Synthetic of Chest X-ray Images

Bita Davoodi davoodibita@gmail.com
Manuela Lo Giudice Manuela99lg@gmail.com

1. Model Description

Dealing with small datasets and a limited number of annotated samples is a problem in the medical imaging domain, especially when employing supervised machine learning algorithms that require labeled data and larger training examples. Although public medical datasets are available online, most datasets are still limited in size and only applicable to specific medical problems. The reason is that collecting medical data is a complex and expensive procedure and researchers attempt to overcome this challenge by using data augmentation. The most classical data augmentation methods include simple modifications of dataset images such as translation, rotation, flip, and scale, which is a standard procedure in computer vision tasks. This is because little additional information can be gained from small modifications to the images (i.e. the translation of the image a few pixels to the right). Synthetic data augmentation of high-quality examples is a new, sophisticated type of data augmentation. Synthetic data examples learned using a generative model enable more variability and enrich the dataset to further improve the system training process. One such promising approach inspired by game theory for training a model that synthesizes images is known as Generative Adversarial Networks (GANs). The model consists of two networks that are trained in an adversarial process where one network generates fake images and the other network discriminates between real and fake images repeatedly.

2. Dataset

The models are trained on the covid chest X-ray dataset which is found at the Github link https://github.com/vj2050/Transfer-Learning-COVID-19. The dataset is well structured and contains 4 classes namely, covid, normal, pneumonia bacteria, and pneumonia viral. The dataset has a total of 270 training images (60, 70, 70, 70) and 36 test images (9, 9, 9, 9). We used the default train/test split proposed by the creators of the dataset.

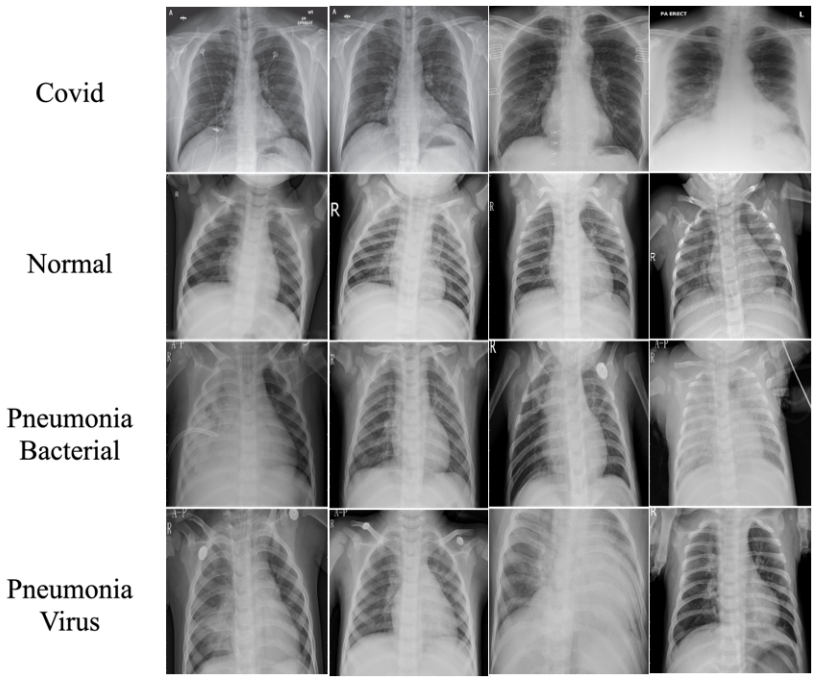


Figure 1: Samples taken from 'COVID-19 image data collection' test set.

3. Training procedure

3.1. Generator Architecture

The generator network takes a vector of 100 random numbers drawn from a uniform distribution as input and outputs a chest X-ray image of size $64 \times 64 \times 1$. The network architecture consists of a fully connected layer reshaped to size $4 \times 4 \times 512$ and four fractionally-strided convolutional layers to up-sample the image with a 4×4 kernel size. A fractionally-strided convolution can be interpreted as expanding the pixels by inserting zeros in between them. Batch-normalization is applied to each layer of the network, except for the output layer. Normalizing responses to have zero mean and unit variance over the entire mini-batch stabilizes the GAN learning process and prevents the generator from collapsing all samples to a single point. ReLU activation functions are applied to all layers except the output layer which uses a tanh activation function. The figure shows the generator network architecture along with the trainable parameters.

