CS551G - Assessment 1

Jonathan Cauchi

March 2021

Task 1: Create Synthetic Covid-19 X-Ray Images with Conditional Generative Adversarial Networks

Synthetic Image Generation

For this task, using the Covid-19 dataset (100 images) and a cGAN model, we will generate synthetic images. We will create a script that will preprocesses the data, train the cGAN model, generate synthetic images and save them to a folder. The first step was to obtain the path of the Covid-19 pneumonia chest X-ray images using the operating system standard library. The images are converted into gray-scale using *OpenCV* to save computational resources. Some of the images are visualized to get an intuition of how the classification will take place. Similar to a radiologist, the model will learn to look for white spots in the lungs that identify a viral presence.

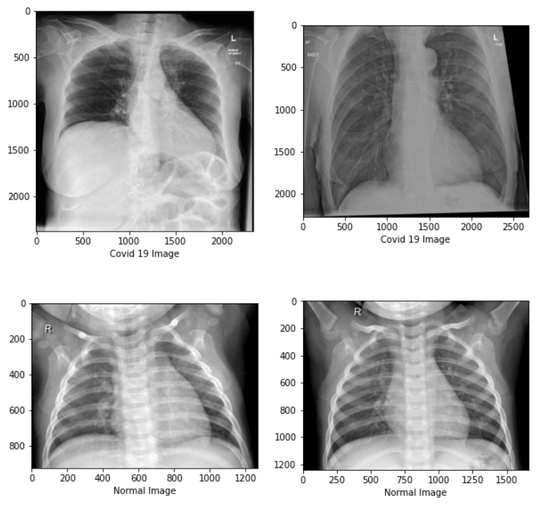

Figure 1: Comparison between Covid-19 and Healthy chest x-rays

Data Augmentation and Image Generation Techniques

The images were reshaped to a size of ***446 x 446***. The value for each pixel was normalized to a value between 0 and 1. Image data augmentation could be used to create alternative versions of existing images to expand the dataset's overall size and variance. However, for this project, data augmentation was not implemented.

As a generation technique, Generative Adversarial Networks (GANs) have shown exciting performance, especially in the healthcare domain. The main objective in GANs is in the form of a zero-sum game, in which the discriminator tries to classify between real and fake images, and the generator tries to trick the discriminator by generating fake images that the discriminator will classify as authentic. In cGANs, a conditional setting is applied, meaning that both the generator and discriminator are conditioned on some sort of auxiliary information, which in our case are two vector of labels (0's and 1's of length batch size).

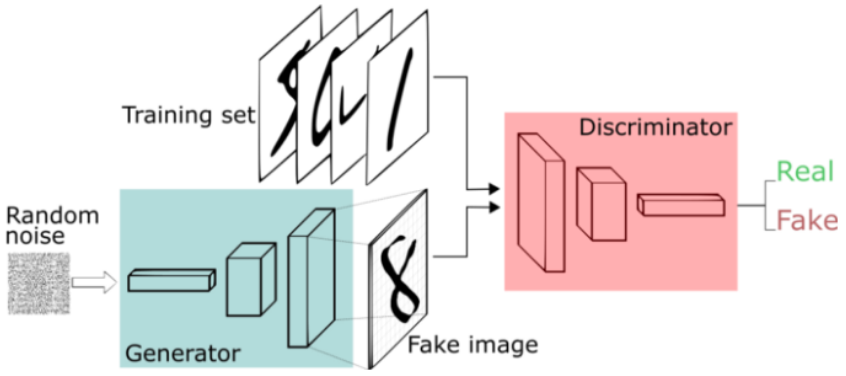

Figure 2: CGAN Architecture - Example

Image Generation

The main benefit of using cGANs is that it minimizes the consequences of data scarcity. Using image generation and augmentation techniques, it is possible to increase the volume and variance of a dataset to train impactful deep learning models for various tasks. For this project, image generation was successful by training the generator on 100 Covid-19 images. For the generator and discriminator, the architectures were based on the work done by Georgios Leontidis and Stefanos Karakansis (2021) [1].

