A Partial Fulfillment of Internship Project on

Deep Convolutional GAN for Minority Class Oversampling: Evaluating Impact Across Classifiers

Completed at,

Indian Institute of Engineering Science and Technology, Shibpur

Under the Supervision
of

Prof. Asit Kumar Das
Department of Computer Science and Technology
Indian Institute of Engineering Science and Technology, Shibpur

Submitted By,

Sankha Ghosh (Reg. No. A01-1112-117-015-2022)
B.Sc. in Computer Science (Hons.)
Sauvik Dutta (Reg. No. A01-1112-117-014-2022)
B.Sc. in Computer Science (Hons.)

Department of Computer Science
Ramakrishna Mission Vivekananda Centenary College
Rahara, Kolkata - 700 118

Acknowledgments

Inscribing these words of gratitude feels akin to painting a masterpiece on the canvas of appreciation. This remarkable journey of learning and exploration would not have been possible without the steadfast support and encouragement of the individuals who have illuminated my path.

I reserve a special place in my heart for my beloved parents, whose unwavering love, constant support, and deep faith in my abilities have been the foundation upon which my aspirations have flourished. Their sacrifices and belief in me have been my greatest strength throughout this academic endeavour.

First and foremost, I owe a tremendous debt of gratitude to my esteemed guide, **Dr. Asit Kumar Das**, under whose guidance this project has taken shape. Their insightful feedback, patience, and encouragement have been instrumental in every phase of the work. Their mentorship not only enriched the quality of this project but also broadened my perspective as a student and a researcher. I am deeply thankful for the trust and freedom I was given to explore ideas and for the continuous support throughout the journey.

I would also like to express my sincere thanks to my friends and fellow students, whose companionship and encouragement helped make this journey smoother and more fulfilling. Their presence brought joy, motivation, and a sense of shared purpose that I will always cherish.

Finally, I extend my heartfelt appreciation to everyone who, in one way or another, contributed to the successful completion of this project. This experience has been invaluable, and I will always look back on it with gratitude.

Sankha Ghosh
Sauvik dutta

Abstract

This research endeavour centres on the exhaustive pipeline of image classification and generation utilising sophisticated deep learning methodologies. Initially, Convolutional Neural Networks (CNNs) are utilised to categorise authentic images, capitalising on their proficiency in feature extraction and spatial hierarchies. To assess both performance and adaptability, a variety of conventional and deep learning-based classifier algorithms are concurrently implemented and evaluated. In the subsequent phase, Generative Adversarial Networks (GANs) are employed to produce synthetic images that closely resemble the actual dataset. The concluding phase entails the reclassification of these GAN-generated images employing the same classifier algorithms to evaluate their efficacy and resilience on synthetic data. This comprehensive approach not only underscores the potential of CNNs and GANs in processing image data but also offers valuable insights into the performance of diverse classifiers on both authentic and artificially generated images.

Introduction

Medical image analysis plays a pivotal role in early disease detection and diagnosis, particularly in radiology, where imaging techniques such as chest X-rays are frequently used to identify pulmonary abnormalities. The use of automated systems to classify chest X-ray images through machine learning and deep learning has become increasingly important in supporting radiologists, enhancing both diagnostic speed and accuracy.

One major challenge in medical imaging datasets is class imbalance—specifically, the underrepresentation of 'normal' (non-diseased) cases compared to abnormal ones. This imbalance can lead to biased models that generalise poorly and disproportionately misclassify minority class instances, potentially compromising clinical reliability.

To mitigate this issue, this project applies Deep Convolutional Generative Adversarial Networks (DCGANs)[12] to generate synthetic "normal" chest X-ray images. This technique enables us to balance the dataset through oversampling of the minority class, leading to improved model training and more equitable performance across both classes.

We conducted a comparative study involving several classification algorithms: Decision Tree, Random Forest, Support Vector Machine (SVM), Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and DenseNet-121, a deep convolutional network well-suited for medical imaging tasks due to its efficient feature propagation and deep architecture. Each model was trained and tested on both the original imbalanced dataset and a balanced version enhanced with GAN-generated images.

The objectives of this study are to:

- Analyse the impact of class imbalance on various machine learning and deep learning models.
- Evaluate the effectiveness of **DCGAN**-based data augmentation for synthetic oversampling.
- Determine the most effective model architecture for chest X-ray classification under both imbalanced and balanced conditions.