Table1: Generator architecture.

Layer (type:depth-idx)	Output Shape	Param #
├─Sequential: 1-1	[-1, 1, 64, 64]	--
│ └─ConvTranspose2d: 2-1	[-1, 512, 4, 4]	819,200
│ └─BatchNorm2d: 2-2	[-1, 512, 4, 4]	1,024
│ └─ReLU: 2-3	[-1, 512, 4, 4]	--
│ └─ConvTranspose2d: 2-4	[-1, 256, 8, 8]	2,097,152
│ └─BatchNorm2d: 2-5	[-1, 256, 8, 8]	512
│ └─ReLU: 2-6	[-1, 256, 8, 8]	--
│ └─ConvTranspose2d: 2-7	[-1, 128, 16, 16]	524,288
│ └─BatchNorm2d: 2-8	[-1, 128, 16, 16]	256
│ └─ReLU: 2-9	[-1, 128, 16, 16]	--
│ └─ConvTranspose2d: 2-10	[-1, 64, 32, 32]	131,072
│ └─BatchNorm2d: 2-11	[-1, 64, 32, 32]	128
│ └─ReLU: 2-12	[-1, 64, 32, 32]	--
│ └─ConvTranspose2d: 2-13	[-1, 1, 64, 64]	1,024
│ └─Tanh: 2-14	[-1, 1, 64, 64]	--

Total params: 3,574,656
Trainable params: 3,574,656
Non-trainable params: 0
Total mult-adds (M): 423.53

Input size (MB): 0.00
Forward/backward pass size (MB): 1.91
Params size (MB): 13.64
Estimated Total Size (MB): 15.54

3.2. Discriminator Architecture

The discriminator network takes the input image (chest X-ray) of size $64 \times 64 \times 1$, and outputs one decision: is this X-ray image real or fake? The architecture of the discriminator network is shown in Figure. The network consists of four convolution layers with a kernel size of 4×4 and a fully connected layer. Strided convolutions are applied to each convolution layer to reduce spatial dimensionality. Batch-normalization is applied to each layer of the network, except for the input and output layers. Leaky ReLU activation functions are applied to all layers except the output layer which uses the Sigmoid function for the likelihood probability (0, 1) score of the image. Also, dropout is applied to each layer of the network, except for the second and the output layers.

Table2: Discriminator architecture.

