Machine Learning in Natural Sciences - Course Project

Deep Network for diagnosis of Covid-19 from Chest X-ray images

Advisor: Deva Priyakumar
By Abhishek Trivedi (20161153), Akshaya Karthikeyan (20171016)
and Kshitij Gupta (20161234)

Background

COVID-19 is an infectious disease caused by SARS-CoV-2. The disease has spread globally, resulting in the ongoing 2019–20 coronavirus pandemic.

Radiology is important as chest X-ray is the first imaging method to diagnose COVID-19 in many countries. One significant factor that limits diagnosis is the duration of tests for the virus. Using deep learning methods on patients' chest x-rays, we aim to detect the COVID-19 disease with high accuracy and in less time.

Chest x-rays are chosen for our model because they are more accessible than CT scans. More potential data will also be available. In case radiologists and medical professionals can't work, AI systems are essential to continue diagnosis.

Problem Statement

Detection of Covid-19 from Chest X-ray images using deep learning techniques.

Method

We propose a deep learning approach to detect Covid-19 from chest X-ray images. We used a dedicated Covid-19 dataset available on Github, which contains X-ray images of patients after virus-testing. A new dataset was built by merging both Covid-19 and NIH datasets through *uniform sampling*.

We extended the process of dataset formation by using *GAN*. It partially helped generate X-Ray image instances to eventually handle the data imbalance problem in classification. The dataset still had very few Covid+ image instances. Thus, we include the technique of *transfer learning*, which accelerates network training with less amount of data. We take the pre-trained NIH model (Dense-Net) weights and modify their architecture to detect Covid-19. The modified DenseNet was trained on the new dataset to classify the X-ray images as Covid (+) or Covid (-) . We use the technique of *min-batch resampling* and *weighted classification losses* to tackle the problem of data imbalance (w.r.t to Covid cases) in the new dataset. We compare our transfer learning approach with baseline CNN models.

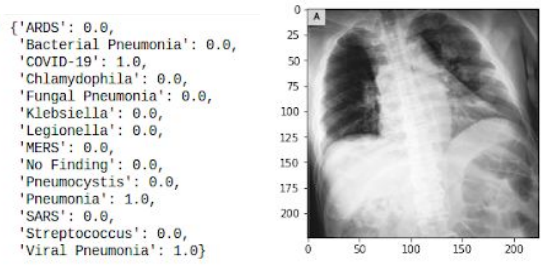
Implementation Details

Dataset

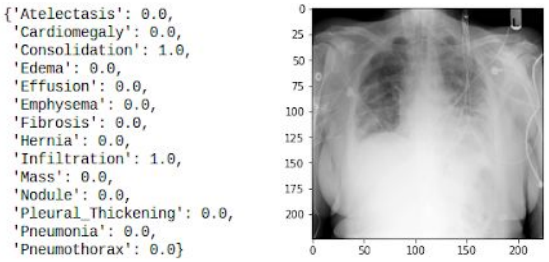
We have used the COVID-19 open-source dataset. It consists of COVID-19 images from publicly available research, as well as lung images with different diseases such as SARS, Streptococcus, and Pneumocystis.

We only selected **Posteroanterior views (PA)** of CXRs, which are the most common types of x-ray scans. We resize all images to 224 x 224 and convert them to greyscale in the preprocessing stage.

A sample from COVID19 dataset:



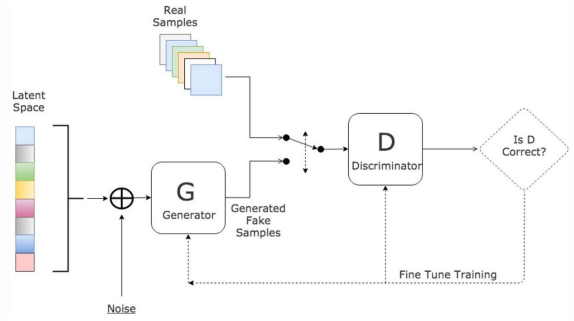
To increase the size of our dataset, we added scans from the NIH dataset. A sample:



We created a balanced dataset of all the 15 classes from the complete covid dataset and a subset of the NIH dataset. Train set: 1118 covid negatives, 104 covid positives. Val set: 21 covid negatives, 21 covid positives

Expanding the dataset using GAN

Since the number of data instances are less, we generated chest x-ray using Generative Adversarial Networks.



Discriminator: Takes a flattened image as input, and returns the probability of it being real, or fake. The input size for each image is 28x28=784 (downscaled).

Structure: Three hidden layers, each followed by a Leaky-ReLU and a Dropout layer to avoid overfitting. Then, a Sigmoid function is applied.

Generator: The input is a latent variable, and the output is a 784 valued vector, which corresponds to a flattened 28x28 image.

Structure: Three hidden layers, each followed by a Leaky-ReLU. The output layer has a TanH activation function.

Optimizer: Adam (lr = 0.0002)

Loss function: Binary Cross Entropy Loss

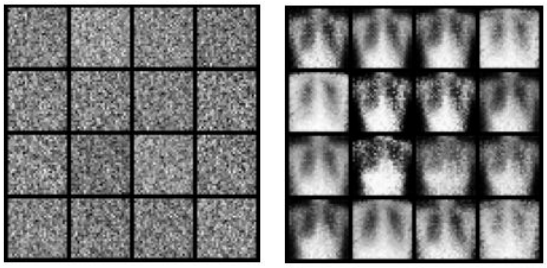


Fig: (left) Generated images at epoch 1, pure noise

(right) Generated images at epoch 200

Since the generated images are of low resolution, we considered a small number of images to account for data imbalance only.

