Using a GAN for data augmentation of chest X-Ray images for detection and classification of COVID-19 virus in patients

Group Members
Varun Govind (varung2) varung2@illinois.edu
Vyom Thakkar (vnt2) vnt2@illinois.edu

1 Introduction

In this project, we aim to create a Generative-Adversarial-Network (GAN) to augment existing chest X-Ray images of patients lungs with different preexisting conditions to aid with the detection of COVID-19, and additionally create a classifier that is able to classify the X-Rays into the patients conditions (COVID-19, Pneumonia, Normal, etc.). The final goal of this project is to be able to augment an existing X-Ray dataset with the GAN and additionally reliably detect if a patient has COVID-19 (or some other condition) or if he is healthy by looking at his/her chest X-Rays.

Currently the world is still suffering from the hold of COVID-19 and while medicine has advanced considerably with the creation and distribution of vaccines and testing kits, people are still getting infected and are getting hospitalized due to extreme symptoms. Additionally, while vaccines are being distributed and the rate of infections are reducing, it is still unclear whether COVID-19 will be eradicated or if it will remain alongside us (humans) after the fact. This project would help validate results from the paper and perhaps push new directions in helping detect the virus.

2 Member Roles and Strategy

Our code is being shared by Google Colab and by GitHub. We meet on weekends to discuss project status and goals/deliverables for the next week, any issues that may have arisen, and the total project plan. The data is being shared through google drive. Both of us are working closely for data cleaning, model architecture, and training. However, the responsibilities can be broadly split into the following items:

1. **Varun Govind:** Data cleaning, Data retrieval, Model architecture for GAN, Training GAN.
2. **Vyom Thakkar:** Model architecture for Classifier, Training Classifier.
3. **Both Members:** Performance/Result Analysis, Report Write-up and Evaluations.

For the purposes of this midterm report, nothing has changed significantly although the member roles for training specific model architectures have been swapped (the changes have been appended above).

3 Design Approach

For our machine learning model architecture, we follow in the footsteps of Khalifa M. [4]. The paper describes a generator architecture and discriminator architecture in which they use to synthetically boost their dataset to train their classifier. Although we follow some parts of the architecture, we ultimately take a slightly different approach to the process. In our design, we are creating a Conditional GAN. The conditional GAN will essentially take in a label prior along with some random noise so that the image it is trying to generate can specifically create what we want. In the reference paper ([4]), they only use the GAN to generate X-Ray images of patients with COVID-19, however, in our approach, we want to create a GAN that can create X-Ray images of patients with and without COVID-19. The solution is to train a GAN that takes in labels and noise as input and outputs an image with respect to the label. It is simple in theory but difficult to train in practice, as we will show later.

For some background, GANs are typically constructed with two main parts: a generator and a discriminator. The generator's purpose is to create a 'fake' object (typically an image but can be something else in a different problem domain). The discriminator's purpose is to determine if the image given to it is fake or real. We construct a loss out of the discriminator on whether it was right or wrong and pass that to our generator. So the idea is to create an adversary for the generator in which the generator tries to fool. So when the discriminator gets better at distinguishing real images versus fake images, the generator has to improve to fool the discriminator. At some point, the discriminator has trouble distinguishing between real and fake and more training needs to be done for the discriminator. This process goes back and forth until we get a model that can produce something realistic (according to what training data was given). This form of unsupervised learning forces the model to learn what the distribution is of the training data that is given. In our specific case, we want our model to learn the specific patterns, features, and differences between X-Ray chest scans of patients with and without COVID-19.

3.1 Generator Architecture

For our architecture design, we follow the deep convolution layers closely but since we are creating a Conditional GAN, some of the layers at the beginning will vary. For our generator architecture, we use an input embedding layer for the labels and a dense layer for the image. We then concatenate the output of those two layers channel wise and input them into our special convolution layers. Note that our special convolution layers can be decomposed into simpler blocks referred to as **conv_layer_set**. We then have a final set of layers, which is similar to the **conv_layer_set** but it omits the 2D batch normalization and has a Tanh as the final activation function. As you can see in the figure below (Figure 1), it is a relatively straight forward architecture. For the number of filters, we use 64 filters and a square filter width of 4 pixels. Note that we use 2D convolution transpose which serves to upsample the input by convolving with weights. Typical 2D convolution will keep the same shape or reduce it.