By combining classical machine learning approaches, deep convolutional networks, and synthetic data generation, this project presents a practical and scalable solution to address class imbalance in real-world medical image classification.

Literature Review

Class Imbalance Problem in Image Classification

Class imbalance is a significant challenge in the world of image classification, where images of a certain class are present in the dataset compared to the other classes. This scenario might lead to biased models that perform well on the majority class while failing to generalise to minority classes. In medical imaging and rare disease detection, this problem can affect the performance of the model in critical diagnostic tasks.

Traditional approaches to address this issue include random oversampling, undersampling, and algorithmic techniques like SMOTE (Synthetic Minority Over-sampling Technique), ADASYN (Adaptive Synthetic Sampling Approach for Imbalanced Learning), etc. These methods offer marginal improvements; they fail to capture the underlying intrinsic features of the image data. Random Oversampling, undersampling risks overfitting, whereas SMOTE, which interpolates between feature vectors, is more suited for tabular data and often produces blurred or unrealistic synthetic samples in the image domain.

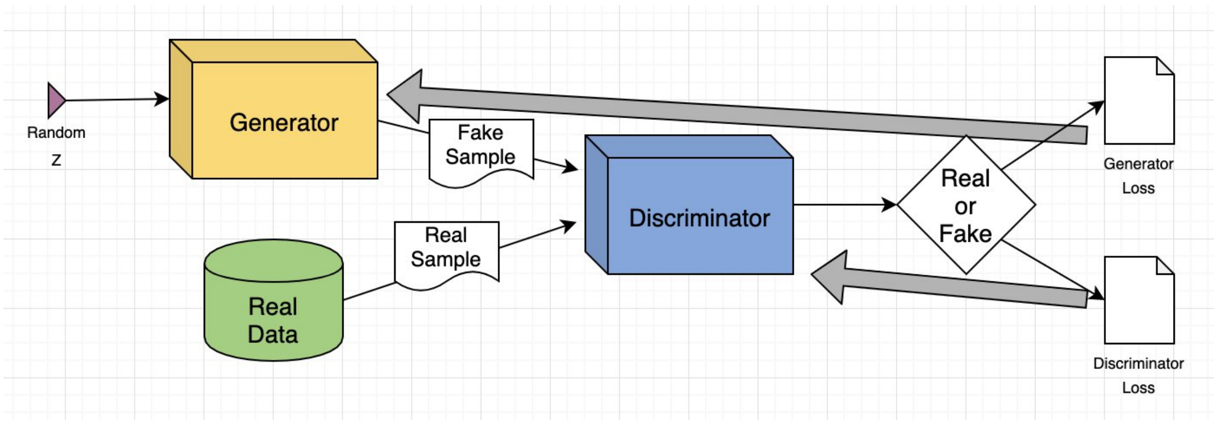


Figure 1: System of Generative Adversarial Network

Generative Adversarial Network for Data Augmentation

Generative Adversarial Network(GAN) was introduced by Goodfellow et al. (2014)[4]. GANs learn to model complex data distributions through an adversarial setup, in which two models, the generator and the discriminator, are trained in a simultaneous manner.

Deep Convolutional GANs (DCGANs), proposed by Radford et al. (2015)[12], extend this capability by incorporating convolutional and transposed convolutional layers, along with batch normalisation and ReLU/LeakyReLU activations. This architecture has become a standard baseline for image synthesis due to its stability and capacity to produce high-resolution images from random noise vectors.

The ability of DCGANs to generate entirely new yet realistic samples makes them a promising candidate for solving the class imbalance problem through data augmentation. Unlike conventional oversampling methods, GANs do not merely duplicate or interpolate existing data; they learn the underlying feature space and create new examples that enhance the diversity of the minority class.

Methodology

In this project, we address the problem of class imbalance in image datasets by leveraging Generative Adversarial Networks (GANs) for synthetic image generation. The generated images are used to augment the minority class, effectively oversampling the dataset. Following this, we conduct a comparative study of image classification using multiple models—Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Decision Tree, Support Vector Machine (SVM), Random Forest, and DenseNet121—to evaluate the impact of oversampling on classification performance.

The Generative Adversarial Network

Generative Adversarial Network is a framework (shown in Figure 1) for estimating generative models via an adversarial process.[4] In this framework, we train two multi-layer perceptron (MLP) models, a generative model G that captures the data distribution and a discriminative model D that estimates the probability that a sample came from the training data rather than G.

