Final Report

Goals and Discussion

Essential Goal 1

Implement a generative adversarial network model, which consists of a generator neural network and a discriminator neural network. The generative model will provide new examples based on its training set while the discriminator model will discern between original data and synthetic data produced by the generative model.

Progress

We implemented a DC-cGAN (deep convolutional conditional GAN) with 2 classes, “normal” and “with pneumonia”. In our first attempt, we tried just using a cGAN with fully-connected layers, but the generated image quality is poor as shown through very fuzzy and noisy images. Even with extensive hyperparameter tuning, we could only achieve a FID score of around 400. Thus, we switched to this new architecture taking into account that deep convolutional layers are better suited for complex image data.

Here are our results. Figure 1 shows our DG-cGAN generated images for the normal class Figure 2 shows the generated images for the pneumonia class. Figure 3 and 4 show our generator and discriminator losses as a function of training step.

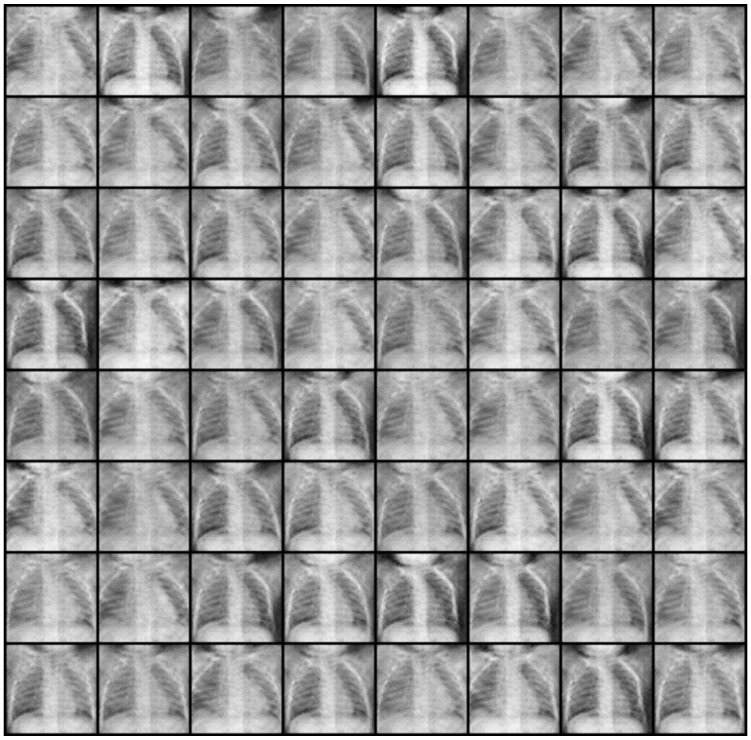
- 

Figure 1: Generated images after 200 epochs (Normal)

