Exploration on Data Augmentation and Learning Rate Optimization on Detection of COVID-19 via Chest X-ray Images

Xi (Cassie) Guo
Team 1122
https://youtu.be/mbksdIBT4iw
https://github.com/lilsummer/CS598_DL
xig2@illinois.edu

ABSTRACT

The global pandemic of COVID-19 has lead to more than 137 million cases and 2.97 million deaths worldwide. The viral RNA RT-PCR used for screening COVID-19 is highly sensitive but requires a short sampling window and the test can be costly and time-consuming. The chest X-ray images (CXR) can provide a faster and more cost-effective way of diagnosis. There has been a considerable number of research in deep learning focusing on building image classifiers to detect COVID-19 and other types of lung diseases. We presented the performance of using several reported transfer learning methods to build a four-class classifier. We provided a comprehensive comparison on data augmentation and learning rate optimization. The performance of models were evaluated using sensitivity, specificity, PR-AUC score, precision-recall curve and training time. We discovered that the model performance was improved due to increased data size compared with the model reported one year ago. The data augmentation technique used in this study was not very effective in increasing sensitivity and might have harmed specificity. The best sensitivity score (0.9784) was through DenseNet121 with cropping augmentation and the best specificity (0.9969) was from baseline DenseNet121. Furthermore, we implemented one cycle learning policy and reduced learning rate policy on ResNet50, resulting in better performance than training with constant learning rate within shorter training time. Overall, this study explored the state-of-the-art transfer learning approaches and investigated a collection of methods to improve model performance.

Author Keywords

COVID-19, image classification, transfer learning, deep learning, learning rate optimization

1. INTRODUCTION

The global pandemic of COVID-19 has caused 125 millions of cases and 2.75 million deaths worldwide [1]. It has led to long-standing impact in the practice of public health, including disease screening, contact tracing, policy making, and vaccine development [2]. The screening test of COVID-19 via RT-PCR is massively adopted in the US and many other countries. The test is highly sensitive but the procedure can be time-consuming, costly and sometimes inefficient [3, 4]. The success of RT-PCR test depends on the sufficient RNA sampled from a swab. Depending on the sampling time point, false negative could occur. Chest X-ray (CXR) is another way of diagnosis and can be cost-effective for screening. Additionally, it does not depend on the sampling time as well.

Recent research in deep-learning based COVID-19 classification on CXR images have adopted three major approaches, including:

1. Convolutional neural networks built from scratch: COVID Net is a deep learning based image classifier built on 14,000 x-ray images [4]. It utilizes projection-extension (PEPX) design pattern. The model compromises: 1) First-stage projection; 2) Expansion; 3) Depth-wise representation; 4) Second-stage projection; 5) Extension.

2. Transfer learning based models: Minaee et al have reported the performance of transfer learning models on binary classification of CXR images [5]. The study used pretrained model as a feature extractor, and trained classifier on top of the pretrained model.

3. Ensemble model: FLANNEL (Focal Loss bAsed Neural Network Ensemble) is an ensemble model, which uses ensemble of five different popular model (ResNet, ResNeXt, DenseNet, Vgg and Inception) as a base learner, and neural weight module as a top module. To address the issue of imbalanced class, it replaced cross-entropy loss with focal loss[6].

Transfer learning is a widely-used technique in computer vision [7]. It enables researchers and machine learning practitioners to be able to prototype a robust image classifier without ramping up lots of computational resources to train on a large dataset [8]. There are three main advantages of transfer learning when compared with building deep learning model from scratch. First and foremost, when the dataset is small, transfer learning eliminates the need to collect a large training dataset. Secondly, since the dataset such as imageNet has been quality-controlled [9], it ensures the label of the training data. Last but not least, since we can load the weights directly from the pretrained model, transfer learning makes the training more efficient [5].

Data augmentation is a set of techniques to increase the quality and quantity of image dataset, thus to improve the performance of deep learning model [10]. Broadly speaking, these techniques can be categorized into two types: basic data augmentation and deep-learning based data augmentation. Basic data augmentation encompasses a variety of commonly used techniques, such as flipping, rotation, and color transformation. Deep learning based data augmentation is a set of deep learning models that help to improve the representation of images. For instance, generative adversarial network (GAN) has been applied to generate fake images so that more similar features of the original dataset can be retained [10].

Learning rate policy, such as 1cycle learning policy [11], has achieved good performance on various deep learning models. 1Cycle policy, or one cycle policy was developed by L. Smith in 2018 [11]. It includes two components: step size and the minimum maximum boundaries. This gives the cyclic momentum to learning rate explorer in the model. According to a study by S. Gugger [12], using 1cycle learning policy to replicate a resNet56 model on Cifar10 dataset can achieve same or even better precision, with only 1/6 of the original epochs. The idea is to start to train the model with a small learning rate; to slowly increase the learning rate until it reaches to a maximum; then to decrease the learning rate to minimum or to a lower rate. To find the maximum learning rate, the author suggested to train the model by a very small learning rate and pick a value before training loss reaches the minimum on the loss curve. The dynamic of 1cycle learning rate is shown in Figure 1 below.

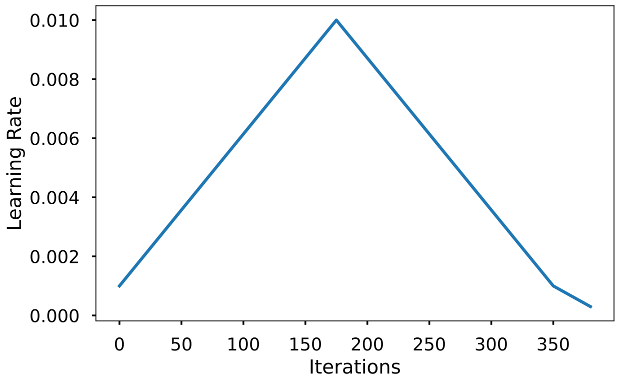


Figure 1: An example of learning rate in one cycle policy. Adopted from S. Gugger [12]

2. APPROACHES AND METRICS

2.1 Data Source

The dataset (Set 1) is summarized as below in Table 1. To split the dataset, at the time of training, 80% of images from each class were randomly selected to construct the training data; the rest of images were used for testing. Several studies pointed out that to prevent data leakage, images from the same patient cannot both exist in training and testing directories [6]. We examined the patient information in these datasets, as described below. Based on our understanding, the prepprossessed final dataset does not have any data leakage since each image represents each unique patient.

Table 1: Summary of X-ray image dataset in Set 1

	COVID-19	Bacterial	Viral	Normal
Train	2892	2224 (reduced to 1149)	1076	8153
Test	724	556 (redcued to 288)	269	2039
Total	3616	2780 (reduced to 1437)	1345	10192

We have considered and used two main data sources as described below:

Table 2: Summary of X-ray image dataset in Set 2

	COVID-19	Bacterial	Viral	Normal
Train	1437 (augmented to 3000)	569	539	4081
Valid	371	149	133	1015
Test	1808	719	673	5096
Total	3616 (augmented to 5179)	1437	1345	10192

1. Kaggle Radiography Database: This dataset contains images from four classes (COVID: 3616 images; viral-pneumonia: 1345 images; normal: 10192 images; lung opacity: 6012 images). This dataset combines the following data source:
   - Italian Society of Medical and Interventional Radiology (SIRM) [13]
   - Valencia Region Image Bank (BIMCV) padchest dataset [14]
   - Dataset maintained by J.P. Cohen et al [15]
   - Kaggle and Radiological Society of North America (RSNA) [16]

   Because this dataset is relatively comprehensive and contains many popular datasets used by other studies (For instance, the COVID dataset maintained by J.P. Cohen [15] has been used in several studies), we decided to use this as the main source of images. There was no metadata related to patient ID in the dataset as well as the associated papers [17, 18], as described in this post[1]. Hence we assumed each image is unique for each patient.