We train the framework by achieving the optimal condition in a min-max game played between both the networks, so that the generator could actually make realistic samples by making them real data samples for the discriminator model.

To learn the distribution of the G, p_g over data x, we define an input noise variable $p_z(z)$. Then we represent a mapping to the data space as $G(z;\theta_g)$ and $D(x;\theta_d)$, which outputs a single

scalar. Here, G is the differentiable function defined by Generator MLP, $D(x)$ represents the probability that x came from the data rather than θ_g, θ_g and θ_d are the parameters(weights and biases) of the Generator and Discriminator MLP, respectively. We train D for correctly assigning the labels to the samples coming from real data and generated by G and train G to minimise $\log(1 - D(G(z))$.
We define this min-max game with a value function $V(G, D)$,

$$\min_{\mathbf{G}} \max_{\mathbf{D}} V(D, G) = E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

In the training process, we train to D to distinguish between fake samples and real samples by converging to $D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$. After several steps of training, both the model cannot improve themselves as it reaches the scenario where $p_g = p_{data}$ and the value function reaches its global optimum. Thus, the discriminator becomes unable to distinguish between the samples, which means $D(x) = \frac{1}{2}$. The procedure is formally shown in Algorithm 1.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. Here, the algorithm is shown using $k = 1$.

```
for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
    • Update the discriminator by ascending its stochastic gradient:
        $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$
  end for
  • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  • Update the generator by descending its stochastic gradient:
        $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
end for
The gradient-based updates can use any standard gradient-based learning rule.
```

Deep Convolutional Generative Adversarial Networks

Deep Convolutional Generative Adversarial Networks (DCGANs) are a class of GANs that use convolutional and transposed convolutional layers to enable stable and efficient training for image generation tasks. Introduced by Radford et al.(2015)[12], DCGANs consist of two main components

- A generator, which learns to map random noise vectors to synthetic images.
- A discriminator, which tries to distinguish between real and generated images.

Key architectural innovations include:

- Replacing fully connected layers with deep convolutional layers.
- Using Batch Normalisation to stabilise training.
- Employing ReLU activations in the generator (except for the output, which uses Tanh) and LeakyReLU in the discriminator.

- Removing pooling layers in favour of strided convolutions and fractionally-strided convolutions.

Figure 2 is an example of DCGAN architecture [13].

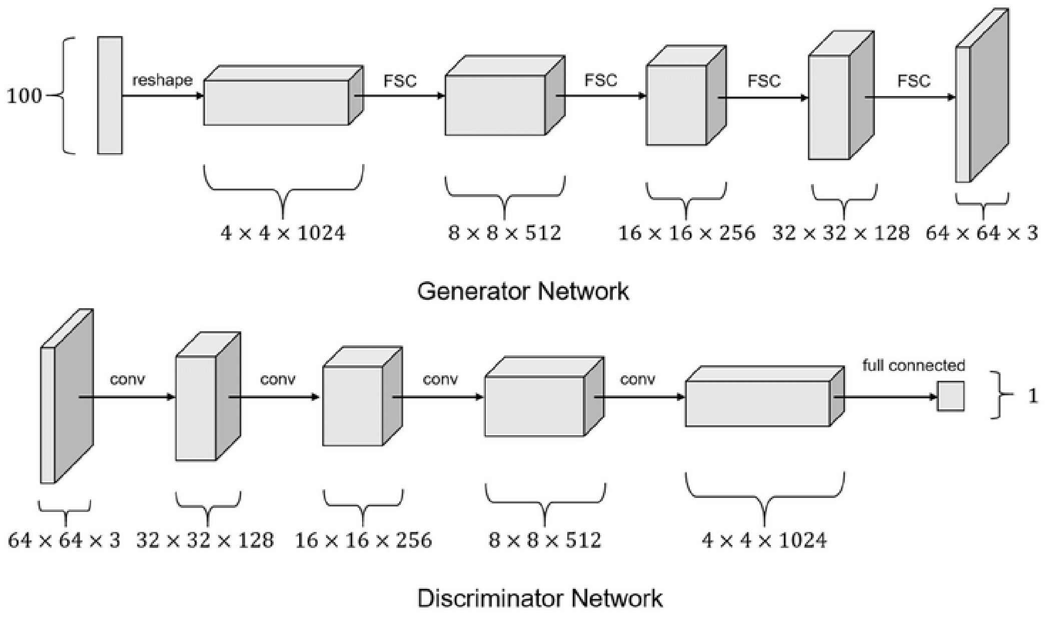


Figure 2: Example of a DCGAN Architecture

DCGANs are particularly effective in generating visually coherent images and have been applied to various domains such as art synthesis, data augmentation, and medical imaging, including the generation of synthetic Chest X-ray images to address data scarcity in healthcare AI systems.