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─Sequential: 1-1                        [-1, 1, 1, 1]             --
|    └─Conv2d: 2-1                       [-1, 64, 32, 32]          1,024
|    └─LeakyReLU: 2-2                    [-1, 64, 32, 32]          --
|    └─Dropout: 2-3                      [-1, 64, 32, 32]          --
|    └─Conv2d: 2-4                       [-1, 128, 16, 16]         131,072
|    └─BatchNorm2d: 2-5                  [-1, 128, 16, 16]         256
|    └─LeakyReLU: 2-6                    [-1, 128, 16, 16]         --
|    └─Conv2d: 2-7                       [-1, 256, 8, 8]           524,288
|    └─BatchNorm2d: 2-8                  [-1, 256, 8, 8]           512
|    └─LeakyReLU: 2-9                    [-1, 256, 8, 8]           --
|    └─Dropout: 2-10                     [-1, 256, 8, 8]           --
|    └─Conv2d: 2-11                      [-1, 512, 4, 4]           2,097,152
|    └─BatchNorm2d: 2-12                 [-1, 512, 4, 4]           1,024
|    └─Dropout: 2-13                     [-1, 512, 4, 4]           --
|    └─LeakyReLU: 2-14                   [-1, 512, 4, 4]           --
|    └─Conv2d: 2-15                      [-1, 1, 1, 1]             8,192
|    └─Sigmoid: 2-16                     [-1, 1, 1, 1]             --
==========================================================================================
Total params: 2,763,520
Trainable params: 2,763,520
Non-trainable params: 0
Total mult-adds (M): 104.48
==========================================================================================
Input size (MB): 0.02
Forward/backward pass size (MB): 1.38
Params size (MB): 10.54
Estimated Total Size (MB): 11.93
==========================================================================================
```

3.3. Training

3.3.1. **GAN**

Batch Size: Considering the size of the dataset we chose to train our model on, we took a small batch size of 32. The batch size was quite suitable for our case, and we observed that both the discriminator as well as the generator could train with stability.

Hyperparameters: Four separate DCGANs were trained to synthesize chest X-ray images for each of the four classes of chest X-ray images, namely, covid, normal, pneumonia bacteria, & pneumonia virus. The training process was done iteratively for the generator and the discriminator. As mentioned earlier, we used mini-batches of $m = 32$. In the Leaky ReLU (Rectified Linear Unit), the slope of the leak was set to leak $= 0.2$. Weights were initialized to a zero-centered normal distribution with a standard deviation of 0.02. We applied stochastic gradient descent with the Adam optimizer, an adaptive moment estimation that incorporates the first and second moments of the gradients, controlled by parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$, respectively. A learning rate of 0.0002 was kept for both the generator and the discriminator.

3.3.2. **Classifier**

Batch Size: The strategies we used operate in a small-batch regime where in the set of the training data, a subset, namely 256 elements, is randomly sampled to compute an approximation to the gradient. This is because it has been observed in practice that when using a larger batch there is a significant degradation in the quality of the model, as measured by its ability to generalize.