2. Bacterial pneumonia dataset by P. Mooney [19]. This dataset is from retrospective cohorts of pediatric patients (1-5 years old) from Guangzhou Women and Children's Medical Center. Therefore, all images were from children. This created some unexpected differences between this dataset and the Kaggle Radiography Database, because there are anatomy differences between adult lung x-ray images and children's. Therefore, the models trained in the present study may have not learned features directly related bacterial pneumonia, but features related to the anatomy differences between adults and kids. However, we didn't find any other available dataset on bacterial pneumonia from adult patients. In this dataset, the image title such as `person1000_bacteria_2931.jpeg` is associated with the patient ID. To create the dataset from the initial 2780 bacterial pneumonia CXR images, we randomly selected one image per patient, resulting in a total of 1437 images. We also used `cv2` library to transform `jpeg` to `png` format.

2.2 Model Training

We used these pre-trained models as feature extractors and trained four-class classifiers on top of them. The classes include: *normal, COVID-19, non-COVID viral pneumonia and bacterial pneumonia.* The models include

- ResNet50
- SqueezeNet
- DenseNet121

[1]https://www.kaggle.com/tawsifurrahman/covid19-radiography-database/discussion/230951

The reason for choosing this method instead of training the entire network was that the dataset was still relatively small. Additionally, using pretrained model requires less training time and less computing power than training the entire model.

2.3 Metrics

Because the number of images from each class were imbalanced, we chose to use precision, recall instead of accuracy as metrics. When classes are imbalanced, reporting accuracy can be misleading. We also followed the Deep-COVID paper and used sensitivity (also known as recall) and specificity (also known as positive predictive value) for the COVID-19 CXR image class.

$$\text{Sensitivity} = \frac{\#\text{Images correctly predicted as COVID-19}}{\#\text{Total COVID-19 images}}$$

$$\text{Specificity} = \frac{\#\text{Images correctly predicted as non-COVID-19}}{\#\text{Total non-COVID-19 images}}$$

Since sensitivity and specificity are mainly focused on correct labels, we also introduced PR-AUC as a metric, which indicates the area under the precision-recall curve.

Practically speaking, the ultimate goal is to accurately identify COVID-19 positive cases as many as possible, hence the cost of a false negative is much higher than the cost of a false positive. Therefore, recall/sensitivity is more important in our case.

2.4 Model Implementation and Computing Infrastructure

The models were implemented in `pytorch` and `python`. All models were trained on Kaggle Notebook (with GPU kernel). Kaggle Notebook has 40 hr quota of free GPU resource on a weekly basis. We estimated we used 20-30hr per week for the entire project. The code and notebooks are in the github repository [2].

2.5 Transfer Learning

2.5.1 *ResNet*

The innovation of ResNets is skip connections, or residual learning. Many problems emerge with deeper networks, including vanishing/exploding gradient, difficulty in optimization and degradation. The researchers of ResNet tried to address these issues by creating skip connections like Figure 2 below. We used ResNet50 for the current study. The implementation was to change the dimension of last fully connected (fc) layer, and add a softmax function on top of it.

2.5.2 *SqueezeNet*

SqueezeNet has a simpler architecture and can be compressed to a lightweight model [21]. The model has 50x fewer parameters but similar level of performance compared to AlexNet. The model architecture is composed of 1) first conv layer; 2) 8 Fire modules; 3) Final conv layer. We used `squeezenet10` in pytorch and changed the output dimension of the convolutional (conv) layer in the model classifier.

2.5.3 *DenseNet*

The motivation for the author of DenseNet is similar to ResNet, which is the gradient vanishing problem for deeper network [22]. The author introduced more connectivity between layers, so that each layer can receive feature maps from other layers in order to reuse more features. To use DenseNet121 as a feature extractor, we followed a Kaggle

[2]https://github.com/lilsummer/CS598_DL

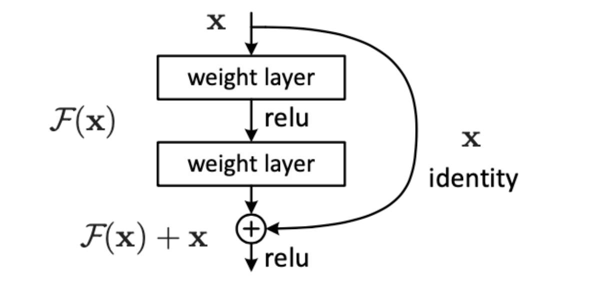

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
Figure 2. Residual learning: a building block.

Figure 2: Residual block diagram, from the study by He et al [20]

Notebook [23], which creates a DenseNet() class by changing the 1) output dimension of the first conv layer; 2) the output dimension of the classifier.

2.6 Data Augmentation

Data augmentation was implemented in Augmentor. Compared with `torchvision.transforms`, Augmentor has several improved functionalities such as rotation, perspective skewing, and shearing[3]. Additionally, Augmentor's `process` and `sample` operations make it easier to purposefully process and sample images, and inspect the entire process. We used two separate directories to inspect the processing of the Augmentor pipeline from the source images to the prepared images for the model training.

2.7 Learning Rate Optimization

We used `torch.optim.lr_scheduler` to adjust the learning rate and Set 2 in Table 2 for training, validation and testing. For Experiment 3, we used the methods described by A. Nain [24] to implement the 1cycle policy. For Experiment 4, we used the basic `torch.optim.lr_scheduler.ReduceLROnPlateau` and `catalyst`[25] to implement the policy.

3. EXPERIMENTAL RESULTS

3.1 Exploration on Data Augmentation

3.1.1 *Experiment 1: rotation*

Table 3: Comparison on performance between baseline and rotation data augmentation on Set 1

	ResNet50		SqueezeNet10	
	Sensitivity	Specificity	Sensitivity	Specificity
Baseline	0.9627	0.8925	0.9641	0.8817
Rotation	0.9171	0.9507	0.8508	0.9353

We first split the dataset into training and testing set by 8:2 ratio as shown in Table 1. In this table, each model was run three times. The training and the testing sets were randomly assigned each time. The learning rate was set to $5e-4$, and all models were trained for 25 epochs. The results were shown in Table 3. For baseline, we used `resize` and `center_crop` to make sure the images have the correct sizes. The reason for using `center_crop` was that we noticed the range of the chest selected for x-ray varied. For rotation, we used `rotate` from Augmentor. The rotation helps to correct the tilted images in the dataset (see Figure 3).