Playing with DCGAN Architecture

For this project, the DCGAN is implemented with minor modifications to accommodate the specific properties of Chest X-ray data, which includes the LeakyReLU activation function usage in the generator instead of ReLU and the removal of the Batch normalisation layer. We used the public implementation of the GAN model available on Kaggle[1].

Generator Architecture

takes a 100-dimensional random noise vector as input and produces a 128×128×3 synthetic image through a series of transposed convolutional layers. The architecture omits Batch Normalisation to simplify training and test the model's robustness under reduced regularisation.

Architecture Summary:

- **Input**: 100-dimensional noise vector.

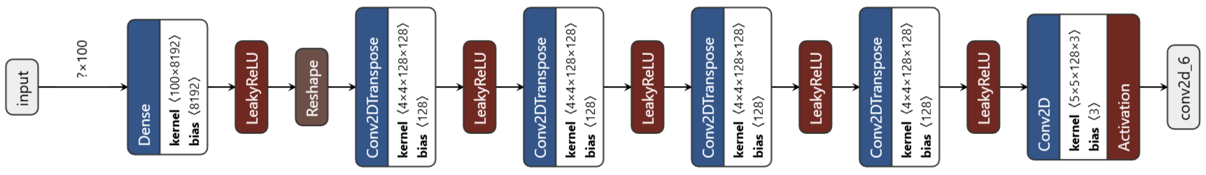


Figure 3: The Generator Model used in DCGAN training.

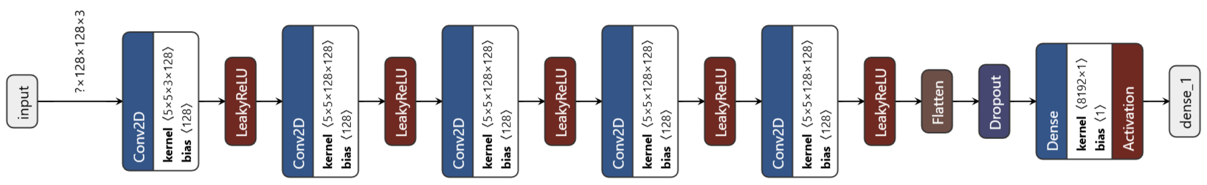


Figure 4: The Discriminator Model used in DCGAN training.

- **Dense Layer**: Fully connected layer projects to 8192 units, reshaped to a low-resolution feature map.
- **Conv2DTranspose** (×4): Upsamples spatial resolution progressively from 4×4 to 128×128.
- **Activation**: LeakyReLU in intermediate layers, Tanh at the output layer.
- **Output**: RGB image of size 128×128.

This design aligns (refer to Figure 3) with core DCGAN principles, with the exception of LeakyReLU activations in the generator, which replace the canonical ReLU, and the absence of batch normalisation.

Discriminator Architecture

receives an image (real or generated) and outputs a single probability indicating its authenticity. It uses a series of 2D convolutional layers to downsample the input image (refer to Figure 4).

Architecture Summary:

- **Input**: 128×128×3 image.
- **Conv2D** (×5): Each convolution reduces spatial dimensions while increasing channel depth.
- **Activation**: LeakyReLU used consistently to allow gradient flow even in negative domains.
- **Flatten + Dropout**: Prevents overfitting and introduces regularization.
- **Dense Layer**: Final fully connected layer with a sigmoid activation outputs a binary classification.

In both the discriminator and the generator, we have omitted the Batch Normalisation layer because Batch Normalisation might lead to fine-grained features of the Chest X-ray images and empirical results during development showed that networks using Batch Normalisation produced unstable outputs, including saturated colours (when an RGB image was used) or pure noise. In contrast, removing the Batch Normalisation layer led to more consistent and visually plausible image synthesis.

