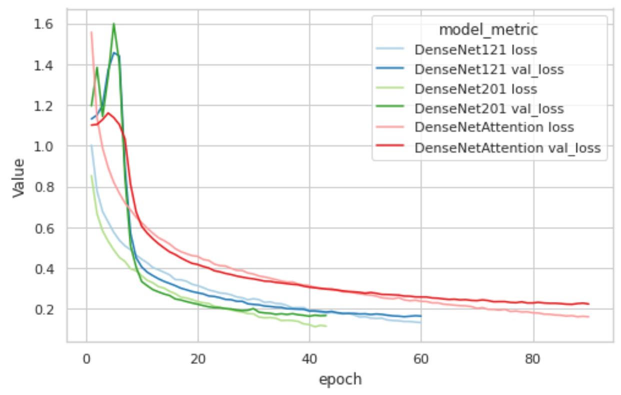


Fig. 12: Training and validation loss curves DenseNet-121, DenseNet-201 and DenseNet with attention on resized dataset.

DenseNet-121 trained on resized images showed the best validation performance (95.49% accuracy). The test accuracy was equal to 94.2%. Figure 13 shows the ground truth labels and the labels predicted by the DenseNet-121. We can see that the model is mostly confusing pneumonia and normal cases. The model is better at identifying COVID images with the highest F1 score for this class.

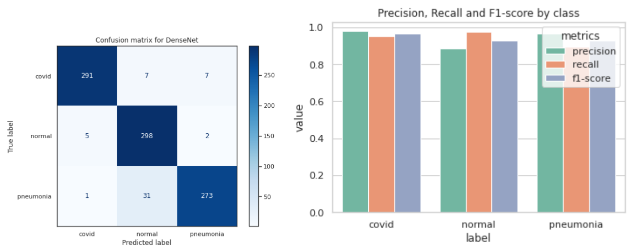


Fig. 13: Confusion Matrix, Precision, Recall and F1-score by label produced by DenseNet-121 on the test set.

Computational efficiency. Figure 14 illustrates the computational efficiency of the implemented models. ResNet is the most efficient architecture if the accuracy, training time and memory requirement are taken into consideration. Although it's the most lightweight model with the smallest number of parameters (628,227), it outperforms shallow networks CNN-2 and CNN-5 as well as deeper models like RepVGG and DenseNet with Attention. DenseNet architectures perform better that others, but this comes with the high computational costs, memory requirement and long training time.

The efficiency of ResNets can be explained by the fact that, the idea behind them is to allow for the flow of information through the network. This helps in mitigating the vanishing gradient problem that is commonly encountered in neural networks, and also faster convergence during training. The residual connections in ResNets allow the network to effectively backpropagate gradients, leading to faster convergence during training. Hence these models compared to others achieve high accuracy with fewer number of parameters.

RepVGG is the smallest model trained with a size of 7.56 Mb. Although it has more parameters than ResNet and same residual connections during training but it failed to achieve higher accuracy than other models specially ResNet. However, The advantage of RepVGG is that this model is capable of reducing its size and number of parameters of 10%, Which is suitable when the memory capacity and computational cost is limited. But when the model is not too deep the advantage of this model is not considerable. Also the based on our experiments the validation accuracy was fluctuating.

VII. Concluding Remarks

We have analysed the performance of the 8 machine learning algorithms on the given image classification task. DenseNet type models perform better in terms of accuracy, but require more time to train and memory. ResNet and RepVGG have more than 10 times less parameters than DenseNet and therefore, are faster to train and more memory efficient. Shallow CNN with 2 convolutional layers also performs well on the task reaching the accuracy comparable to deeper models.

Such a similar performance of shallow and deep networks might have been caused by the biasness introduced in the dataset as images labeled as "pneumonia" were on average wider than those labeled as "covid" and "normal". This hypothesis was also proved by the fact that data augmentation techniques where "normal" and "covid" images also randomly changed dimensions showed significantly inferior performance compared to the dataset without augmentation.

Standard model architectures, such as DenseNet and ResNet, demonstrate superior performance compared to our experimental models, such as DenseNet with Attention and RepVGG.

References

[1] L. Wang, Z. Q. Lin, and A. Wong, "Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images," *Scientific reports*, vol. 10, no. 1, pp. 1–12, 2020.

[2] J. P. Cohen, L. Dao, K. Roth, P. Morrison, Y. Bengio, A. F. Abbasi, B. Shen, H. K. Mahsa, M. Ghassemi, H. Li, *et al.*, "Predicting covid-19 pneumonia severity on chest x-ray with deep learning," *Cureus*, vol. 12, no. 7, 2020.

[3] G. Hong, X. Chen, J. Chen, M. Zhang, Y. Ren, and X. Zhang, "A multi-scale gated multi-head attention depthwise separable cnn model for recognizing covid-19," *Scientific Reports*, vol. 11, no. 1, pp. 1–13, 2021.

[4] J. Liu, G. Zhao, Y. Fei, M. Zhang, Y. Wang, and Y. Yu, "Align, attend and locate: Chest x-ray diagnosis via contrast induced attention network with limited supervision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10632–10641, 2019.

Architecture	Total parameters	Epochs to train	Time to train (sec)	Size	Validation accuracy
CNN-2	9,198,915	8	87	110.4Mb	0.9358
CNN-5	1,054,723	8	53.3	12.7Mb	0.9399
DenseNet-121	7,044,547	77	1614	88.2Mb	0.9549
DenseNet-201	18,338,499	70	2265	222.1Mb	0.9508
DenseNet with Attention	11,701,571	71	1625	142.17Mb	0.9317
RepVGG	786,691	60	366	7.56Mb	0.9347
ResNet	628,227	31	234	7.82Mb	0.9481

Fig. 14: Computational efficiency of the trained neural network architectures.

[5] A. Asraf, "Covid19, pneumonia and normal chest x-ray pa dataset," *Mendeley Data*, vol. 1, 2021.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[8] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13733–13742, 2021.

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[10] S. Jetley, N. A. Lord, N. Lee, and P. H. Torr, "Learn to pay attention," *arXiv preprint arXiv:1804.02391*, 2018.