[3]as shown on the github page https://github.com/mdbloice/Augmentor

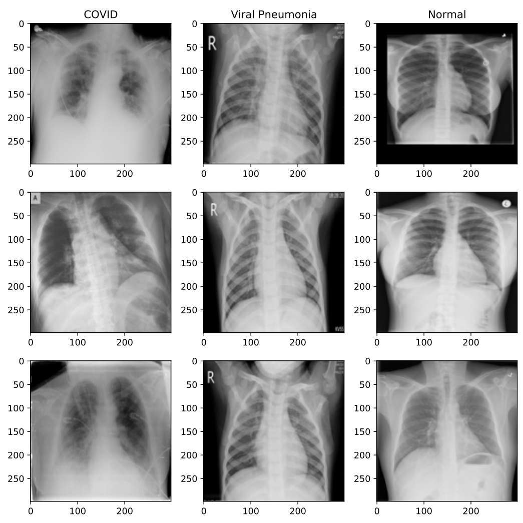


Figure 3: Visualization of Kaggle COVID-19 dataset. The labels of the images are on the top of the columns. Tilted images are in: [0,0], [0,1], [1,0], [2,0].

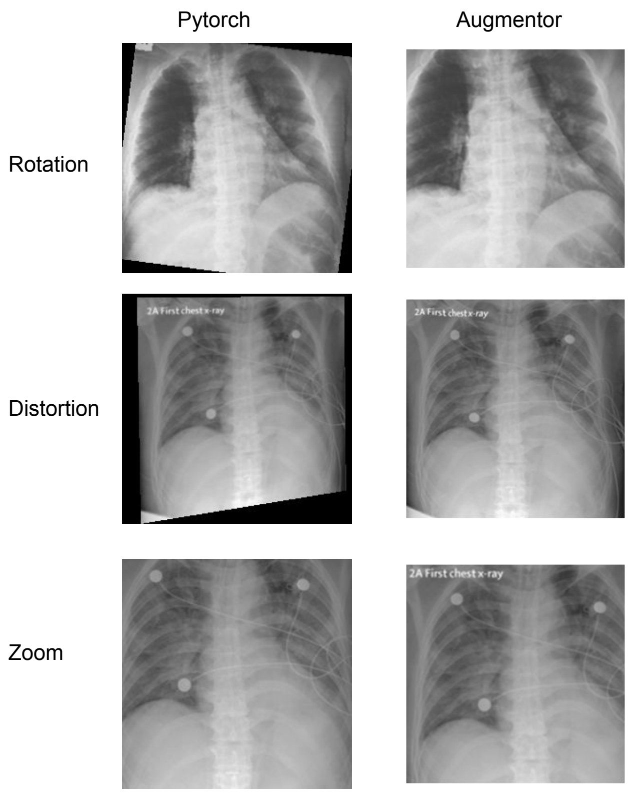


Figure 4: Comparison of augmentation function in Pytorch and in Augmentor. For distortion, the left figure uses random perspective with 0.3 distortion in Pytorch; the right figure uses elastic distortion in Augmentor. For zoom, the left figure uses center crop and the right figure uses zoom in Augmentor.

This rotation function has combined both cropping and rotation, which is an improvement compared to the rotation function in `pytorch` (see Figure 4 for comparison). For rotation in Augmentor, we used `rotate()` with probability of 0.1 and maximum left and right rotation by 15 degrees. Processed data was then loaded using `dataloader` in Pytorch, with a normalizer of `mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]`[4]and a batch size of 32. We observed from Table 3 that:

- Rotation decreased the sensitivity while increased the specificity.
- The validation loss stopped to decrease after 10 epochs. Figure 5 shows the loss curve for ResNet50 model training for 25 epochs. This indicates that the model has not learned the features in the validation dataset. In other words, the model didn't generalize well. The reasons likely to cause such model behavior includes: 1) the training dataset was over-engineered so that it had a large discrepancy compared with the testing set[5]; 2) inefficient training caused by bugs in the code.
- The training time increased beyond linear rate with the increased epochs. To train the model from 10th to 25th epochs, the time spent was at least three times that to train the model from the 1st to 10th epochs.

Given above results, we first checked the augmentation pipeline and discovered that there were some discrepancies between the data augmentation on training and on validation. We resolved the inconsistency in the pipelines.

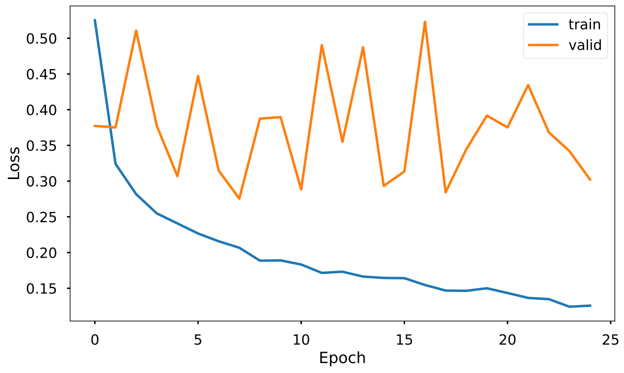


Figure 5: Loss curve for ResNet50, trained on Set 1, lr=5e-4, 25 epochs

3.1.2 Experiment 2: multiple comparisons

In this experiment, we used a different split dataset as shown in Table 2. The test set was selected once and remained unchanged for the entire experiment. To split training and validation dataset, a random seed was set each time the model was trained to ensure the diversity of training set. We chose to use such setup due to the following reasons: 1) DeepCOVID study [5] has used a smaller augmented training set; 2) We saw that the performance of Set 1 in Experiment 1 was sufficient even though the images were not properly processed. Additionally, since the performance stopped to improve after 15-20 epochs, the number of iterations were reduced to 20 epochs for Experiment 2. `Augmentor` were used to increase the training size of COVID-19 image set to 3616. The data augmentation pipeline was designed as below:

[4]This parameter is from ImageNet, see discussion here https://discuss.pytorch.org/t/understanding-transform-normalize/21730

[5]We later discovered that center crop was used in the training data and random crop was used in the validation data. We corrected the mistake for the following experiments.

Table 4: Performance of Experiment 2

	Augmentation Pipeline	Baseline	Baseline + Crop	Baseline + Crop + Distortion	Baseline + Crop + Zoom
ResNet	Senitivity	0.8715	0.9274	0.9303	0.8752
	Specificity	0.9812	0.9219	0.9055	0.9626
	PR-AUC	0.9795	0.9424	0.9291	0.9634
	Time	1997	2176	2041	2063
SqueezeNet	Senitivity	0.9128	0.8722	0.9134	0.9045
	Specificity	0.9730	0.9761	0.9718	0.9673
	PR-AUC	0.9733	0.9680	0.9693	0.9657
	Time	1673	1796	1708	1706
DenseNet	Senitivity	0.9580	0.9784	0.9718	0.9596
	Specificity	0.9969	0.9909	0.9897	0.9912
	PR-AUC	0.9950	0.9956	0.9945	0.9939
	Time	3567	3581	3605	3579

Table 5: Standard Error of Experiment 2

	Augmentation Pipeline	Baseline	Baseline + Crop	Baseline + Crop + Distortion	Baseline + Crop + Zoom
ResNet	Senitivity	0.0542	0.0195	0.0234	0.0428
	Specificity	0.0132	0.0099	0.0251	0.0251
	PR-AUC	0.0011	0.0134	0.0047	0.0021
	Time	16.7	141.0	4.6	33.6
SqueezeNet	Senitivity	0.0148	0.0364	0.0085	0.0109
	Specificity	0.0052	0.0089	0.0056	0.0051
	PR-AUC	0.0008	0.0010	0.0006	0.0012
	Time	9.9	86.7	7.3	18.3

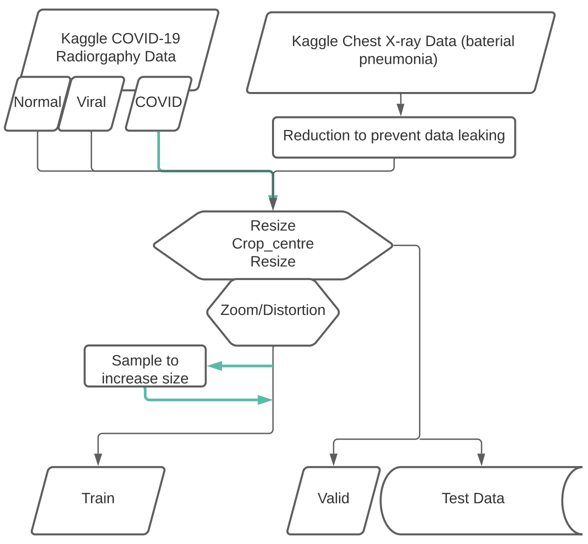

Kaggle COVID-19 Radiorgaphy Data
Normal
Viral
COVID
Kaggle Chest X-ray Data (baterial pneumonia)
Reduction to prevent data leaking
Resize
Crop_centre
Resize
Zoom/Distortion
Sample to increase size
Train
Valid
Test Data

Figure 6: Diagram of Pipeline 3 and 4

Table 6: Confusion Matrix of ResNet50 with Pipeline 1, Trial 3 on Set 2 (sensitivity:0.9635)

	prediction			
	bacteria	COVID	normal	viral
bacteria	636	4	10	69
COVID	0	1742	65	1
normal	0	275	4790	31
viral	6	13	29	625

Table 7: Confusion Matrix of ResNet50 with Pipeline 3, Trial 3 on Set 2 (sensitivity:0.9613)

	prediction			
	bacteria	COVID	normal	viral
bacteria	624	2	1	92
COVID	4	1738	59	7
normal	7	407	4621	61
viral	35	5	26	607

1. Baseline: this pipeline only included resizing.
2. Crop: baseline + `crop_centre` with a probability of 0.1. This pipeline was applied to both training, validation and testing set because the range of the CXR images were universally wider than we needed and all images required some type of cropping.
3. Crop & distortion: random distortion of 10×10 square was applied with 0.1 probability and 3 magnitude. Distortion was only applied on the training set. An example of elastic distortion in `Augmentor` is shown in Figure 4.
4. Crop & zoom: random zoom of 80% area with a probability of 0.1 was applied. Zoom was only applied on the training set. An example of zoom is shown in Figure 4.

Figure 6 illustrated Pipeline 3 and 4. Distortion was adopted in the Deep-COVID study [5]. Zoom was used to magnify the ground glass opacity feature in the COVID-19 X-ray images [26]. The area identified as signs of COVID-19 are often a small area of the lung; hence zooming in the images could be helpful. For this experiment, we reduced epochs to 20, and used the same learning rate $5e-4$.

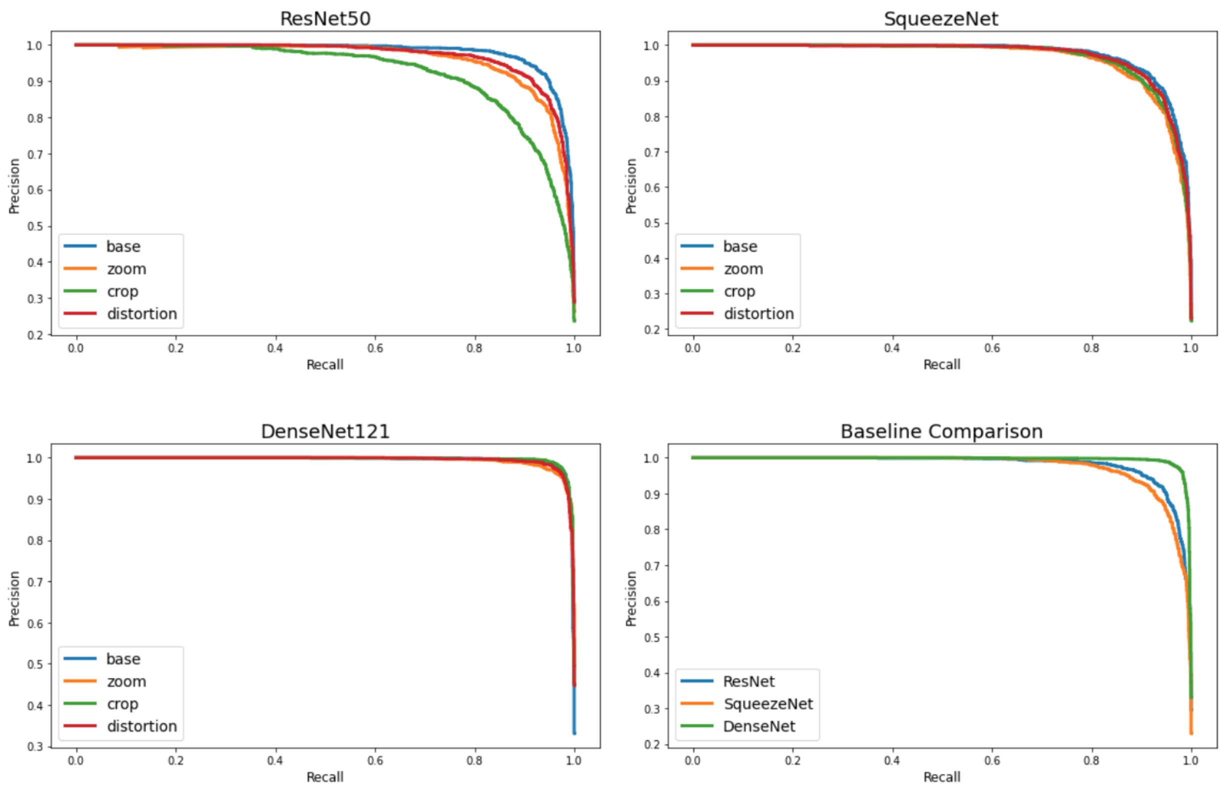


Figure 7: Precision-recall curve of ResNet for all Pipelines, trained on Set 2

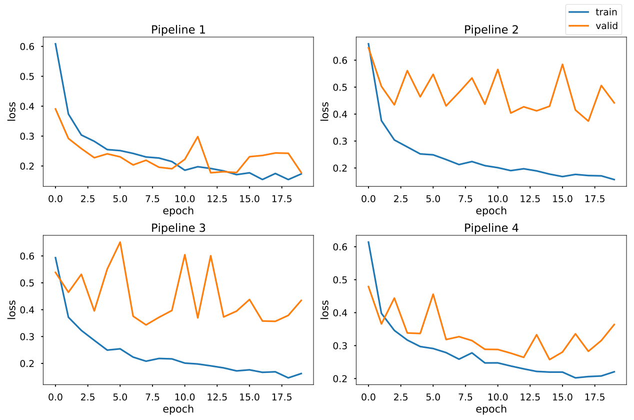


Figure 8: Loss curve for ResNet50 in Experiment 2 on Set 2.

The model performance and associated training and inferencing time is listed in Table 4. In this table, the performance was calculated on the test set and from the average of three times of training and inferencing for each combination [6]. The time spent on the entire pipeline was recorded, including data prepossessing, training, and inferencing. The standard error calculated from three times of trial was listed in Table 5. Among three pretrained-models, DenseNet121 reached the best sensitivity, specificity, and PR-AUC score, but it took the longest time to train. Pipeline 2 of DenseNet reached the highest sensitivity score and the highest PR-AUC. For ResNet50, Pipeline 3 increased sensitivity by 0.0029 (compared with the lowest sensitivity score), while lowered the specificity score by 0.09 (compared with the highest specificity score). SqueezeNet was the fastest to train and the Pipeline 3 achieved the best sensitivity, while Pipeline 1 maintained the best specificity. The computation time used for training SqueezeNet was about the half of the time for DenseNet121.

In Table 5, the standard error from each model run was recorded. The standard error associated with sensitivity scores were mostly higher than that associated with the specificity score, except for Pipeline 3 in ResNet. This was likely caused by the number of images from other three classes (normal, viral and bacterial) were higher than the number of images in the COVID-19 class alone.

After the correction on the augmentation pipeline, the training loss and validation loss showed more similar decreasing trends (Figure 8).

We further demonstrated the precision-recall curves and the confusion matrices of the model performance on the testing dataset. As shown in Figure 7, precision-recall curves for COVID-19 class were plotted. The baseline pipeline shows the best ability to classify over all models. Although sensitivity and specificity were measured, these metrics are not quite robust since they do not take false predictions into account. This is also reflected in the comparison of sensitivity-specificity and the PR-AUC (Table 4), as the highest PR-AUCs do not always associate with the highest sensitivity-specificity scores. Taking the third run for ResNet50 Pipeline 1 and Pipeline 3 as an example, the sensitivity of both models were similar (Pipeline 1: 0.9635; Pipeline 3: 0.9613), but their precision-recall curves had differences. From the confusion matrices, we see that the classifier with distortion augmentation often confuses COVID-19 images with normal images and thus generated more false positives and lowered precision. Nevertheless, both classifier

[6]DenseNet was trained once for each pipeline due to limited GPU quota

have correctly predicted more than 1730 images of COVID-19 class. The heightened average sensitivity in the distortion augmentation of ResNet50 in Table 4 was likely caused by random effect. This finding highlighted the importance of using multiple metrics to comprehensively evaluate the performance of classifiers.

3.2 Learning Rate Optimization

3.2.1 *Experiment 3: one cycle learning*

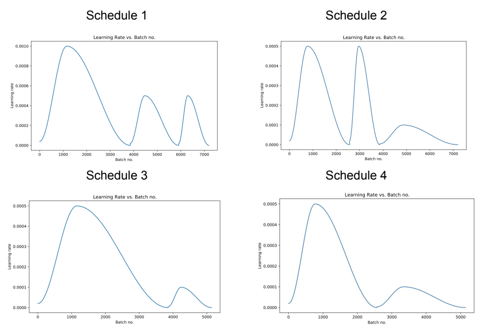


Figure 9: Visualization of Learning Rate for 1cycle Policy

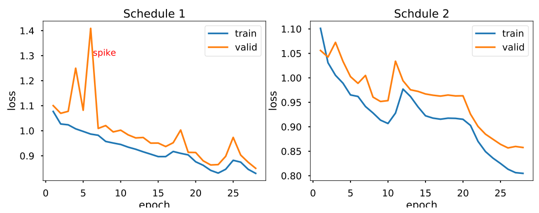


Figure 10: Loss curve for 1cycle policy, Schedule 1 and Schedule 2. The spike was labeled in the left figure.

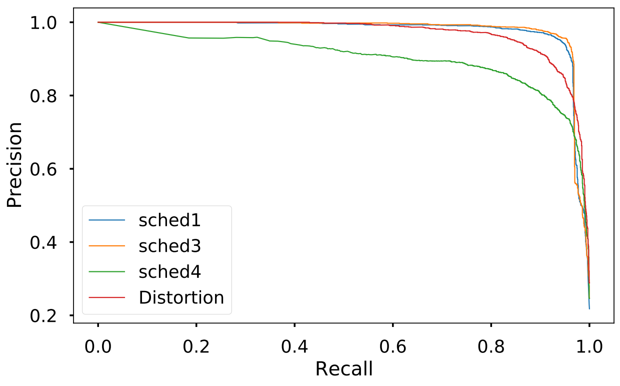


Figure 11: PR curve for 1cycle learning policy, compared with Experiment 2

We fit the model through batch-by-batch dynamic learning rate as demonstrated by A. Nain [24]. We used `fit_one_cycle` function with `torch.optim.lr_scheduler.OneCycleLR` to implement the one cycle learning in ResNet50. The design of schedules were shown in Table 8, and the learning rates were shown in Figure 9. We first experimented with different maximum learning rate within 28 epochs in Schedule 1 and Schedule 2. In those two schedules, we also added weight decay parameter (1e-4) to help regularize and gradient clipping (0.1) to prevent gradient explosion. Figure 10 shows that Schedule 2 performed slightly better in decreasing the validation losses as it did not cause the spike of validation loss in Figure 10. Both Schedules showed similar level of performance. We also find that a large cycle, such as 10 to 15 iteration is better than a small cycle such as 5 iterations. We then added another two schedules with 20 epochs in total, so this matched the number of epochs used in Experiment 2. The results in Table 8 was calculated based on two model runs. We see that the sensitivity scores from Schedule 3 and 4 (average sensitivity: 0.9496) were improved by at least 0.4 compared with the models in Experiment 2 (average sensitivity: 0.9019) with only a constant learning rate. It appeared that this policy also helped to train a more robust classifier, as shown in Figure 11.

3.2.2 *Experiment 4: reduced learning rate on plateau*

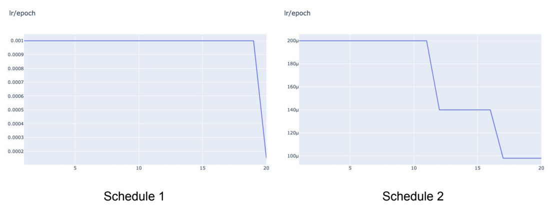


Figure 12: Learning Rate Schedule for ReduceOnPlateau

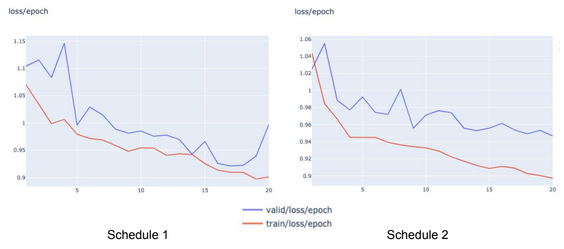


Figure 13: Loss Curve for ReduceOnPlateau Policy

We trained ResNet50 using a reduced learning rate policy - `ReduceLROnPlateau` in pytorch. This type of adding ‘patience‘ to a learning rate policy was also reported in the COVID Net study [4]. Two schedules were applied in this study as listed in Table 9. The starting learning rate of Schedule 2 is lower than Schedule 1, while the factor for decreased learning rate is higher. Schedule 2 has the same parameter used in the COVID Net study [4]. For this study, we also used `catalyst` to train the model instead of writing a customized training function. The final learning rates for 20 epochs were plotted in Figure 12 and the loss curves were in Figure 13. Table 9 recorded the sensitivity and specificity averaged from two independent model runs. We discovered that starting from a smaller learning rate resulted in better performance since a large learning rate made the loss curve fluctuate at the epochs in the end (Figure 13). The sensitivity in Schedule 2 reached the same level with the Schedule 4 in the 1cycle experiment. The specificity in Schedule 2 was lower than Schedule 4 in the 1cycle experiment.