- **Generator Architecture:**

The discriminator is a sequential feed-forward neural network that takes an image of 446 x 446 resolution as input and outputs a binary outcome (real or fake). The discriminator is composed of four dense layers and an output layer. For each dense layer, dropout regularization is implemented to avoid over-fitting. The images are flattened into a 1-D vector of length/size image height multiplied by image width (446 x 446 = 198916). ReLU activation functions were applied to all dense layers except for the output layer, which implements a sigmoid activation function, given its a binary classification problem.

Not much experimentation has been done on the number of layers, which was borrowed from a paper published in a high impact journal. Regarding the number of nodes for the hidden layers, there are methods of obtaining the optimal values for these parameters, which could be random (trial and error) or via heuristic techniques (e.g. genetic algorithms). However, not much experimentation has been performed on this part.

- **Discriminator Architecture:**

  The generator consists of a neural network that receives a latent space, which is a vector consisting of 100 random numbers as input. The architecture consists of four dense layers and an output layer. The first four layers all implement Leaky ReLU as activation functions with an alpha value of 0.2. For all dense layers, except the output layer, batch normalization is applied with a momentum value of 0.8.

  Similar to the discriminator, no systematic experimentation took place to calculate the optimal value of each layer's number of nodes. The output layer results in a 1-D vector of 196, with a hyperbolic tangent (*tanh*) activation function. Adam optimizer was implemented, which is a form of stochastic gradient descent. The learning rate was set at 0.002 and p at 0.5. The number of epochs was experimented with, varying from 4000 to 10000 epochs.

Visualizing Synthetic Images and Evaluation

During the training of the cGAN model, images were saved to a folder after some specified epochs. The image quality was not evaluated with the use of discriminator metrics but from a visual exam. In an ideal experimental scenario, a human expert radiologist would examine the quality of the synthetic images. Below there are visualizations of images generated by the generator.

Hyperparameter Tuning

Generative Adverserial Networks are notoriously difficult to train. The reason for this challenge comes down to the fact that both the generator and discrimi-

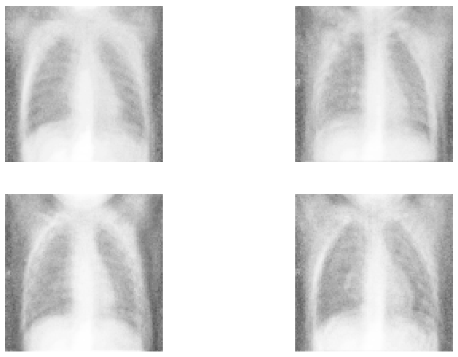

Figure 3: Synthetic Images 1

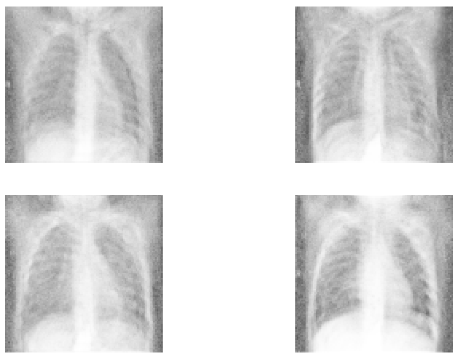

Figure 4: Synthetic Images 2

nator models are trained simultaneously. This means that improvements to one model come at the expense of the other model. The optimal solution would be finding a balance between the two competing parties

For this task, we will generate synthetic images based on three different hyper-parameters, which are different learning rates, epochs, batch size, etc. We will show the generated images differ according to epochs and different learning rates. The generator was implemented with the Adam optimizer, the learning rates were experimented with value of 0.0002 & 0.01. Here are the results generated at epochs 100 & 200.