Hyperparameters: For training, we used a batch size of 256. We used stochastic gradient descent optimization with a high learning rate (lr $= 0.01$), which performed very

well for this specific task, and Nesterov momentum updates ($m = 0.9$), where instead of evaluating the gradient at the current position we evaluated it at the "lookahead" position which improves the optimization and speed up the process. Separates CNN were trained on different augmented chest X-ray scan types for each of the available datasets for variable epochs quantities depending on the dataset dimensions, between 150 epochs for small datasets to 300 epochs for larger ones.

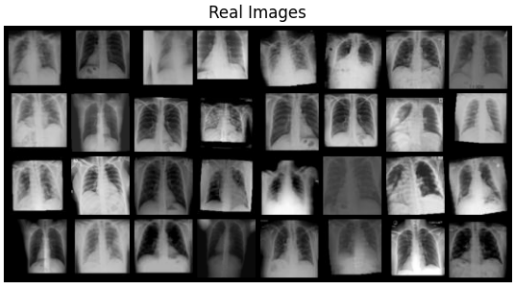
3.4. Evaluation Metrics

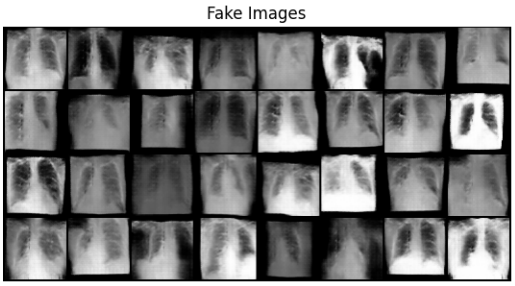
3.4.1. GAN

In order to evaluate the performance of a GAN, we record the following training statistics:

- **Loss_D**: This represents the discriminator loss calculated as the sum of losses for all the real and all the fake batches ($\log(D(x)) + \log(D(G(z)))$).
- **Loss_G**: It is the generator loss calculated as $\log(D(G(z)))$.
- **D(x)**: It represents the average output of the discriminator for all the real batches. This should start close to 1 and then theoretically converge to 0.5 when G gets better.
- **D(G(z))**: This is the average discriminator output for the all-fake batch. The first number is before D is updated (D_G_z1) and the second number (D_G_z2) is after D is updated. These numbers should start at 0 and converge to 0.5 as G gets better.
- **FID**: Calculates Fréchet inception distance (FID) which is used to assess the quality of generated images.
- **IS**: Calculate the Inception Score (IS) which is used to assess how realistic generated images are.

Epoch 200 100%— 34/34 [00:10¡00:00, 3.37it/s] [200/200][33/34] Loss_D: 1.2568 Loss_G: 3.0651 D(x): 0.7847 D(G(z)): 0.5819 / 0.0646 FID: 1.1241259574890137 InceptionScore: 2.7996814250946045





3.4.2. Classifier

We calculated confusion matrices, sensitivity, specificity, accuracy, and F1 score measures for each lesion category in order to determine the classifier performance.

3.4.3. **Results**

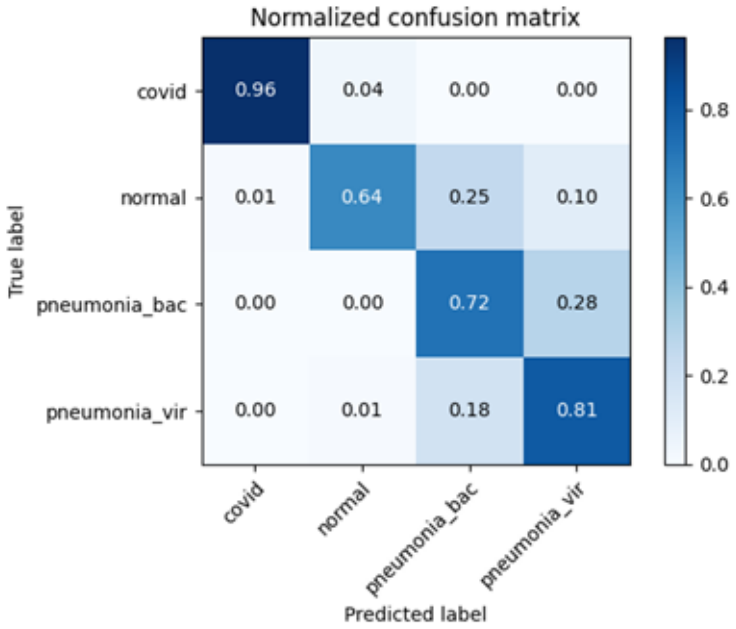


Table 4 (a): Classic train set, 300 epochs, 0.780 total accuracy.

4-fold_300_ep_0.780_acc	Accuracy	Sensitivity	Specificity	F1Score
covid	0.988	0.962	0.997	0.978
normal	0.898	0.635	0.985	0.767
pneumonia_bac	0.82	0.716	0.855	0.776
pneumonia_vir	0.855	0.808	0.87	0.835

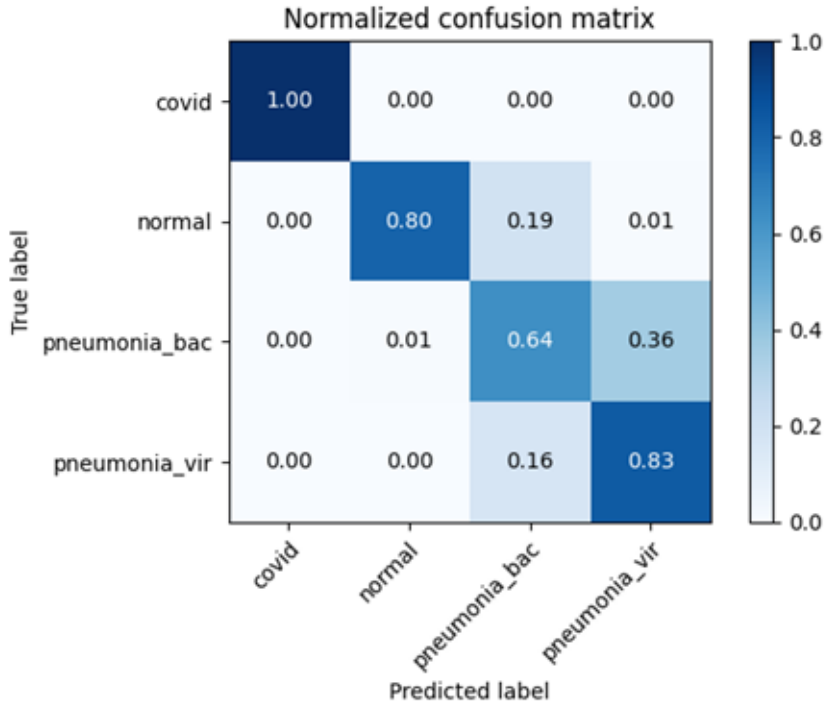


Table 4 (b): Synthetic augmented train set, 300 epochs, 0.818 total accuracy.

4-fold_300_ep_0.818_acc	Accuracy	Sensitivity	Specificity	F1Score
covid	1	1	1	1
normal	0.948	0.802	0.997	0.888
pneumonia_bac	0.82	0.636	0.882	0.733
pneumonia_vir	0.867	0.833	0.878	0.852

4. Conclusion

The number of images of the collected dataset was 270 images for four types of classes. The classes are the covid, normal, pneumonia bacteria, and pneumonia virus. The task of classification and detection of the four classes is quite challenging in the absence of a large dataset. It was shown how the generated synthetic images could be used to augment the original dataset and eventually lead to a higher classification and detection accuracy of the chest X-ray images.