Anatomy of Classifiers

This section presents concise descriptions, pseudocode, and an analysis of the advantages and disadvantages—including those relevant to medical image classification—of six widely used classification algorithms: Decision Tree, Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Random Forest, Support Vector Machine (SVM) and DenseNet121.

Decision Tree

Decision Tree is a supervised algorithm that recursively splits data based on feature values, building a tree of decisions leading to class labels, shown in Algorithm 2. It is widely used for its interpretability and visual representation. Decision Trees are valued for their clear,

Algorithm 2 Decision Tree

```
Require: Image dataset $D$, where each image is represented by an extracted feature vector;
    with features $F$
 1: if all images belong to the same class then
 2:   return class label
 3: else if $F$ is empty then
 4:   return majority class label in $D$
 5: else
 6:   Select best feature $f$ based on splitting criterion
 7:   for all value $v$ of $f$ do
 8:     $D_v \leftarrow$ subset of $D$ where $f = v$
 9:     Child $\leftarrow$ DecisionTree($D_v$, $F \setminus \{f\}$)
10:     Attach Child to tree on branch $v$
11:   end for
12:   return node
13: end if
```

interpretable structure and minimal preprocessing requirements, making them appealing for applications where model transparency is crucial, such as clinical decision support. However, they are highly prone to overfitting and can be unstable, especially with complex or high-dimensional image data, limiting their effectiveness in detailed medical imaging scenarios.

Random Forest

Random Forest is an ensemble method that builds multiple decision trees on random samples and features, combining their outputs by majority vote to achieve better generalisation. The working procedure is shown in Algorithm 3. Random Forests improve upon single decision trees

Algorithm 3 Random Forest

```
Require: Extracted feature vectors (from images) as the training set $D$, number of trees $N$
 1: Forest $\leftarrow$ [ ]
 2: for $i = 1$ to $N$ do
 3:   Sample $D_i \leftarrow$ random subset of images and features from $D$
 4:   Tree $T_i \leftarrow$ DecisionTree($D_i$)
 5:   Add $T_i$ to Forest
 6: end for
    Prediction for a new image $x$:
 7: predictions $\leftarrow$ [ ]
 8: for each tree $T$ in Forest do
 9:   prediction $\leftarrow T$.Predict($x$)
10:   Add prediction to predictions
11: end for
12: return MajorityVote(predictions)
```

by reducing overfitting and handling high-dimensional data efficiently. They offer some measure of feature importance, aiding interpretability to a certain extent. Nonetheless, their aggregated decision-making process is less transparent than standalone trees, and ensemble methods can be computationally heavier, which may hinder their adoption when transparency and speed are vital.

Support Vector Machine

Support Vector Machine finds the optimal hyperplane that separates classes in feature space, maximising the margin between them. Kernel functions allow SVMs to handle nonlinear boundaries. The mechanism is shown in Algorithm 4. Support Vector Machines (SVMs) are effective in

Algorithm 4 Support Vector Machine

```
1: Extract feature vectors from all images
2: Initialize weights $\mathbf{w}$ and bias $b$
3: for each epoch do
4:   for each feature-label pair $(x, y)$ do
5:     if $y \cdot (\mathbf{w} \cdot x + b) < 1$ then
6:       Update $\mathbf{w}$ and $b$ to penalize misclassification
7:     else
8:       Update $\mathbf{w}$ to minimize regularization loss
9:     end if
10:  end for
11: end for
```

high-dimensional settings and generally perform well with smaller datasets—situations often encountered in specialized medical imaging tasks. However, they require careful parameter tuning and can become computationally intensive as data size increases. Their decisions are not easily interpretable, presenting challenges for deployment in clinical workflows where understanding and justifying automated decisions matter.