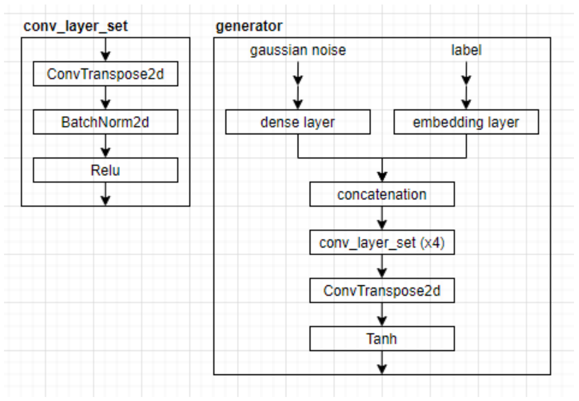

Figure 1: Generator Architecture

3.2 Discriminator Architecture

In our discriminator design, it is very similar to our generator architecture, with some subtle differences. We refer to the same special convolution layer as **conv_layer_set** but it is different than the one mentioned in the generator architecture as it contains two different layers (Figure 2). The first difference is that we don't use 2D convolution transpose and the second is the activation function after the 2D batch normalization; we use 'LeakyReLU' as an activation.

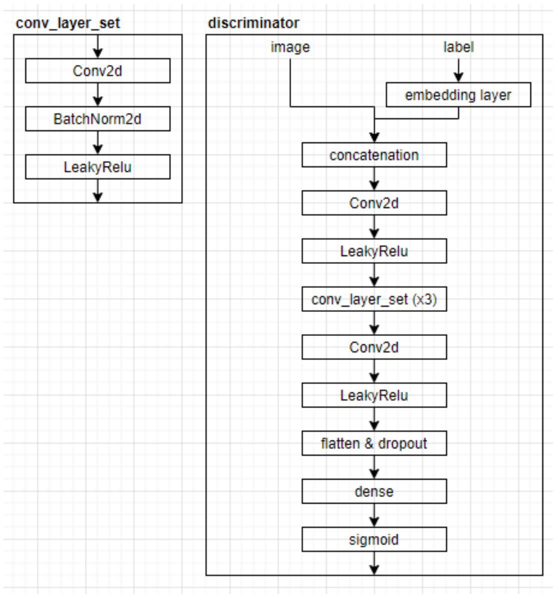

Figure 2: Discriminator Architecture

In our discriminator, we take our image and concatenate it with our embedding layer which takes as input, our label. We initially pas our concatenated layer into a 2D convolution layer with a 'LeakyReLU' activation function. Notice that we omitted the batch normalization after the convolution layer. This is intentional. For our next set of layers, we use our **conv_layer_set** three times and follow it with another convolution and 'LeakyReLU' (again without batch normalization). This ends our convolution layer and leads to our fully connected layer. We perform a flattening of the layer and dropout some weights (at a probability of 50%) and follow it with a dense layer and a sigmoid activation function. The sigmoid's activation function is to serve as the final step in determining the likelihood of the input image being real or fake.

As you can see from our figure above, our discriminator is slightly more complicated than our generator but is essentially constructed of the same components. Note that we don't use the same 2D convolution as the

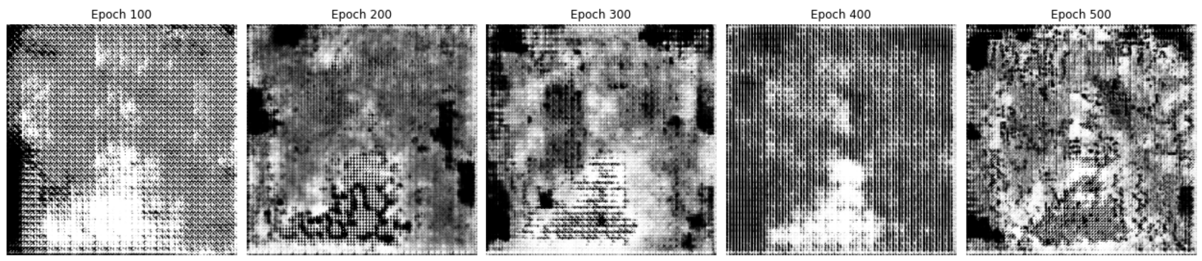

Figure 3: GAN image generation per epoch of training

generator; we use the normal 2D convolution layer and not the 2D convolution transpose.

4 Procedure and Results

Our current procedure for training the GAN involves running (and re-running) a training loop within Google Colab. Since we are personally using laptops, we can leverage Google Colab's free GPU to train our model with some restrictions. Since Google Colab kicks of the user if the user is idle of from a usage limit, a framework is needed to reliably save the model at a checkpoint after training is done in order to preserve the latest model. To do this, a wrapper class is created which contains the model we want to train and contains some special utilities for logging and tracking the training process. It additionally contains the framework to save and load a model from its latest checkpoint in order to resume training from where it last left off. This allows us to reliably train our model without worrying about loosing progress.

4.1 GAN Training Procedure

Our training procedure for our GAN is pretty straightforward. We split our training data set into 2500 images of patients with COVID-19 and 2500 more images of patients without; this gives us a total of 5000 training images to train on. For data transformation, we convert the images to a resolution of (300, 300) and normalize the values such that they are between -1 and 1. We then train with a batch size of 4. For our training parameters, we use and Adam optimizer with a learning rate of 2e-4 and a momentum (beta1) of 0.5. An interesting thing that we do to help the GAN converge is to train the discriminator every 10 training steps that we train the generator. This helps prevent the discriminator from learning to fast which causes the generator to diverge and create bad and noisy results.

4.2 GAN Initial Results

Our initial results are not very good. We are hoping that with more training we will be able to converge to a good result but for now, even with 500 epochs of training a set of 5000 images leads to the results shown in the figure above (Figure 3). You can see that the generator is somewhat attempting to learn the distribution of the chest and lung. You can faintly see the white figure, the dark spots of the lungs, and the stomach and liver organs at the bottom. However the results after training for so long is still not very good. We plan on continuing training and seeing where it goes but it is possible that we may need to retrain the model with new different hyperparameters in order to get better results.

5 Resources, Data, and References

For our data we are using several existing datasets which can be found in the respective links in the **References** section from GitHub and Kaggle [1, 3, 2, 5, 4].We may add more data sets if necessary as we make progress on the project. We are currently using the pytorch python libraries as a machine learning framework for our GAN architecture. We have not yet decided what machine learning framework our classification architecture will be implemented on but as it is using python, it will be trivial to integrate into our project. We are not currently using any external code, but that may change in the future due to time constraints. Recall, our goal is to build, train, and test the model from scratch.

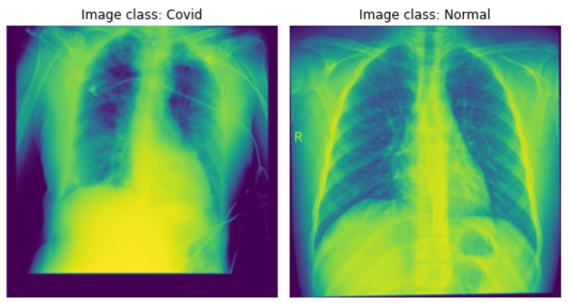

Figure 4: Sample of two patients' chest x-rays with different conditions

An example of our dataset with some labels can be shown in the figure above (Figure 4). This specific set of images comes from the *COVID-19 Radiography Database* ([3]). The set consists of 3616 COVID-19 images, 10k of Normal images, and many more images (which are not relevant at the moment - they consist of viral pneumonia and lung opacity scans). Our data is of dimension (299, 299, 1) ordered by height, width, and channel. Our image is shown with a color map but they are essentially grayscale images.

One interesting thing to note is that a short look at the images can give a quick conclusion that if the patients lungs are cloudy, then the patient has some abnormal lung condition. A quick comparison to the normal image shows that a normal patient's lungs look clear. However, it should be stated (even if obvious) that cloudy lungs does not equate that the patients have COVID-19 but rather that it could be a different underlying condition.

6 Reservations & Concerns

Currently, we are having issues training the GAN as it is not converging and it has been trained on a significant number of epochs. It is critical to ensure that we can effectively augment the current dataset and improve the model performance. If we are not able to learn the distribution of the dataset effectively, then our model performance on a generalized dataset would not be strong. The "minimum goal" of our project is to estimate the distribution of the dataset as closely as possible, and augment the data to an extent that meaningful patterns can be recognized by our model for prediction. It is unclear whether that goal will be met.

Another concern is that our dataset (Kaggle - [2]) may not be accurate as there are some images that are flipped which can cause issues when training the GAN. In addition, some of the data can be corrupted; corrupted in the sense that the images may not be of someone with COVID-19 but rather some other underlying condition. A similar concern was made on the forums in which the dataset was located. Moreover, the data may not be an accurate set which people can use to train their models. This issue was also raised in the same forum.

7 Relationship to our Background

Both of us have had some prior experience with building deep learning models such as CNNs. Although we have a theoretical understanding of generative models and GANs, we have not implemented such generative models in practice. While one of us has some prior experience with the pytorch framework, the other person has some prior experience in working with pytorch and tensorflow frameworks.

References

[1] J. P. Cohen, P. Morrison, L. Dao, K. Roth, T. Q. Duong, M. Ghassemi, Covid-19 image data collection: Prospective predictions are the future, arXiv 2006.11988.
URL `https://github.com/ieee8023/covid-chestxray-dataset`

[2] Kaggle, Chest x-rays datasets compilation (2021).
URL `https://www.kaggle.com/c/vinbigdata-chest-xray-abnormalities-detection/discussion/208035`

[3] Kaggle, Covid-19 radiography dataset (2021).
URL `https://www.kaggle.com/tawsifurrahman/covid19-radiography-database`

[4] N. E. M. Khalifa, M. H. N. Taha, A. E. Hassanien, S. Elghamrawy, Detection of coronavirus (covid-19) associated pneumonia based on generative adversarial networks and a fine-tuned deep transfer learning model using chest x-ray dataset (2020).

[5] M. Loey, F. Smarandache, N. E. M. Khalifa, Within the lack of chest covid-19 x-ray dataset: A novel detection model based on gan and deep transfer learning, Symmetry 12 (4).
URL `https://www.mdpi.com/2073-8994/12/4/651`