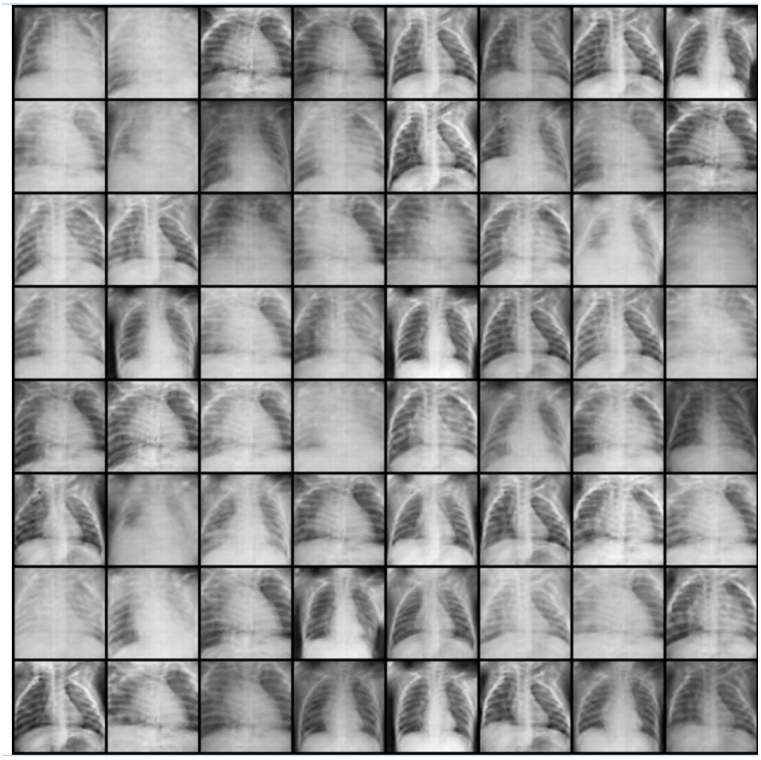


Figure 2: Generated images after 200 epochs (Pneumonia)

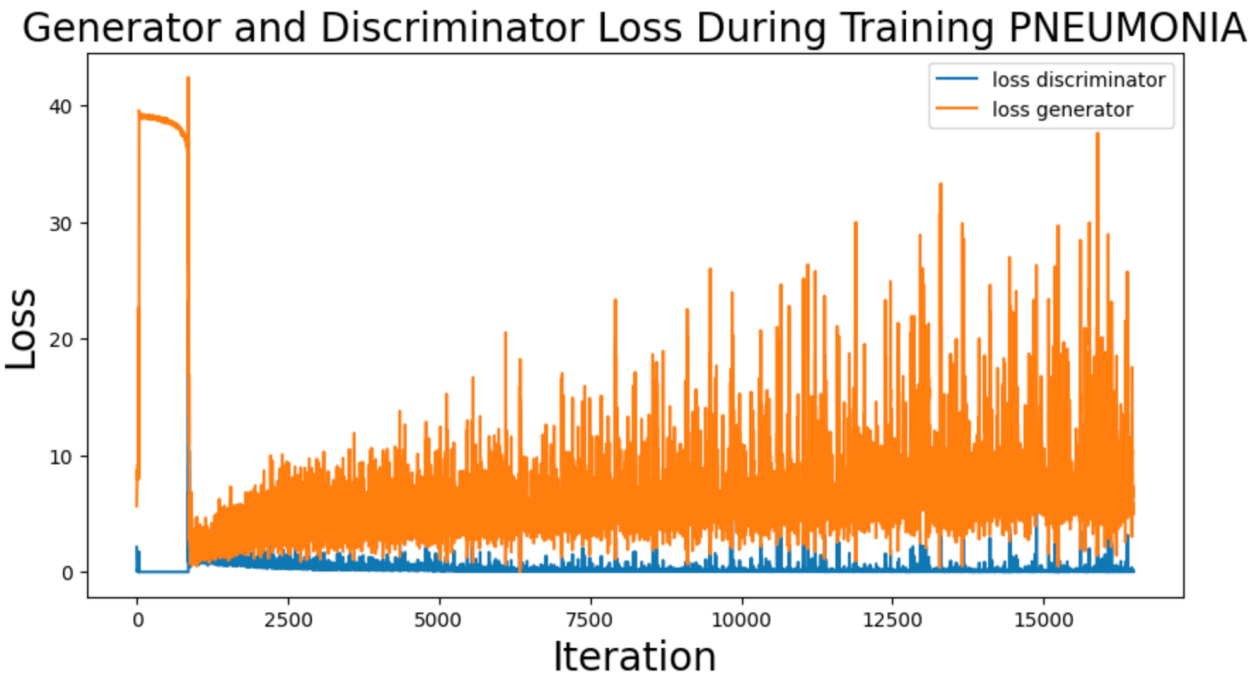


Figure 3: G and D training losses for pneumonia-class

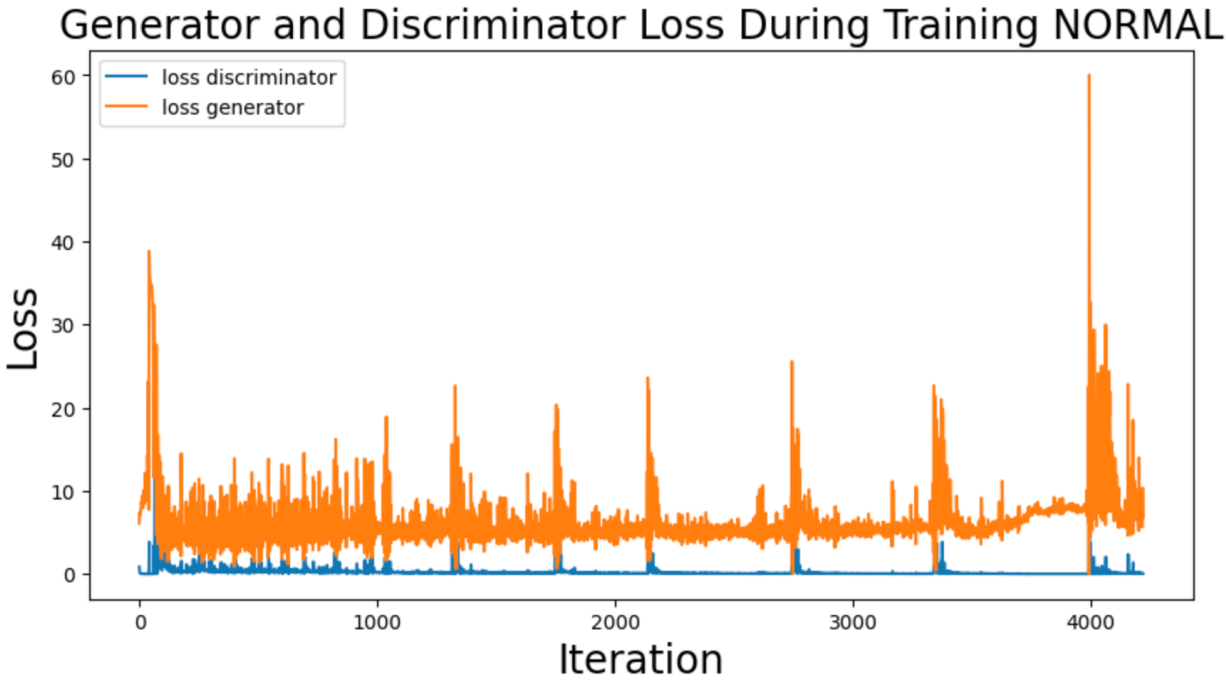


Figure 4: G and D training losses for normal-class

Interesting Parts and Challenges
As mentioned above, we had a lot of trouble initially getting the fully-connected cGAN to work; after extensive experimentation with hyperparameters, we realized that no further hyperparameter tuning is going to improve the quality of images and that the poor quality is an architectural problem. It was difficult to come up with the DC-GAN architecture, so we referenced PyTorch demos and what people have done in the past and adapted it to our dataset. We had a lot of trouble debugging the DC-GAN, specifically with regards to getting the correct dimensionality.

It was interesting to observe how different types of layers used in the GAN effect performance for this task. Our final model uses convolutional layers as opposed to dense layers, and immediately we can see the impact of this architectural change. We also learned a lot about hyperparameter tuning, which we were able to apply to tune our DG-cGAN.

Essential Goal 2

Qualitatively and quantitatively assess the images produced by the generative model by comparing the images across epochs and use the Fréchet inception distance (FID) assessment method outlined above to determine how successful the cGAN is.

Progress
We implemented FID measurements and arrived at a FID of ~219.0 for the set of normal-class images generated, and a FID of 199.5 for the set of pneumonia-class images generated (Figure 5 ,6). FID for pneumonia-class was slightly better.

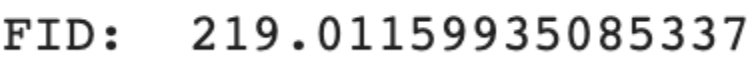


Figure 5: FID of Normal

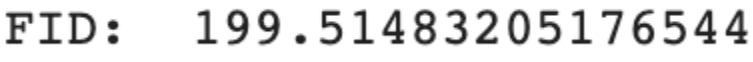


Figure 6: FID of Pneumonia

Interesting Parts and Challenges
Interestingly, the FID values seem to contradict the naked eye. As shown above, a few of the generated images for the Pneumonia positive model appear slightly noisy.