Artificial Neural Network

Artificial Neural Network (ANN) is a network of interconnected nodes (neurons) arranged in layers that learn to approximate functions by adjusting weights in response to input data (refer to Algorithm 5). They can model complex, nonlinear relationships. Artificial Neural Networks

Algorithm 5 Artificial Neural Network (ANN)

```
1: Initialise network weights randomly
2: for each epoch do
3:   for each image-label pair $(x, y)$ in dataset do
4:     Forward propagate $x$ to obtain prediction $\hat{y}$
5:     Compute loss between $\hat{y}$ and $y$
6:     Backpropagate error to compute gradients
7:     Update network weights using gradients
8:   end for
9: end for
```

(ANNs) excel at capturing complex, nonlinear relationships in data and offer great flexibility across various domains. When sufficient data is available, they perform well; however, their lack of transparency poses challenges for clinical acceptance, and the substantial data and computational resources they require can be prohibitive in medical contexts where high-quality labelled datasets are scarce.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are specialised ANNs designed for grid-like data such as images. They utilise convolutional and pooling layers to extract hierarchical patterns, making them highly effective for visual tasks. The algorithmic approach used is detailed in Algorithm 6. Convolutional Neural Networks (CNNs) have set the standard for medical image classification by automating feature extraction and achieving remarkable accuracy, particularly in tasks like

Algorithm 6 Convolutional Neural Network (CNN)

```
1: Initialize CNN parameters
2: for each epoch do
3:     for each image-label pair $(X, y)$ in dataset do
4:         Pass $X$ through convolutional and pooling layers
5:         Flatten and pass output through fully connected layers
6:         Compute prediction $\hat{y}$ and calculate loss with $y$
7:         Backpropagate loss; update parameters
8:     end for
9: end for
```

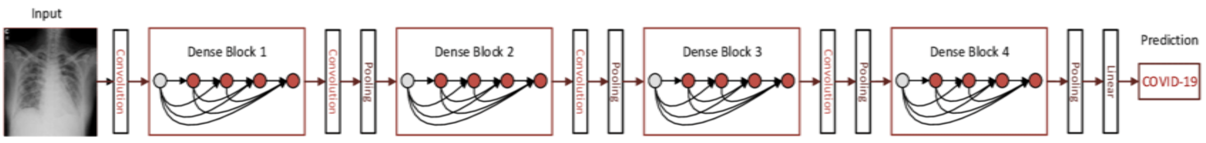


Figure 5: An architecture of DenseNet121 for COVID-19 detection[8]

disease detection and segmentation. Despite their superior performance, CNNs require large volumes of annotated data and significant computational resources. Additionally, their "black box" nature makes it difficult to provide model explanations, which is a barrier for clinical trust and regulatory approval.

DenseNet121

DenseNet121 is a 121-layer convolutional network that consists of Dense blocks, transition layers, and a fully connected layer, created by Gao Huang et. al[6], shown in Figure 5. It is characterised by its unique connectivity pattern and parameter efficiency.

In the Dense block in the architecture, a collection of convolutional layers is connected to each other. Each layer receives the feature maps from the previous layers, which means each layer adds some features on top of the existing feature maps. The arrangement of layers, i.e. Convolutional, Batch Normalisation, in the dense block is shown in the figure below.6 To make the feature shape the same across the dense layers, the transition layers perform convolution operations with 1×1 kernel and average pooling of 2×2. The key advantages of the DenseNet121 architecture in the field of image classification are:

- The features learned by each layer are connected to the other layer, because of the interconnectivity of layers in the Dense block, which strengthens feature propagation.

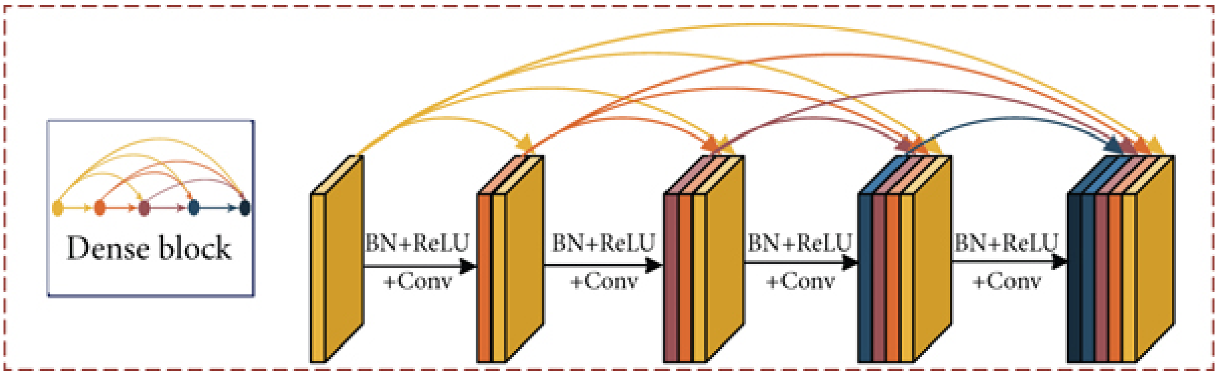


Figure 6: Internal Configuration of a Dense Block

- Intermediate and later layers in dense block don't need to learn the features learned by the previous one, promoting feature re-usability, helping the network to learn more complex features.
- The bottleneck structure and dimensional reduction at transition layers reduce the number of parameters by increasing the parameter efficiency.