Table 8: Schedule and Performance for One Cycle Learning in ResNet50

Schedule	Learning rate and epochs	Sensitivity	Specificity	Epochs	Time
Schedule 1	15 epochs (1e-3); 8 epochs (5e-4); 5 epochs (5e-4)	0.9519	0.9737	28	3716
Schedule 2	10 epochs (5e-4); 5 epochs (5e-4); 13 epochs (1e-4)	0.9502	0.9861	28	3551
Schedule 3	15 epochs (5e-4); 5 epochs (1e-4)	0.9483	0.9856	20	2902
Schedule 4	10 epochs (5e-4); 10 epochs (1e-4)	0.9509	0.9608	20	2904

Table 9: Schedule for Reduce on Plateau Learning Policy in ResNet50

Schedule	Learning rate parameter	Sensitivity	Specificity	Epochs	Time
Schedule 1	Start LR=1e-3, patience=2, factor=0.15	0.8637	0.8176	20	3242
Schedule 2	Start LR=2e-4, patience=2, factor=0.7	0.9544	0.9044	20	3294

3.3 Overall Comparison

3.3.1 Experiment 5: CNN from scratch

As a comparison we also implemented a simple CNN as a comparison. The model consists of six layers with one conv layer, one max pooling as shown in Figure 14. We ran the model for 20 epochs with learning rates of $2e-4$ and $5e-4$ respectively. This model was run twice and the results were listed in Table 10. We found that the model with learning rate of 5e-4 achieved better sensitivity than of 2e-4. Compared with experiment 3 and 4, this model has similar level of specificity but sensitivity was quite low.

3.3.2 Summary of all experiments

We selected the best sensitivity from each experiment and compared the results in Table 11. In summary, DenseNet from Experiment 2 has exceeded other models, but it took the longest time to train. 1cycle learning policy in Experiment 3 is capable of using 20% less time used in Experiment 2 to achieve above 0.95 sensitivity. ReduceOnPlateau policy has also achieved above 0.95 sensitivity. From the comparison on PR-AUC scores and PR curves, we discovered that the data augmentation techniques used in this study were not beneficial to the robustness of classifiers. Baseline pipelines in ResNet50 and SqueezeNet in Experiment 2 have reached the best PR-AUC scores.

Table 10: Model performance for Experiment 5 using different learning rates

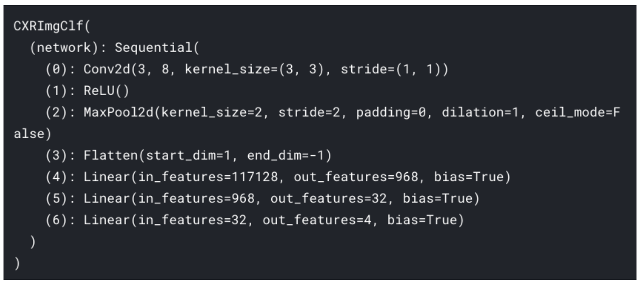
Learning Rate	Sensitivity	Specificity	Time
2.00E-04	0.9032	0.9817	1750
5.00E-04	0.8968	0.9761	1921

Table 11: Overall comparison of sensitivity and specificity. Best performance was selected from the model that achieved the best sensitivity from each experiment.

	Sensitivity	Specificity	Time
Experiment 2	0.9784	0.9909	3581
Experiment 3	0.9509	0.9608	2904
Experiment 4	0.9544	0.9044	3294
Experiment 5	0.9032	0.9817	1750

4. DISCUSSION

In this section, we discuss the data source, the technique, the implications of transfer learning and potential future improvement.



```
CXRImgClf(
  (network): Sequential(
    (0): Conv2d(3, 8, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=F
alse)
    (3): Flatten(start_dim=1, end_dim=-1)
    (4): Linear(in_features=117128, out_features=968, bias=True)
    (5): Linear(in_features=968, out_features=32, bias=True)
    (6): Linear(in_features=32, out_features=4, bias=True)
  )
)
```

Figure 14: Architecture of CNN model used for comparison

4.1 Data Source

The original data source used by the Deep-COVID study[5], curated by Cohen et al [15] has only 184 images in the COVID-19 category. This dataset has expanded into 3616 images after a year into the pandemic. The increase of the dataset is undoubtedly a contributing factor to the improvement of the performance. In the Deep-COVID study, 100 epochs of funetuning achieved more than 98% sensitivity rate. The DenseNet121 in the present study only used 20 epochs to achieve above 97% sensitivity. However, this is not a fair comparison since the dataset for other categories are not the same. Compare to the number of global hospitalizations, the size of this dataset is still quite trivial. We expect as more people engage in the research of screening of diseases by X-ray images, more dedicated resource will be allocated to collect and label these images, and images for other type of diseases.

4.2 Transfer Learning

Transfer learning has provided a convenient and efficient approach in image recognition. In a considerable number of data science competitions in image classification, the winning solutions have incorporated some flavor of transfer learning. To name a few, SIIM-ACR Pneumothorax Segmentation [27] and APTOS 2019 Blindness Detection [28] both have used transfer learning. The goal of SIIM-ACR Pneumothorax Segmentation Competition is to build an image classifier to detect pneumothorax on x-ray imgaes, which is sometimes difficult for radiologist to diagnose. The winning solution was an ensemble model which includes blending of resNet34, resNet50 and SCSEUnet [29]. APTOS 2019 Blindness Detection challenge asked participants to build a model to classify the severeness of retinopathy from retina images. The winning solution of this challenge was an average of eight models including seResNet, inception and inception_resnet, etc [30]. Interestingly, since the images were highquality, the competitor chose not to use any data augmentation except resize.