However, the FID would indicate that this model is generating higher quality images than its Pneumonia negative counterpart.

Desired Goal 1

Determine how certain cGAN hyperparameters affect the training speed and final quality of the generated images. A select of best hyperparameters shall be chosen which generate the highest quality at a certain reasonable training time.

Progress

We experimented with these hyperparameters while tuning the DG-cGAN: latent vector size, feature map size, image size, batch size, number of training epochs, learning rate, and optimizer.

We determined our final hyperparameter set (Figure 7) by analyzing the quality of generated images over 20 epoch intervals and measuring the FID. During training the model state was saved every 20 epochs, allowing us to observe the fluctuations in generated image quality and FID.

```
#latent vector size
params.nz = 100

#size of feature maps
params.ngf = 64
params.ndf = 64

params.cuda = True
params.ngpu = 1
params.imageSize = 64

#batch size
params.batchSize = 64

#epochs
params.niter = 201

#learning rate
params.lr = 0.0002

#beta optimizer for Adam optimizers
params.beta1 = 0.5
```

Figure 7: final hyperparameter set

Our key observations:

- In our fine-tuning, we found the model performance to be extremely sensitive to changes in learning rate. Values higher than 0.0002 resulted in unstable training, while lower values resulted in a much higher FID.
- The image quality ceased to improve after 200 epochs.
- Initially, we attempted to use a feature map size of 128x128, but due to insufficient RAM, this was reduced to 64x64. We experimented with lower feature map sizes, but image quality decreased significantly.

Interesting Parts and Challenges

It was difficult to analyze the effect of changing hyperparameters in tandem. For example, we had general good results when lowering feature map size to 40x40 while also decreasing learning rate. However, each of these changes produced worse images

when made independently. Moreover, the limitations in computation power limited our ability to conduct as many experiments as we'd like.

It was interesting to see that small changes in learning rate led to big changes in training results. In addition, we observed that, sometimes, combining hyperparameter changes seemed to positively affect performance despite the same changes having negative effects individually. This ties into the challenge of testing changing hyperparameters in tandem.

Desired Goal 2

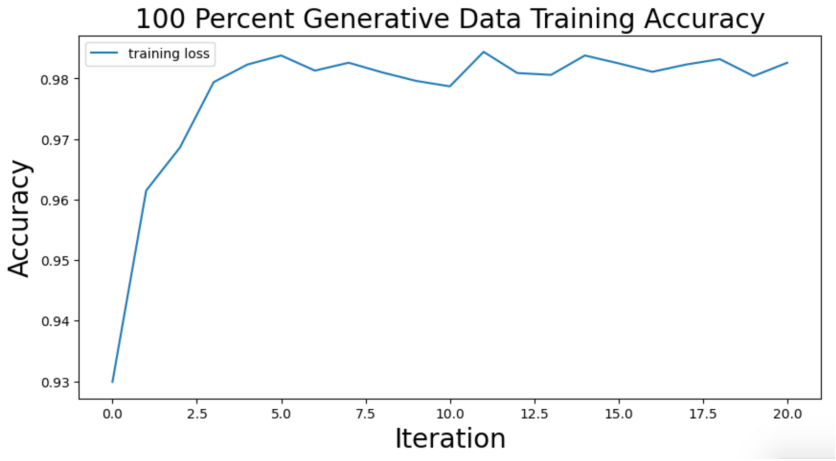
Train classifiers on both the real and an expanded dataset (including synthetic data). Compare the accuracies of these classifiers. Ideally, the accuracy of the synthetic dataset trained classifier should be improved compared to the real dataset trained classifier, indicating that using cGANs is a useful method of improving the performance of classifiers in similar datasets. However, if it is decreased, then we know the cGANs are not correctly learning certain aspects of the data that are important for classification.

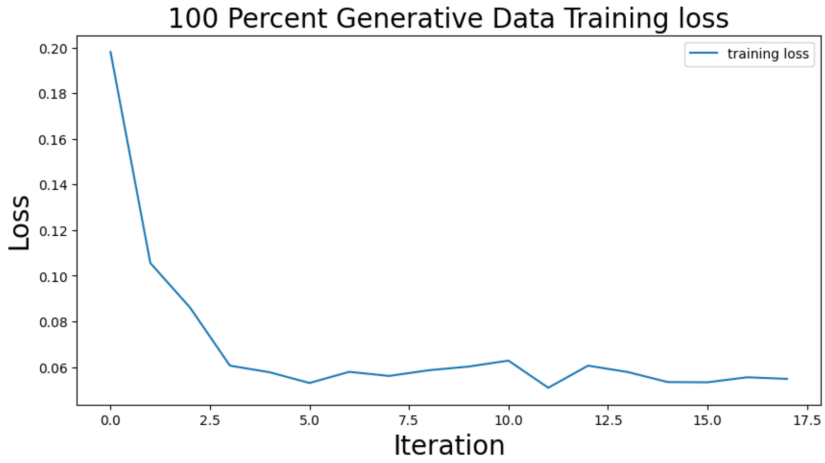
Progress
We used the pretrained ResNet-152 model to perform transfer learning in order to build our classifier. To test the synthetic images' ability to improve the classifier, we built multiple training sets composed of 0%, 10%, 20%, and 100% synthetic data. After training each model separately, the accuracy was measured on a uniform test dataset of 624 images.

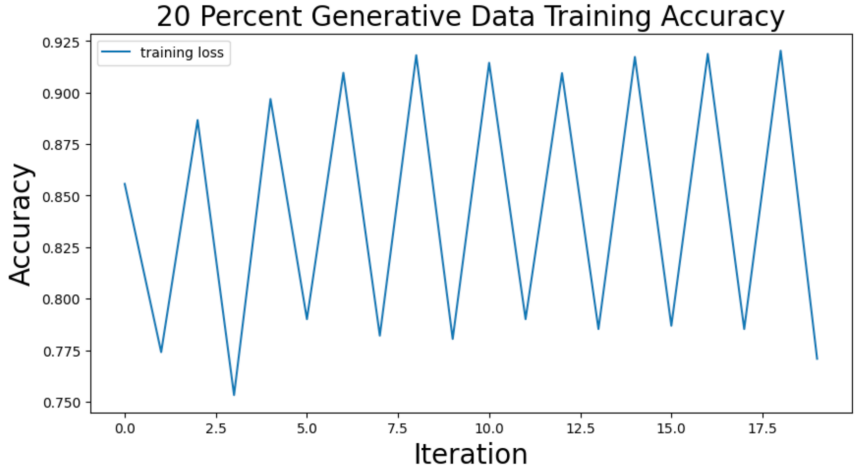
Results:

- The models trained on 0%, 10%, 20%, and 100% synthetic data achieved accuracies of 80.13%, 81.25%, 79.01%, and 65.71% respectively.
- Our generative model doesn't conclusively generate images that assist in the training of the resnet152 pretrained model architecture.
- Having the training dataset be 10% generative may improve the model performance on the testing set; more extensive experiments should be conducted to test this hypothesis, especially given that our improvement is small.
- Based on the behavior of the training of a model trained on 100% generative data, it seems like the resnet152 model is able to easily determine the core features of the images we generated, implying that the differences in our two classes were too easy to discern. This is supported by the immediate convergence of the training accuracy and the diminished training loss (Figure 8).

- The discrepancy of behavior of training the two models separately (Generator and Discriminator Loss Normal/Pneumonia) is the likely culprit to the generative images of each class displaying an inherent trait that the resnet152 model is able to discern quickly.







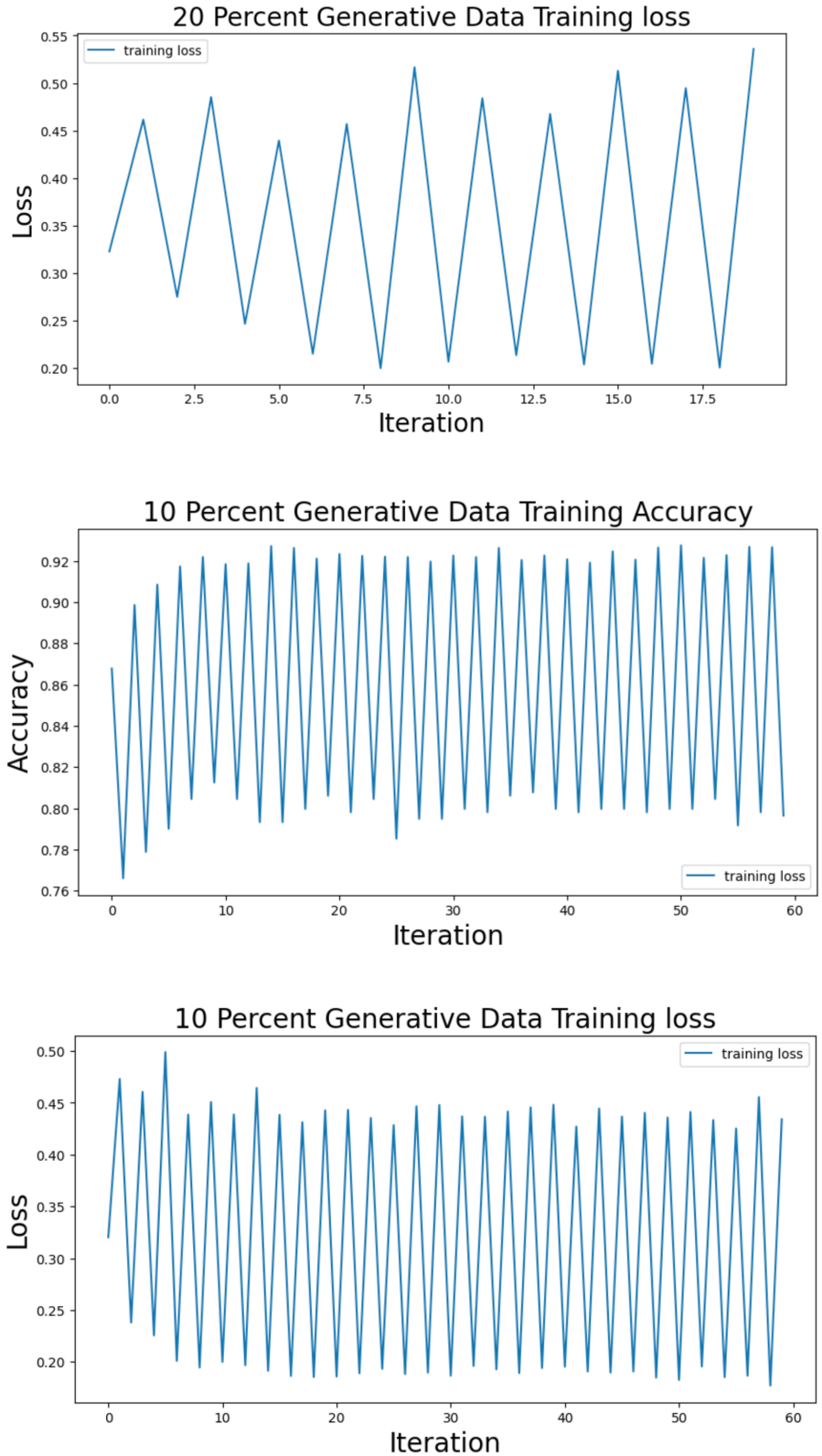


Figure 8: Training accuracy and loss for datasets with 100%, 20%, 10% synthetic data

Interesting Parts and Challenges
Training the classifier was far more straightforward than doing so for the generative model. However, it was difficult to conclusively prove whether or not the generated images had a positive, causal relationship on classifier accuracy. With a sufficiently large training set of non-synthetic images, we would expect the addition of synthetic data to have minimal effect, but we were at least able to observe a single example where a model trained partially on synthetic data outperformed the baseline.