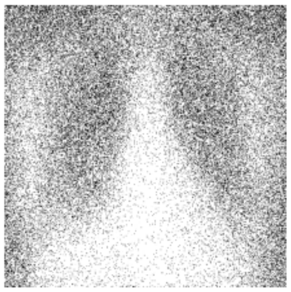

Figure 5: Epoch 100

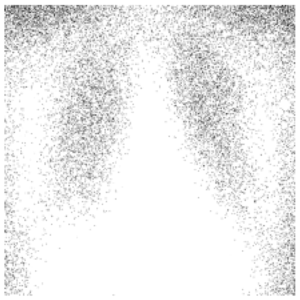

Figure 6: Epoch 200

Figure 7: Synthetic Images at epoch 100 & 200 with learning rate *0.01*

The images generated with a learning rate of *0.01*, which is the default learning rate for the Adam optimizer, resemble some degree a Covid-19 chest x-ray. The images generated with learning rate *0.002* are still very noisy, with little to no resemblance to an actual Covid-19 chest x-ray. Although the higher learning rate shows more significant promise earlier on, this might not be the case later, as it might fail to converge at optimal local optima. Before performing any hyper-parameter tuning, it is necessary to research what approaches can be undertaken.

According to Radford et al.(2015) [2], it is recommended that Adam optimizer is used with a learning rate of 0.0002. It was found that 0.001 was too high. Also, It was found that a momentum value of 0.9 resulted in training instability, which lead to a momentum reduction of 0.5 that performed better. In our model, we follow Leontidis and Karakansis approach by applying a momentum value of 0.8. However, a learning rate of 0.0001 was applied for the generator and discriminator.

ReLU is recommendable for the generator but not for the discriminator. For the discriminator, LeakyReLU is advised, which is a variant of ReLU. Ideally, the generator will use the hyperbolic tangent function (tanh) for the output layer, which inputs data to the discriminator scaled from -1 to 1. It is also recommended that the slope of the LeakyReLU is set to 0.2.

It is also recommended to use batch normalization, which normalizes the activation from a previous layer to have zero mean and unit variance. This allows for the stabilization of the training process. It is recommended to be applied to each dense layers of both the generator and discriminator except for the output of the generator and the input of the discriminator. It is studied that applying batch normalization to all layers will result in sample oscillation and model instability. **50 images were generated at epoch 4000 and saved to a folder for later use.**

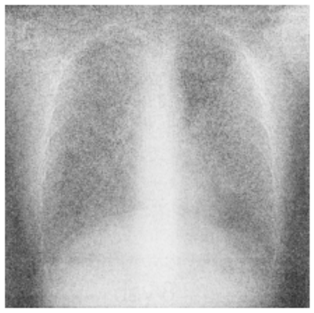
Figure 8: Fake Image 1

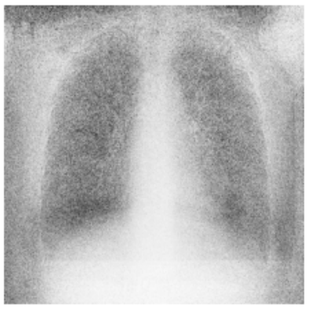
Figure 9: Fake Image 2

Detecting Covid-19 from Chest X-Ray Images using pre-trained networks

ResNet 50

ResNet is a neural network used for many computer vision tasks. The ResNet50 consists of 5 stages, each with a convolution and identity block. Each convolution has 3 convolution layers and each identity block also has 3 convolution layers. The model has been tested on the original dataset which consists 100 Covid-19 images and 100 Normal images.

The model was also applied to the same dataset, with 50 of the Covid-19 images generated by the cGAN model. The weights were set as *ImageNet* which are the optimized weights used for the ImageNet competition. The pooling was set at *avg*, which means that global average pooling will be applied to the output of the last convolutional block, and thus the output of the model will be a 2D tensor.

Data Augmentation

The images gone through the following process prior to model training:

- Horizontal flipping
- Shearing
- Rescaling

Hyperparameter Tuning

For the ResNet model, the last layer was frozen and three dense layers were added. The output of the ResNet model was flattened and normalized. The first of the added dense layers included a ReLU activation function, batch normalization and a dropout of 0.4. The final layer includes a sigmoid activation function, given its a binary classification problem. The optimizer is SGD with a learning rate of 0.001, and a batch size of 25.

Prior, the model was trained on a learning rate of 0.01, batch size of 32, three dense layers after ResNet and dropout values of 0.2. The model had high accuracy on the train set and the accuracy on the test set remained unchanged (0.5 throughout). The conclusion was that the model got stuck in a undesired local optima, therefore, the learning rate was changed from 0.01 to 0.001 as well as the batch size from 32 to 25. Here, the model improved, but still it was performing badly on the test set (over-fitting). Finally, the dropout was increased to 0.4 and one of the added dense layers was removed.

Model Evaluation

The model was evaluated on the test set. The train-test split was set at 80:20. The number of epochs is set at 100. The model performed consistently across the whole training process and achieved high accuracy rates across train and test sets for the original dataset.

For the synthetic dataset (50 fake images), the same model was fit to the dataset with some modifications. The learning rate remained unchanged, however, the batch size was decreased to 16. In **Figure 12 & 13**, initially, the model starts performing poorly on the test set and then starts to constantly achieve good performance throughout the iterations. The test accuracy and loss are better than the training set accuracy and loss due to the small size of the test set.

Resources (Links)

- **Code/Notebook - Google Colab**
- **Generated Images - Google Drive**

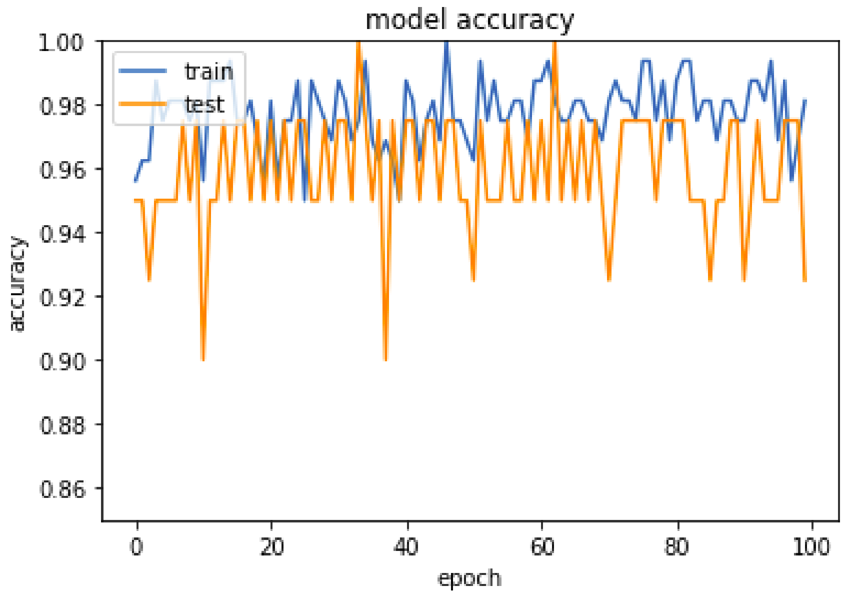

Figure 10: Model Accuracy - Original Dataset

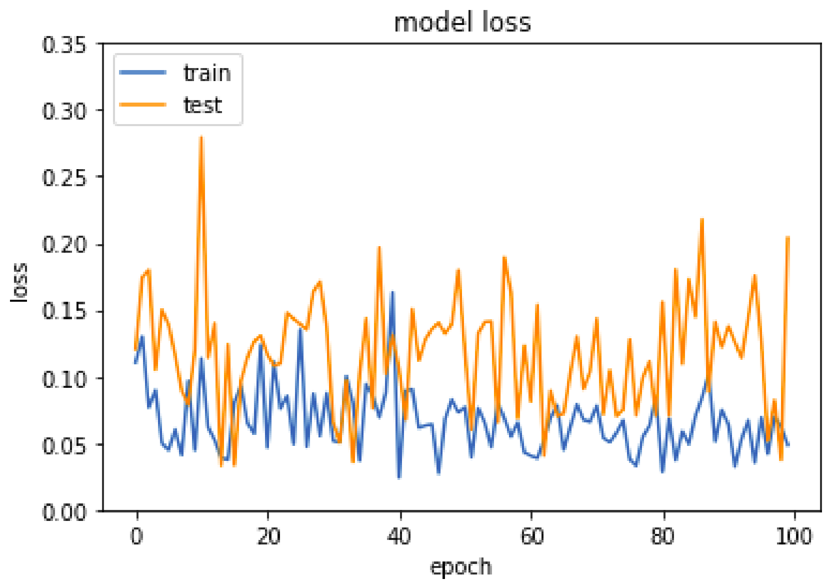

Figure 11: Model Loss - Original Dataset

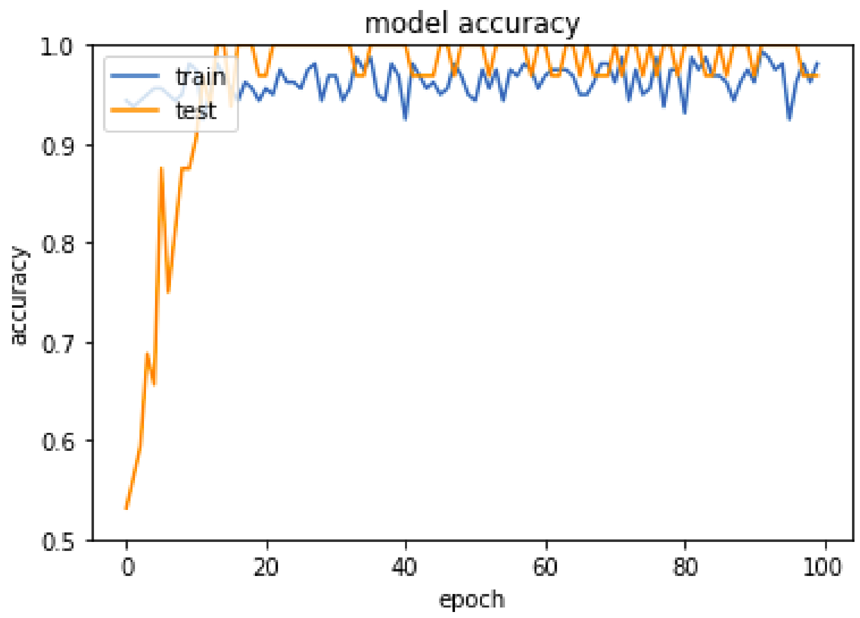

Figure 12: Model Accuracy - Synthetic Dataset

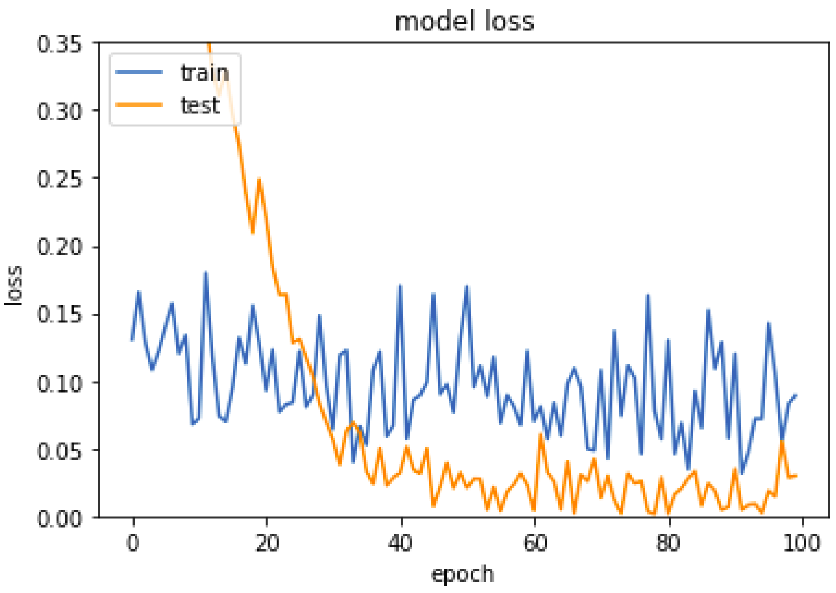

Figure 13: Model Loss - Synthetic Dataset

References

[1] Karakanis, Stefanos Leontidis, Georgios. (2021). Lightweight deep learning models for detecting COVID-19 from chest X-ray images. Computers in Biology and Medicine. 130. 104181. 10.1016/j.compbiomed.2020.104181.

[2] Radford, Alec Metz, Luke Chintala, Soumith. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.