4.3 Challenges in Deep Learning Project

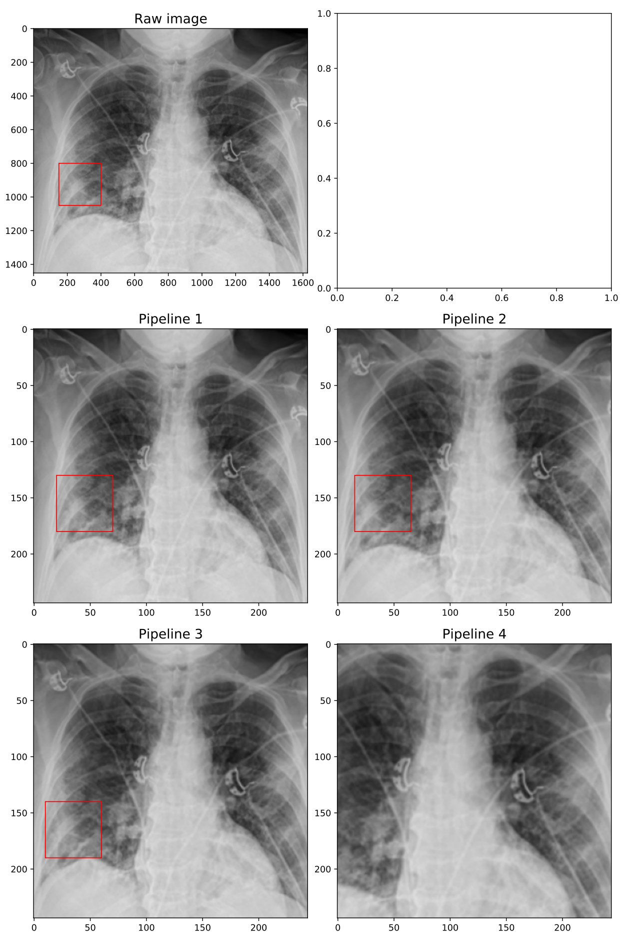


Figure 15: Example of an CXR image with regions of infection of COVID-19 identified by D. Geffen [26], and images transformed by each pipeline in Experiment 2. The red rectanglar region is typical COVID-19 infection representation, multifocal rounded opacities and nodules

We encountered several challenges in this study. The biggest challenge was to utilize limited GPU resources and run pipeline efficiently. Kaggle Notebook has 40hrs of free GPU resource weekly, but this was still not sufficient for this study. Specifically, the Save and Commit function allows users to manually trigger a versioned pipeline. However, multiple pipelines cannot be run simultaneously so we spent several hours waiting for the results. This can be solved by using more powerful cloud computing resources, but the users would need to migrate the dataset, and configure the environment as well. Moreover, the version for notebook artifacts and the model performance were recorded manually. This was not convenient and prone to mistake. We hope to explore other tools or functionalities to better orchestrate the model training, to document the model performance and to version control the pipelines.

Furthermore, we have not created efficient and robust data augmentation pipeline to improve the model performance. Originally, we wanted to use these techniques to highlight or magnify the special features in the COVID-19 class. Rotation was used to correct the tilted images. Distortion was used to add variety of the grounded-glass opacity. Cropping and zooming were to magnify the specific opaque area. However, these techniques have created unintended effect to undermine the similarity between training and validation data. This has changed the distribution of the data and hurt the learning ability of the model, as seen in Figure 8 (Pipeline 2 and 3). There existed a gap between training and validation loss in Pipeline 2 and Pipeline 3. In Pipeline 1, the learning curves of training and validation appeared to have more similar trends, and it was likely because the center cropping was not used. It was not clear to us why the gap between two loss curves became narrower again in Pipeline 4, maybe the effect of zoom counterpart the effect of cropping. It is suspected that the zoom augmentation have decreased the visibility of lung morphology and cardiothoratic ratio (CTR), which are both critical biomarkers for CXR images [31]. Figure 15 is an example of using a CXR image identified by the experts, and transformed by the pipelines in Experiment 2. It is clearly that the infected region has been erased in Pipeline 4. Looking retrospectively, it would be beneficial to collaborate with radiologists to identify the correct data augmentation techniques for the purpose of diagnosing COVID-19.

Additionally, oversampling the images in COVID-19 class also created unintended effect to make the training process more unpredictable. The standard error for sensitivity and specificity from three separate trials (Table 5) can be as high as 0.05. We could reduce the randomness in the data pipeline, and create more controlled data augmentation process. Nevertheless, this random effect did not affect the evaluation of results as we have seen steady better performance in DenseNet compared with other models.

4.4 Application and Future Direction

This sets of models are not production ready due to the following reasons: 1) The explainability of the model: this is a universal problem for deep learning models [32]. There are tools to improve explainability such as LIME, CNNViz, etc [33, 34]. Furthermore, image segmentation and object detection can help to narrow down the area of specific characteristics. In the Deep-COVID study [5], the author compared the heatmap generated from the pretrained model and the region of COVID-19 infection identified by the radiologists. This helped to understand and diagnose the model performance as well. 2) Performance: The current performance is still unstable and requires hyperparameter tuning and model selection. 3) Deployment, update and inference: to build an AI-assisted image recognition system, the model has to be deployable, and there has to be production-grade pipeline to support the deep learning model. These components include the data collection, feature extraction, serving, monitoring, etc [35].

The biggest advantage of CXR imaging for COVID-19 screening is that the result can be immediately available, while an RT-PCR test takes at least 2-3 hours for its procedure and up to 5 days to report a result to the patients [3]. This rapid procedure of CXR imaging can help to inform the patients early and implement self-isolation to further prevent the disease spread. The downside of CXR imaging for COVID-19 diagnosis includes: 1) CXR images cannot detect asymptomatic cases and very early stage of infection; 2) The cost of an CXR image is \$260-\$460 [36], which is higher than the cost of a RT-PCR test (\$100) [37].

Chest X-ray image dataset is one of the most abundant source of medical images, and there has been a couple of studies demonstrating high-level performance in disease diagnosis [38]. We hope that this study can contribute to the knowledge of applying deep learning in medical image diagnosis, and facilitate triage, supporting physicians and radiologist to make quicker and more data-driven diagnosis. Future direction of the present study includes:

- Design customized data augmentation for identifying the features in COVID-19 and other diseases, collaborate with radiologists to understand the disease features.
- Improve the sensitivity and specificity of prediction of other classes.
- Create architectures that utilizes ensemble models, and train classifiers on those models.
- Validate the predictions by working with radiologists, to better analyze prediction errors.

5. CONCLUSIONS

While many studies have reported robust performance of customized convolutionary neural network models, this study demonstrates that using pretrained model can still perform well enough in detection of COVID-19 from CXR images. Using only 1 hrs of training time, the pretrained model can generally achieve above 95% sensitivity and above 95% specificity. To assess the effect of data augmentation, we designed experiments using techniques that help to retain the special features of COVID-19 CXR images. Cropping and distortion can increase the model sensitivity by a small amount, while sacrifice the specificity. It would also be advisable to apply other methods of augmentation, and different sampling treatment towards different classes of images. We also reiterated the importance of learning rate optimization in model training. Notably, one cycle learning policy and ReduceOnPlateau policy are capable of optimizing performance within the same number of iterations, but to determine the initial parameter requires repeatable validation and experimentation. These policies can also increase the efficiency of model training.

The area of deep learning on medical images is active, as more studies have reported human-level performance. To choose the suitable model to help with COVID-19 detection on CXR images, model performance, computation time, interpretability and portability have to be taken into consideration.

6. ACKNOWLEDGMENTS

We would like to thank the instructor and the TAs for this course for helping to address issues in this study.

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] H. Ritchie, E. Ortiz-Ospina, D. Beltekian, E. Mathieu, J. Hasell, B. Macdonald, C. Giattino, C. Appel, M. Roser, E. v. Woerden, D. Gavrilov, M. Bergel, J. Crawford, and M. Gerber, "Coronavirus Pandemic (COVID-19)," Apr. 2021. [Online]. Available: https://ourworldindata.org/coronavirus

[2] D. E. Bloom, D. Cadarette, M. Ferranna, R. N. Hyer, and D. L. Tortorice, "How New Models Of Vaccine Development For COVID-19 Have Helped Address An Epic Public Health Crisis," *Health Affairs*, p. 10.1377/hlthaff.2020.02012, Feb. 2021, publisher: Health Affairs. [Online]. Available: https://www.healthaffairs.org/doi/full/10.1377/hlthaff.2020.02012

[3] Labcorp, "LabCorp COVID-19 RT-PCR test - EUA Summary," Tech. Rep., 2020.

[4] L. Wang and A. Wong, "COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest X-Ray Images," *arXiv:2003.09871 [cs, eess]*, May 2020, arXiv: 2003.09871 version: 4. [Online]. Available: http://arxiv.org/abs/2003.09871

[5] S. Minaee, R. Kafieh, M. Sonka, S. Yazdani, and G. J. Soufi, "Deep-COVID: Predicting COVID-19 From Chest X-Ray Images Using Deep Transfer Learning," *arXiv:2004.09363 [cs]*, Jul. 2020, arXiv: 2004.09363 version: 3. [Online]. Available: http://arxiv.org/abs/2004.09363

[6] Z. Qiao, A. Bae, L. M. Glass, C. Xiao, and J. Sun, "FLANNEL (Focal Loss bAsed Neural Network EnsembLe) for COVID-19 detection," *Journal of the American Medical Informatics Association*, vol. 28, no. 3, pp. 444–452, Mar. 2021. [Online]. Available: https://academic.oup.com/jamia/article/28/3/444/5943880

[7] E. S. Olivas, J. D. M. Guerrero, M. Martinez-Sober, J. R. Magdalena-Benedito, and A. J. Serrano López, Eds., *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques.* IGI Global, 2010. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-766-9

[8] P. HUILGOL, "Top 4 Pre-Trained Models for Image Classification | With Python Code," Aug. 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/

[9] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *arXiv:1409.0575 [cs]*, Jan. 2015, arXiv: 1409.0575. [Online]. Available: http://arxiv.org/abs/1409.0575

[10] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Jul. 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

[11] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay," *arXiv:1803.09820 [cs, stat]*, Apr. 2018, arXiv: 1803.09820. [Online]. Available: http://arxiv.org/abs/1803.09820

[12] S. Gugger, "The 1cycle policy," section: Experiments. [Online]. Available: /the-1cycle-policy.html

[13] I. S. of Medical and Interventional Radiology, "COVID-19 DATABASE | SIRM," Oct. 2020. [Online]. Available: https://www.sirm.org/category/senza-categoria/covid-19/

[14] "BIMCV-COVID19 – BIMCV." [Online]. Available: https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/

[15] J. P. Cohen, "ieee8023/covid-chestxray-dataset," Mar. 2021, original-date: 2020-02-14T23:22:23Z. [Online]. Available: https://github.com/ieee8023/covid-chestxray-dataset

[16] "RSNA Pneumonia Detection Challenge." [Online]. Available: https://kaggle.com/c/rsna-pneumonia-detection-challenge

[17] T. Rahman, A. Khandakar, Y. Qiblawey, A. Tahir, S. Kiranyaz, S. B. Abul Kashem, M. T. Islam, S. Al Maadeed, S. M. Zughaier, M. S. Khan, and M. E. H. Chowdhury, "Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-rays Images," *Computers in Biology*

and Medicine, p. 104319, Mar. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S001048252100113X

[18] M. E. H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. A. Emadi, M. B. I. Reaz, and M. T. Islam, "Can AI Help in Screening Viral and COVID-19 Pneumonia?" *IEEE Access*, vol. 8, pp. 132 665–132 676, 2020, conference Name: IEEE Access.

[19] P. Mooney, "Chest X-Ray Images (Pneumonia)." [Online]. Available: https://kaggle.com/paultimothymooney/chest-xray-pneumonia

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385

[21] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv:1602.07360 [cs]*, Nov. 2016, arXiv: 1602.07360. [Online]. Available: http://arxiv.org/abs/1602.07360

[22] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *arXiv:1608.06993 [cs]*, Jan. 2018, arXiv: 1608.06993 version: 5. [Online]. Available: http://arxiv.org/abs/1608.06993

[23] "DenseNet121 | pytorch." [Online]. Available: https://kaggle.com/leighplt/densenet121-pytorch

[24] A. Nain, "Deep Learning with PyTorch: Zero to GANs | Jovian." [Online]. Available: https://jovian.ai/learn/deep-learning-with-pytorch-zero-to-gans

[25] S. Kolesnikov, "Catalyst." [Online]. Available: https://catalyst-team.com/

[26] S. Simpson, F. U. Kay, S. Abbara, S. Bhalla, J. H. Chung, M. Chung, T. S. Henry, J. P. Kanne, S. Kligerman, J. P. Ko, and H. Litt, "Radiological Society of North America Expert Consensus Document on Reporting Chest CT Findings Related to COVID-19: Endorsed by the Society of Thoracic Radiology, the American College of Radiology, and RSNA," *Radiology: Cardiothoracic Imaging*, vol. 2, no. 2, p. e200152, Apr. 2020. [Online]. Available: http://pubs.rsna.org/doi/10.1148/ryct.2020200152

[27] "SIIM-ACR Pneumothorax Segmentation." [Online]. Available: https://kaggle.com/c/siim-acr-pneumothorax-segmentation

[28] "APTOS 2019 Blindness Detection." [Online]. Available: https://kaggle.com/c/aptos2019-blindness-detection

[29] A. Aimoldin, "sneddy/pneumothorax-segmentation," Apr. 2021, original-date: 2019-09-06T11:45:28Z. [Online]. Available: https://github.com/sneddy/pneumothorax-segmentation

[30] G. Xu, "1st place solution summary-APTOS 2019 Blindness Detection." [Online]. Available: https://www.kaggle.com/c/aptos2019-blindness-detection/discussion/108065

[31] Y. Oh, S. Park, and J. C. Ye, "Deep Learning COVID-19 Features on CXR Using Limited Training Data Sets," *IEEE Transactions on Medical Imaging*, vol. 39, no. 8, pp. 2688–2700, Aug. 2020, conference Name: IEEE Transactions on Medical Imaging.

[32] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models," *arXiv:1708.08296 [cs, stat]*, Aug. 2017, arXiv: 1708.08296. [Online]. Available: http://arxiv.org/abs/1708.08296

[33] M. T. C. Ribeiro, "marcotcr/lime," Apr. 2021, original-date: 2016-03-15T22:18:10Z. [Online]. Available: https://github.com/marcotcr/lime

[34] P. Malviya, "pranshu28/cnn-viz," Feb. 2021, original-date: 2018-06-22T10:50:00Z. [Online]. Available: https://github.com/pranshu28/cnn-viz

[35] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden Technical Debt in Machine Learning Systems," p. 9.

[36] CostHelper, "How Much Does an X-Ray Cost?" [Online]. Available: https://health.costhelper.com/x-rays.html

[37] K. diagnostics, "COVID-19 RT-PCR Test | KSL Diagnostics | United States." [Online]. Available: https://www.ksldx.com/covid-19-pcr-test

[38] K. Yasaka and O. Abe, "Deep learning and artificial intelligence in radiology: Current applications and future directions," *PLOS Medicine*, vol. 15, no. 11, p. e1002707, Nov. 2018, publisher: Public Library of Science. [Online]. Available: https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1002707