It was unexpected and interesting to see that 10% synthetic data improved accuracy, while >=20% synthetic data did not. We also saw that ResNet-152 is able to discern between the classes of synthetic images extremely easily, likely due to our method of training the models separately.

Stretch Goal 1

Determine that at what real data training set size does the addition of synthetic data increases a trained classifier's training rate and/or accuracy. Furthermore, determine how the introduction of synthetic data changes the properties of a classifier. For instance, does it cause more type I errors even if the accuracy stays the same?

Progress
We were unable to reach this goal in time.

Stretch Goal 2

Determine how we can modify the hyperparameters mentioned above to create synthetic data that can train more accurate classifiers, as this may not exactly align with the best hyperparameters for FID.

Progress
We were unable to reach this goal in time.

Everything else

As a result of our first architecture not working as well as we intended, we ended up exploring 2 different architectures (fully-connected GAN vs deep-convoluted GAN). We observed the different training behavior of these architectures, and how different hyperparameter tuning is needed for each of them. For example, fully-connected GAN loss had trouble stabilizing and convergine, and the discriminator was also training faster than the generator, so we implemented double-training to resolve this issue. However, this did not occur with the DC-cGAN.

Code and documentation

'open_images.py' is used to read in the images from the dataset and process them into the desired format for training. Valid images are cropped and scaled down to a 128x128 image.

'pytorchCGAN.ipynb' contains all of the code used to define and train our convolutional GAN.

'X-ray-classifier.ipynb' is where we implemented transfer learning to train classifiers and test the effectiveness of including synthetic images in the training set.

Reflections

What was interesting?

The lectures included discussion around GANs and, to a more limited extent, cGANs. However, for this project we had to delve into the addition of deep convolutional layers into the structure.

Additionally, the lectures did not have much discussion about best practices involving data formatting and processing which turned out to be a large portion of this project. Different data formats not only increased accuracy, but also had a large effect on training time, determining how quickly we could iterate through designs.

Beyond this, learning to optimize hyper-parameters was an interesting component of this project, as there is not a single best way to do so. Rather we needed to learn to try and gauge the effect of several linked hyper-parameters.

What was difficult?

The biggest difficulty we faced was having to switch from a cGAN to a DC-cGAN late in the project process. A long time was spent trying to optimize the parameters of the cGAN, however we could not get a FID of better than ~400 and the images turned out fuzzy. By the time we had determined that the issues were mostly due to our choice of model, we had relatively little time remaining. While we believe we have achieved decent results in the end, we may have been able to get better ones had we started with a DC-cGAN approach.

Because of limited compute resources, training and testing a given model architecture was a lengthy process. This limited how much we were able to experiment with different models and hyperparameters, which contributed to time crunch once we settled on the DC-cGAN as the best option.

What's left to do?

In terms of our uncompleted goals, we would like to perform more experiments to conclusively determine in what scenarios our synthetic images can improve classifier accuracy. Improving the ability of a classifier to distinguish between X-rays with and without Pneumonia was the whole point of creating a GAN in the first place. Now that we can generate quality synthetic images, we would like to know under what conditions they could be used.

Given $1,000,000 we would first invest a significant portion of the budget in acquiring a larger and more diverse dataset of chest X-rays. This would involve collaborating with healthcare institutions and radiology departments to gather a broader range of images representing various pathologies, demographics, and imaging modalities. By enriching the dataset, we would enable the GAN to learn from a more comprehensive set of examples, resulting in more realistic and clinically relevant image generation.

In addition to expanding the dataset, we would allocate funds towards acquiring more powerful computing infrastructure. By investing in high-performance GPUs or utilizing cloud-based solutions, we could significantly reduce the training time and efficiently handle the increased data size. This would allow us to iterate and experiment with different GAN architectures, hyperparameters, and training techniques more rapidly, leading to better performing models.

We might also allocate a portion of the budget to collaborate with radiologists and other medical professionals. Their insights and annotations would provide invaluable guidance during the training process and help ensure that the generated images align with clinical expectations and requirements. This collaborative effort would enhance the fidelity and clinical relevance of the generated chest X-ray images, making them more useful in real-world medical scenarios.