Model Architectures

2D Convolutional Neural Network (Baseline)

```
XRayNet(
  (conv1): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1))
  (batch1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(32, 56, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (batch2): BatchNorm2d(56, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv3): Conv2d(56, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (batch3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (batch4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (fc1): Linear(in_features=10816, out_features=4096, bias=True)
  (fc2): Linear(in_features=4096, out_features=512, bias=True)
  (fc3): Linear(in_features=512, out_features=64, bias=True)
  (fc4): Linear(in_features=64, out_features=2, bias=True)
  (batch5): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (batch6): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (drop): Dropout(p=0.3, inplace=False)
  (softmax): Softmax(dim=1)
)
28
The model has 46,524,394 trainable parameters
```

Transfer Learning on NIH pre-trained model

We imported the NIH pre-trained model using the torchxrayvision library. Then, we froze all the layers except the final bottleneck convolutional and BatchNorm layers. A fully-connected linear layer for classification was concatenated at end to map the features into a 2-length softmax vector. We initialized lr = 1e-5 for these layers during training.

```
The model has 40,962 trainable parameters
```

Data Loader - Due to a huge data imbalance between covid-positive and covid-negative x-ray images in our dataset, we employed the mini-batch resampling technique to improve the model accuracy. Weights are assigned to each image according to its label's inverse frequency count. In this way, the model gets to look at both positives and negative samples with equal rate in a data batch. We chose a batch size of 10 images to train both our models.

Loss function - We used Weighted Cross-Entropy Loss. The weights assigned to 0-class (covid-negative) and 1-class (covid-positive) are -

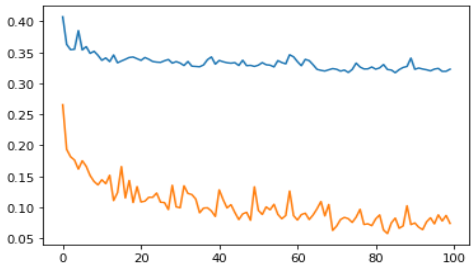
[(No. of covid +ve samples / No. of covid -ve samples) , 1.0] = [104 / 1118, 1.0] = [0.094, 1.0]

Learning rate - 0.001 with AdamW Optimizer

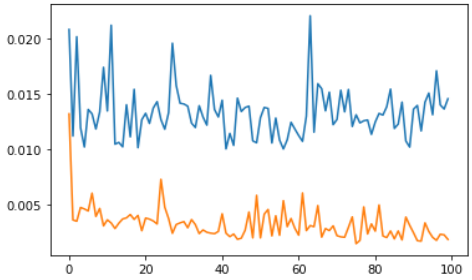
Results

Comparison of XRayNet and pretrained DenseNet model

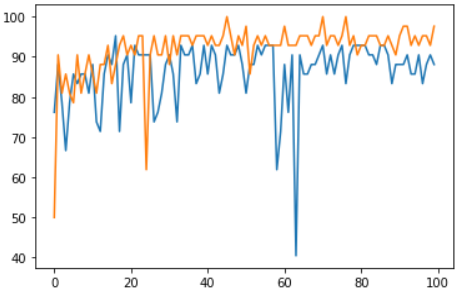
(XRayNet in Blue, DenseNet in orange)



Train Loss (Loss vs Epochs)



Validation Loss (Loss vs Epochs)



Validation Accuracy (Acc % vs Loss)

Model Name	# Epochs	Trainable Parameters	Val. accuracy
XRayNet (Baseline)	100	46,524,394	92.85
DenseNet (Transfer Learning)	100	40,962	99

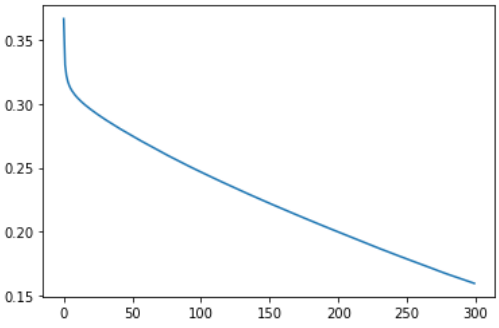
Model Extension

We tried to modify our model for classification of 15 different diseases including Covid-19. The final classification layer was modified to map the features to 15 length softmax vector.

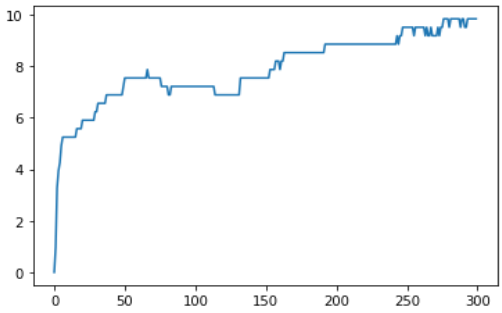
Loss Function: We employed Binary Cross-Entropy With Logits loss to handle the multi-label classification. The positive weight argument is initialized to 6.0 to remove the bias towards the majority of 0s compared to 1s in G.T vector.

Learning rate: 0.002 with AdamW optimizer

Results - Pretrained DenseNet model (15 classes)



Train loss (Loss vs Epochs)



Validation Accuracy (Acc % vs Epochs)

References

https://towardsdatascience.com/artificial-intelligence-deep-learning-for-medical-diagnosis-9561f7a4e5f

https://towardsdatascience.com/using-deep-learning-to-detect-ncov-19-from-x-ray-images-1a89701d1acd

https://arxiv.org/abs/1904.08534

https://medium.com/ai-society/gans-from-scratch-1-a-deep-introduction-with-code-in-pytorch-and-tensorflow-cb03cdcdba0f

https://en.wikipedia.org/wiki/Coronavirus_disease_2019

https://towardsdatascience.com/build-a-super-simple-gan-in-pytorch-54ba349920e4

https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff