COVID-19 Classification with CNNs and GANs Data Augmentation

Ruchika Gupta Karn Jongnarangsin Yashashvini Rachamallu

Abstract

The COVID-19 pandemic [4] has brought attention to the necessity of fast and effective diagnostics in critical health care. Using Deep Learning algorithms, such as Convolutional Neural Networks (CNNs), has shown promising outcomes for accurate, automated diagnosis. However, amassing a large number of samples to train on may not be feasible given the nature of epidemics. The ability to train reliable models on smaller datasets therefore proves to be an efficient alternative for these time-sensitive and critical situations, such as the COVID-19 pandemic. In this project, we utilize generative methods to augment small datasets while also comparing the efficiency of pre-existing neural network architectures trained using synthesized data.

1. Introduction

Covid-19 has been a global concern since 2019, crippling the world economy and health. Biological tools have been used to identify the presence of the virus in body fluids. The standard diagnostic test for COVID-19 detection is RT-PCR. However, there are several difficulties associated with this such as the concerns of having access to the COVID-19 kit based on your geographical location. As can be seen, it is important to have another strategy in case one fails. The COVID-19 virus causes pneumonia which results in inflammation of the lungs. This allows for COVID-19 detection by analyzing chest MRI scans and observing COVID-19-induced pneumonia. Promising results have been achieved in the literature in detecting COVID using medical images like CT scans and X-rays using supervised artificial neural network algorithms. One of the biggest drawbacks of supervised learning models is that they require enormous amounts of data to learn from, such that they can accurately represent underlying relationships. While there are extensive public data sets for typical chest X-rays, there is a noticeable lack of sufficiently large, labeled, and relevant datasets.

Despite the availability of public medical datasets online, their size is often restricted, and they apply to specific medical issues. The inherent complexity and cost of collecting medical data contribute to this limitation. Researchers commonly attempt to overcome this challenge by resorting to data augmentation. Classic data augmentation methods involve basic modifications to dataset images, such as translation, rotation, flip, and scaling—standard procedures in computer vision tasks. However, the value gained from these small modifications is limited. A more sophisticated approach to data augmentation involves generating synthetic data examples using a generative model, offering increased variability to enrich the dataset and enhance the training process.

One promising method in this regard, inspired by game theory, is the use of Generative Adversarial Networks (GANs) [3]. The GANs are comprised of two networks engaged in an adversarial process: one generates fake images, and the other discriminates between real and fake images through iterative training. This synthetic data augmentation technique using GANs allows for the creation of high-quality examples, introducing more variability and improving the overall system training process.

This paper aims to address the concern of model training using small datasets by generating synthetic images to augment the training dataset. To evaluate this method, the testing accuracy of seven well-known deep-learning architectures will be assessed when trained using data augmented by traditional methods and with synthetic images respectively. This will provide insight as to whether augmentation using generative methods can prove to be a viable strategy when compared to traditional augmentation.

2. Dataset

The dataset utilized in this project for COVID-19 comprises images from a Kaggle dataset[7]. It includes approximately 251 MRI scans for training and 66 images for testing, categorized into three classes: Normal, Viral Pneumonia, and COVID-19. The distribution of classes in the dataset is crucial for understanding the balance or potential imbalances in the training data. Figure 1, helps in understanding the distribution of classes.

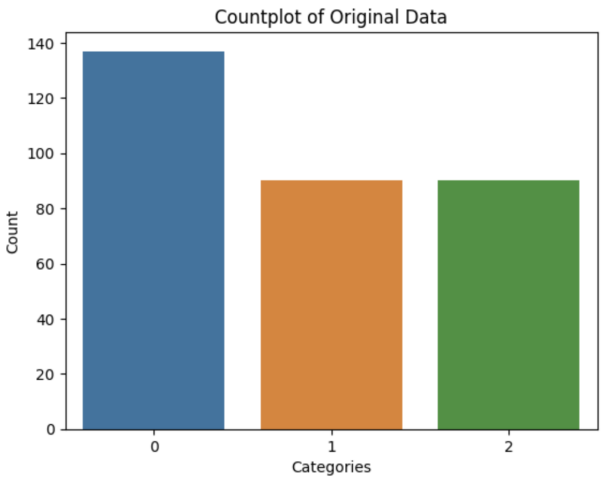

Figure 1. Count plot of Training Data

3. Methodology

3.1. Traditional Augumentation

Due to the relatively small sample size, we initially employed traditional augmentation techniques. These involved applying horizontal flips, translations, rotations, and scaling methods, resulting in an expansion of the training samples to 1255. Given the absence of validation data, we proceeded to split the augmented dataset into training and validation sets. Consequently, we finalized the dataset with 1000 images for training, 255 for validation, and 66 for testing.
Below we can visualize the count plot [Figure 2.] for the augmented data. It shows how many instances we have for each category. This visual helps us understand the distribution of classes in the dataset.

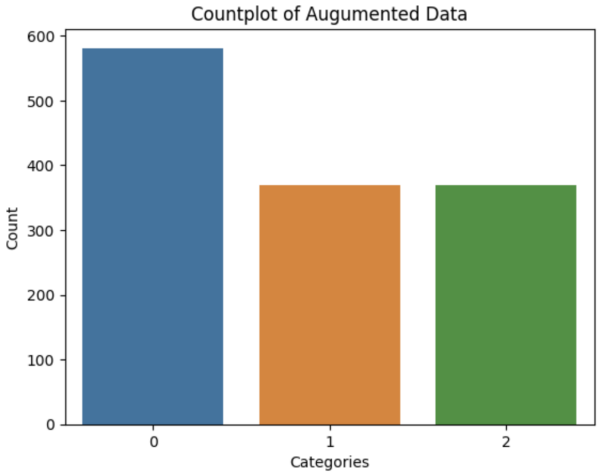

Figure 2. Count plot of Traditional Augmented Dataset

In the plot above, 0 indicates membership in the 'Normal' class, 1 corresponds to 'Covid,' and 2 represents 'Viral Pneumonia.' Below are the sample images for each class:

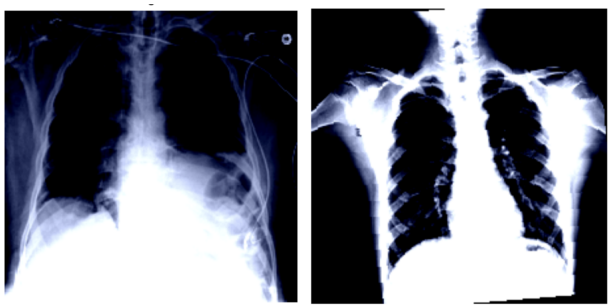

Figure 3. Sample of Normal, Covid Class

Figure 4. Sample of Viral Pneumonia Class

3.2. Data Augmentation Using GANs

Within the domain of deep learning, the pursuit of robust and high-performing models is frequently impeded by the limited availability of diverse training data. This scarcity is especially conspicuous in medical imaging tasks, where the acquisition of labeled samples can be a resource-intensive undertaking. Conventionally, we resort to traditional techniques such as scaling, rotating, and flipping to augment dataset size. However, a novel approach is explored here—one that involves generating images tailored to our dataset. This innovative technique aims to mitigate the risk of underfitting in our final model. In this endeavor, Conditional Generative Adversarial Networks (cGANs) [6] play a crucial role, revolutionizing the augmentation process by synthesizing contextually relevant images that contribute to a more comprehensive and nuanced dataset. The cGAN framework, renowned for its capacity to generate realistic data, has the potential to expand the dataset. The Conditional Generator with the cGAN, driven by carefully defined parameters, takes center stage in the augmentation endeavor.
The Conditional Generative Adversarial Network integrated Conditional Generator and Conditional Discriminator to address challenges in generating realistic synthetic images for enhancing dataset diversity. The ConditionalGenerator class is designed as a sequential Pytorch module, featuring layers like linear transformations, batch normalization, and transposed convolutional operations. This generator, influ-

enced by a latent vector and a conditional input, endeavors to produce synthetic images that seamlessly blend into the existing dataset. On the other side, the Conditional Discriminator assesses the authenticity of the images within the context of given conditions.This involves a convolutional network with leakyRelu activations, dropout layers, and linear connections in a sigmoid activation for probability scoring. The training incorporates Adam optimizers for both the generator and discriminator, with Binary Cross Entropy loss as the metric. The architecture of the Generator is as follows. [Figure 5]

```
ConditionalGenerator(
  (model): Sequential(
    (0): Linear(in_features=103, out_features=16384, bias=True)
    (1): BatchNorm1d(16384, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
    (3): Unflatten(dim=1, unflattened_size=(1024, 4, 4))
    (4): ConvTranspose2d(1024, 512, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
    (5): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): LeakyReLU(negative_slope=0.01)
    (7): ConvTranspose2d(512, 256, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
    (8): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): LeakyReLU(negative_slope=0.01)
    (10): ConvTranspose2d(256, 128, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
    (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (12): LeakyReLU(negative_slope=0.01)
    (13): ConvTranspose2d(128, 64, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
    (14): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (15): LeakyReLU(negative_slope=0.01)
    (16): ConvTranspose2d(64, 1, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2), output_padding=(1, 1))
    (17): Tanh()
  )
)
```

Figure 5. Generator Architecture

The training loop for the CGAN is structured to iterate 500 epochs, with a detailed process for optimizing the discriminator and generator at each step. Real images and corresponding labels are fetched from the dataset, and the discriminator is trained to distinguish between real and synthetic images. Subsequently, the generator is optimized to produce images that deceive the discriminator. Both the generator and discriminator losses are tracked throughout the training, providing insights into the model's performance. During the initial stage of the model, specifically in epoch-1, the generator produces its preliminary set of images, denoted as Figure 6. These initial images serve as a starting point, reflecting the early synthesis capabilities of the model. Interestingly, as the training progresses and reaches the 500th epoch, the generator refines its image generation, resulting in Figure 7. Notably, Figure 7 illustrates a remarkable convergence towards the intricacies of the original dataset, showcasing the model's evolution and its ability to generate images that closely align with the complexities of the authentic data.

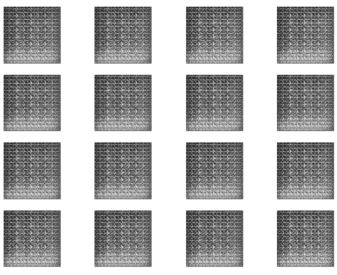

Figure 6. At epoch 0 Generated Images

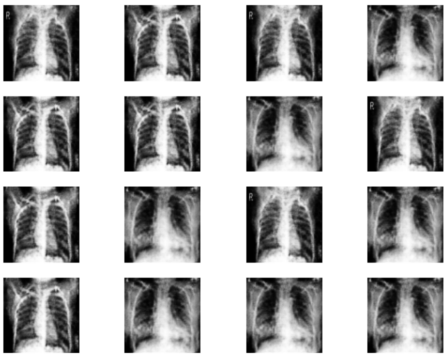

Figure 7. At epoch 500 Generated Images

Recognizing the satisfactory quality of the generated images, we proceeded to augment the dataset by generating an additional 300 images for each class, effectively expanding the dataset's size. The appended images were seamlessly integrated, contributing to a more comprehensive and diverse dataset. The subsequent count plot [Figure 8] visually encapsulates the distribution of instances across different classes within the enriched dataset.

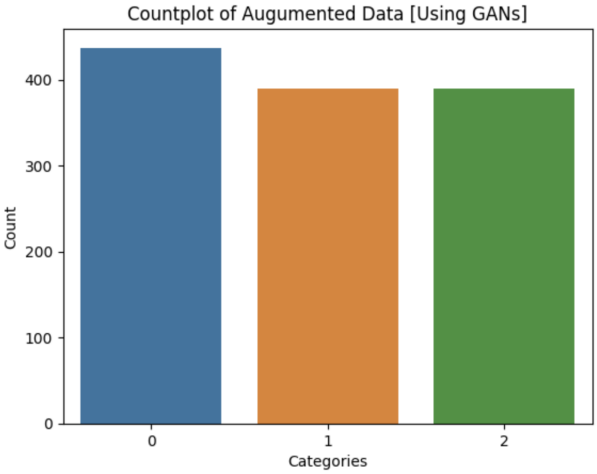

Figure 8. Count plot of the Augmented Data [GAN's]

3.3. Deep Learning Models

Several deep-learning models are planned for use in this COVID-19 classification project. The models that have been implemented and trained using the original, traditional augmented training data:

- ResNet-50[2]: ResNet-50, a 50 layer version of the ResNet (Residual Network) architecture, has become a prominent model for image classification tasks. Developed by Microsoft Research in 2016, ResNet introduced a novel approach to addressing the vanishing gradient problem by incorporating residual connections. ResNet-50's distinctive feature is the use of residual blocks that enable the learning of residual

functions by creating skip connections, aiding in the training of deeper networks without facing degradation issues. For our predictive task, we leveraged ResNet-50 to harness its powerful feature extraction capabilities and hierarchical representations. The model's ability to learn complex patterns and representations made it well-suited for our image classification task, demonstrating its effectiveness in achieving accurate predictions.

- Xception[1]: Xception, a variant of the Inception architecture, is renowned for its effectiveness in image classification. Developed by Google researchers, Xception improves upon Inception by utilizing depthwise separable convolutions, reducing parameters and computations for enhanced performance. This model excels at extracting intricate features from images, thanks to its layered architecture. In our specific use case, we incorporated Xception for its advanced feature extraction capabilities, leveraging its depthwise separable convolutions. The model's architecture, consisting of multiple layers, facilitates the learning of complex representations. For our classification task with three classes, we further tailored the model by modifying its fully connected (FC) layer. This adaptation involves a sequence of linear layers, ReLU activation functions, and dropout layers, enhancing its ability to classify medical images. The final layer employs a softmax activation function, ensuring probability distributions across the three classes.

- MobileNet-v2[5] : MobileNetV2 is a lightweight neural network designed for mobile and edge computing. In our setup, we adjusted the model for our task. We looked at the features it uses, and then we customized its classification part. Now, it has a simple structure with a layer having 512 neurons, followed by a ReLU activation and a dropout for reliability. The next layer helps classify into three classes using softmax for probability distribution. This makes MobileNetV2 effective for classifying medical images in our scenario.

- Custom Model: The custom image classification model is designed for efficiently analyzing images. It utilizes a series of convolutional layers (conv), along with residual connections to capture essential features. The model's architecture involves a stack of these convolutional layers, interspersed with residual blocks for enhanced feature learning. The final section of the model consists of classification layers, including adaptive average pooling, max pooling, dropout for regularization, and a linear layer for predicting classes. This model is tailored for three-class image classification tasks, taking RGB images with 3 channels as input. The instantiation of the model specifies these input and output configurations, and it can be easily adapted for different image classification scenarios.

- VGG16[8]: VGG16 was developed in 2014, yet it has still been considered one of the best image classification algorithms to date. The architecture includes 16 layers with convolution filters where 3 x 3 filters and 2 x 2 max pooling layers are stacked. Between these layers, the ReLU activation function is applied. Then, three fully connected layers contain most of the parameters of the network. Finally, a softmax function is used to produce the probabilities for each classification of pulmonary symptoms.

- Transfer Learning: Transfer learning involves using a pre-trained model on a large data set and adapting it to a different task or data set, in this case, we are using ImageNet.The model trained on ImageNet has been published and other datasets can be fine-tuned using it.

3.4. ResNet-50 Setup and Results

The ResNet-50 model in use has been initialized with pre-trained weights from the PyTorch ResNet-50 model. Subsequently, it underwent additional tuning across 10 epochs using COVID-19 training data. Input images were resized to 224x224 tensors, and the output layer was adjusted to have a dimension of 3 to align with the number of classes in the dataset. The training process employed a learning rate of 0.001, Cross-entropy as the loss criterion, and ADAM as the optimizer. When fine-tuning the model using the original training data augmented using conventional methods, demonstrates a testing accuracy of 94 %. When tuning the model using the dataset augmented with synthetic images from GANs, the model demonstrated a testing accuracy of 78.79%.

To provide a thorough understanding of the training process, comprehensive plots illustrating both the training loss and accuracy are presented below. The following plot [Figure 9,10] represents the metrics for a dataset augmented through traditional methods, while the second plot showcases the metrics for data augmented using Generative Adversarial Networks (GANs). These visualizations offer insights into the performance dynamics and comparative effectiveness of the two augmentation approaches employed during the model training.

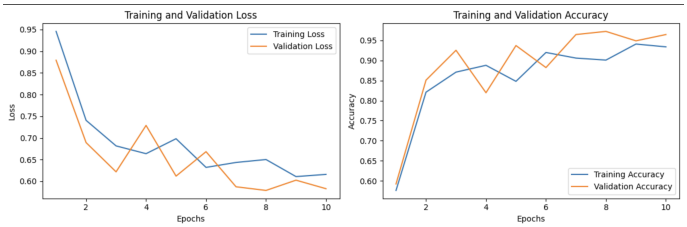

Figure 9. ResNet-50: Training Accuracy and Loss

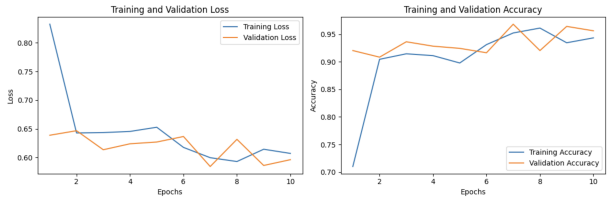

Figure 10. ResNet-50: Training Accuracy and Loss

The classification reports of the models trained using the traditionally augmented dataset can be seen in Table 1 and the GANs augmented dataset can be seen in Table 2.

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	26
1	1.00	0.80	0.89	20
2	0.83	1.00	0.91	20
Accuracy			0.94	66
Macro Avg	0.94	0.93	0.93	66
Weighted Avg	0.95	0.94	0.94	66

Table 1. ResNet-50 With Traditional Augmentation Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.74	0.96	0.83	26
1	0.78	0.90	0.84	20
2	1.00	0.45	0.62	20
Accuracy			0.79	66
Macro Avg	0.84	0.77	0.76	66
Weighted Avg	0.83	0.79	0.77	66

Table 2. ResNet-50 With GANs Augmentation Classification Report

3.5. Xception Setup and Results

The Xception model in use has been initialized with pre-trained weights from the PyTorch Xception model using the timm library. Subsequently, it underwent additional tuning across 10 epochs using COVID-19 training data. Input images were resized to 224x224 tensors, and the output layer was adjusted to have a dimension of 3 to align with the number of classes in the dataset. The training process employed a learning rate of 0.001, cross-entropy as the loss criterion, and ADAM as the optimizer. When fine-tuning the Xception model using the original training data augmented using conventional methods, it demonstrates a testing accuracy of 83%. When tuning the Xception model using the dataset augmented with synthetic images from GANs, the model demonstrates a testing accuracy of 67%.

To provide a thorough understanding of the training process, comprehensive plots illustrating both the training loss and accuracy are presented below. The following plot [Figure 11,12] represents the metrics for a dataset augmented through traditional methods, while the second plot showcases the metrics for data augmented using Generative Adversarial Networks (GANs). These visualizations offer insights into the performance dynamics and comparative effectiveness of the two augmentation approaches employed during the model training.

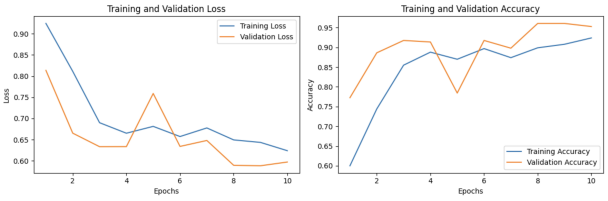

Figure 11. Xception: Training Accuracy and Loss

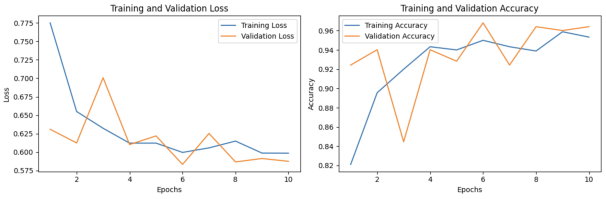

Figure 12. Xception: Training Accuracy and Loss

The classification reports of the models trained using the traditionally augmented dataset can be seen in Table 3, and the GANs augmented dataset can be seen in Table 4.

Class	Precision	Recall	F1-Score	Support
0	0.93	1.00	0.96	26
1	0.72	0.90	0.80	20
2	0.85	0.55	0.67	20
Accuracy			0.83	66
Macro Avg	0.83	0.82	0.81	66
Weighted Avg	0.84	0.83	0.82	66

Table 3. Xception With Traditional Augmentation Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.92	0.85	0.88	26
1	0.67	0.30	0.41	20
2	0.48	0.80	0.60	20
Accuracy			0.67	66
Macro Avg	0.69	0.65	0.63	66
Weighted Avg	0.71	0.67	0.66	66

Table 4. Xception With GANs Augmentation Classification Report

3.6. MobileNet-v2 Setup and Results

The MobileNet-v2 model in use has been initialized with pre-trained weights from the PyTorch MobileNet-v2 model. Subsequently, it underwent additional tuning across 10 epochs using COVID-19 training data. Input images were resized to 224x224 tensors, and the output layer was adjusted to have a dimension of 3 to align with the number of classes in the dataset. The training process employed a learning rate of 0.001, Cross-entropy as the loss criterion, and ADAM as the optimizer. When fine-tuning the MobileNet-v2 model using the original training data augmented using conventional methods, it demonstrates a testing accuracy of 95%. When tuning the MobileNet-v2 model using the dataset augmented with synthetic images from GANs, the model demonstrates a testing accuracy of 92%.

To provide a thorough understanding of the training process, comprehensive plots illustrating both the training loss and accuracy are presented below. The following plot [Figure 13,14] represents the metrics for a dataset augmented through traditional methods, while the second plot showcases the metrics for data augmented using Generative Adversarial Networks (GANs). These visualizations offer insights into the performance dynamics and comparative effectiveness of the two augmentation approaches employed during the model training.

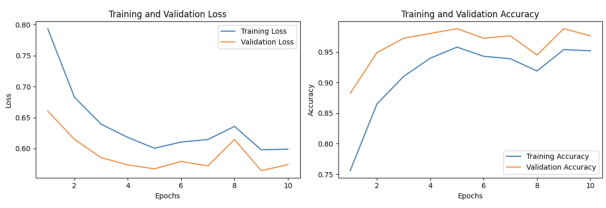

Figure 13. MobileNet-v2: Training Accuracy and Loss

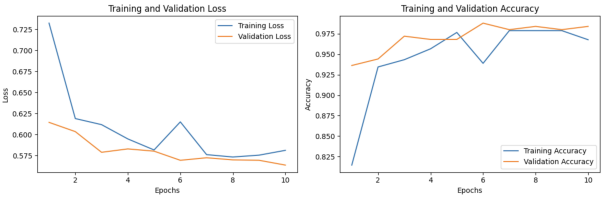

Figure 14. MobileNet-v2: Training Accuracy and Loss

The classification reports of the models trained using the traditionally augmented dataset can be seen in Table 5, and the GANs augmented dataset can be seen in Table 6.

Class	Precision	Recall	F1-Score	Support
0	1.00	0.96	0.98	26
1	0.95	0.90	0.92	20
2	0.91	1.00	0.95	20
Accuracy			0.95	66
Macro Avg	0.95	0.95	0.95	66
Weighted Avg	0.96	0.95	0.95	66

Table 5. MobileNet-v2 With Traditional Augmentation Classification Report

Class	Precision	Recall	F1-Score	Support
0	1.00	0.96	0.98	26
1	0.86	0.90	0.88	20
2	0.90	0.90	0.90	20
Accuracy			0.92	66
Macro Avg	0.92	0.92	0.92	66
Weighted Avg	0.93	0.92	0.93	66

Table 6. MobileNet-v2 With GANs Augmentation Classification Report

3.7. Custom Model Setup and Results

The custom neural network consists of convolutional blocks, residual connections, and fully connected layers. The model is tailored for three classes, with adaptive average pooling, and dropout for regularization. Our proposed custom model is further trained on our augmented training datasets over 10 epochs. The Adam optimizer was used with a learning rate of 0.001 with a dropout ratio of 0.5. When fine-tuning the custom model using the original training data augmented using conventional methods, it demonstrates a testing accuracy of 89%. When tuning the custom model using the dataset augmented with synthetic images from GANs, the model demonstrates a testing accuracy of 85%.

To provide a thorough understanding of the training process, comprehensive plots illustrating both the training

loss and accuracy are presented below. The following plot [Figure 15,16] represents the metrics for a dataset augmented through traditional methods, while the second plot showcases the metrics for data augmented using Generative Adversarial Networks (GANs). These visualizations offer insights into the performance dynamics and comparative effectiveness of the two augmentation approaches employed during the model training.

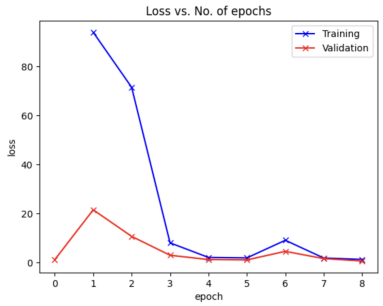

Figure 15. Custom Network: Training Loss

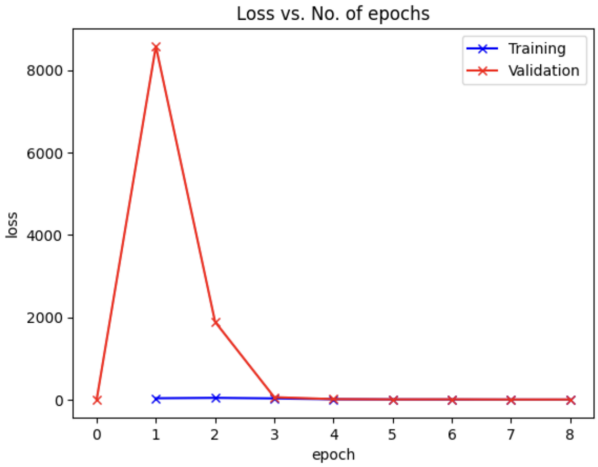

Figure 16. Custom Network: Training Accuracy Loss

The classification reports of the models trained using the traditionally augmented dataset can be seen in Table 7, and the GANs augmented dataset can be seen in Table 8.

Class	Precision	Recall	F1-Score	Support
0	0.89	0.92	0.91	26
1	0.86	0.90	0.88	20
2	0.94	0.85	0.89	20
Accuracy			0.89	66
Macro Avg	0.90	0.89	0.89	66
Weighted Avg	0.90	0.89	0.89	66

Table 7. Custom Model With Traditional Augmentation Classification Report

Class	Precision	Recall	F1-Score	Support
0	1.00	0.96	0.98	26
1	0.75	0.75	0.75	20
2	0.76	0.80	0.78	20
Accuracy			0.85	66
Macro Avg	0.84	0.84	0.84	66
Weighted Avg	0.85	0.85	0.85	66

Table 8. Custom Model With GANs Augmentation Classification Report

3.8. VGG16 Setup and Results

The model begins with a VGG16 base for feature extraction, followed by global average pooling and dense layers. Our proposed VGG16 model is further trained on our augmented training datasets over 10 epochs and uses a batch size of 64. In each epoch, every image is randomly modified with the ImageDataGenerator of Keras. The Adam optimizer from Keras was used with a learning rate of 0.001 with a dropout ratio of 0.5. When fine-tuning the VGG16 model using the original training data augmented using conventional methods, it demonstrates a testing accuracy of 97%. When tuning the VGG16 model using the dataset augmented with synthetic images from GANs, the model demonstrates a testing accuracy of 83%. The classification reports of the models trained using the traditionally augmented dataset can be seen in Table 9, and the GANs augmented dataset can be seen in Table 10.

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	26
1	1.00	0.90	0.95	20
2	0.91	1.00	0.95	20
Accuracy			0.97	66
Macro Avg	0.97	0.97	0.97	66
Weighted Avg	0.97	0.97	0.97	66

Table 9. VGG16 With Traditional Augmentation Classification Report

Class	Precision	Recall	F1-Score	Support
0	1.00	0.81	0.89	26
1	0.71	0.85	0.77	20
2	0.81	0.85	0.83	20
Accuracy			0.83	66
Macro Avg	0.84	0.84	0.83	66
Weighted Avg	0.85	0.83	0.84	66

Table 10. VGG16 With GANs Augmentation Classification Report

To provide a thorough understanding of the training process, comprehensive plots illustrating both the training loss

and accuracy are presented below. The following plot [Figure 17,18] represents the metrics for a dataset augmented through traditional methods, while the second plot showcases the metrics for data augmented using Generative Adversarial Networks (GANs). These visualizations offer insights into the performance dynamics and comparative effectiveness of the two augmentation approaches employed during the model training.

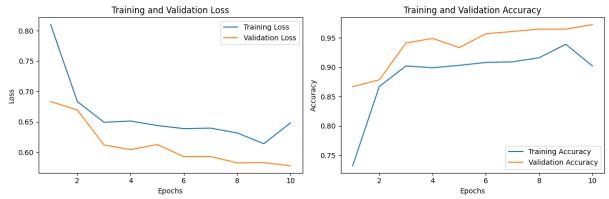

Figure 17. VGG16: Training Accuracy and Loss

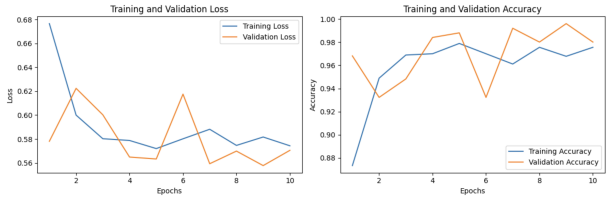

Figure 18. VGG16: Training Accuracy and Loss

4. Conclusion

Model	Traditional	GANs
ResNet-50	94%	79%
Xception	83%	67%
MobileNet-v2	95%	92%
Custom Model	89%	85%
VGG16	97%	83%

Table 11. Testing Accuracies of Models with Traditional and GANs Augmented Training Data

In conclusion, our investigation revealed that deep-learning models demonstrated superior performance when trained on datasets augmented through traditional techniques, as opposed to those enriched with synthetically generated images using Generative Adversarial Networks (GANs). The lower accuracy observed with GAN-augmented datasets was attributed to the substantial computational resources required for GAN training and their sensitivity to hyperparameters. Notably, the integration of GAN-generated images introduced slight noise, potentially impacting accuracy. Attempts to address this noise by augmenting GAN complexity were constrained by computational limitations. Despite these challenges, our thorough analysis of COVID-19 classification on lung MRI scans benefited from a hybrid approach, leveraging both traditional augmentation and GAN-generated images. This dual augmentation strategy facilitated a nuanced dataset representation, empowering the model to capture diverse patterns and features associated with COVID-19 while navigating computational constraints intrinsic to GAN training.

The link for the code of our project can be found here.

References

[1] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian J. Goodfellow, Jean Pouget-Abadie. Generative adversarial networks, 2014.

[4] a Andrea Napolitano a Emanuela Dell'Aquila a Alessio Cortellini b c Francesco Pantano a Bruno Vincenzi a Giuseppe Tonini a Marco Russano, a Fabrizio Citarella and Daniele Santinia. Covid-19 pneumonia and immune-related pneumonitis: critical issues on differential diagnosis, potential interactions, and management, 2014.

[5] Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen Mark Sandler, Andrew Howard. Mobilenetv2: Inverted residuals and linear bottlenecks, 2018.

[6] Simon Osindero Mehdi Mirza. Conditional generative adversarial nets, 2014.

[7] Pranav Raikokte. Covid-19 image dataset. `https://www.kaggle.com/datasets/pranavraikokte/covid19-image-dataset`, 2020.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.