Here, we have used the architecture as it is for the classification of the mentioned categories in the dataset.

Experimental Evaluation

Experimental Setup

The experiments were done on both the local machine and the Kaggle platform. The configurations used here were,

- Local Configurations
  - CPU: Intel Core i5-11400H
  - RAM: 16 GB
  - GPU: NVIDIA GeForce RTX 3050
  - Python 3.10, TensorFlow 2.12, PyTorch 2.0.1, CUDA 11.8
- Kaggle Configurations
  - GPU: NVIDIA Tesla P100 (16 GB, hosted)
  - RAM: 29 GB
  - Pre-installed: TensorFlow 2.x, PyTorch 2.x, scikit-learn 1.3+

All experiments used consistent libraries and fixed random seeds to ensure reproducibility across platforms.

Dataset

In this study, we have used Chest X-ray images of normal and Pneumonia-affected persons in Kaggle[11]. The dataset is organised into 3 folders (train, test, val) and contains sub-folders for each image category (Pneumonia/Normal). There are 5,863 X-ray images (JPEG) and 2 categories (Pneumonia/Normal). The distributions of images are shown in Figure:7 and 8.

Pixels to Patterns: Feature Extraction

To avoid the machine learning memorising the pixel values of the images, which might lead to overfitting of the model, we have done feature extraction using the ResNet18 model introduced by He et al. [6]. For detailed internal architecture, refer to Figure 9. The ResNet18 architecture is a variant of the Residual Network Architecture, consisting of 18 layers, including convolutional layers and residual blocks. A residual block in the architecture works like this: it is made up of a collection of layers, where the data goes through the layer and around skip connections. In the next residual block, the tensor which is fed as the input is the output from the previous block and the input from the skip connection, added or concatenated. The key advantages of this Residual Block and skip connections are:

- In the beginning of each residual block the input is augmented which leads to simpler learning of the network.

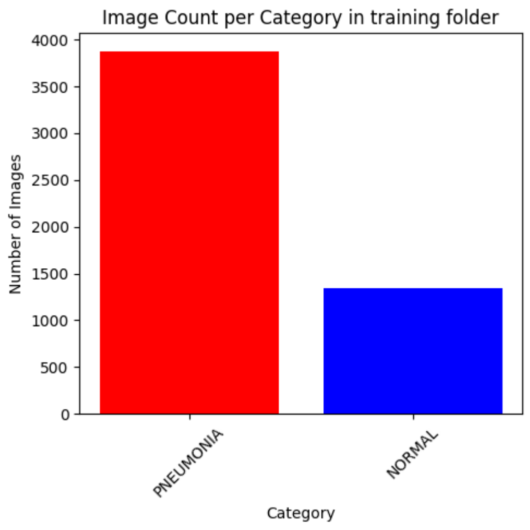


Figure 7: Image Distribution in train folder

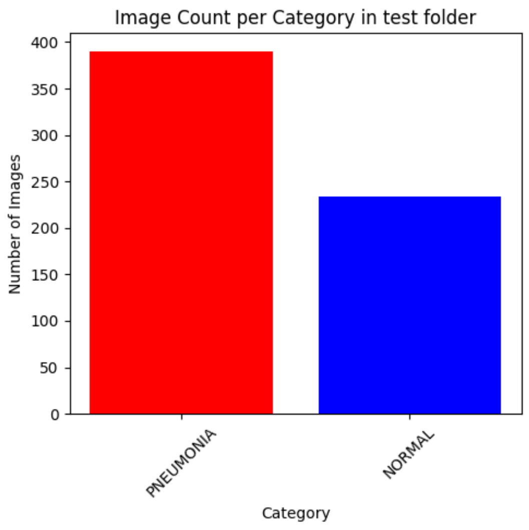


Figure 8: Image Distribution in test folder

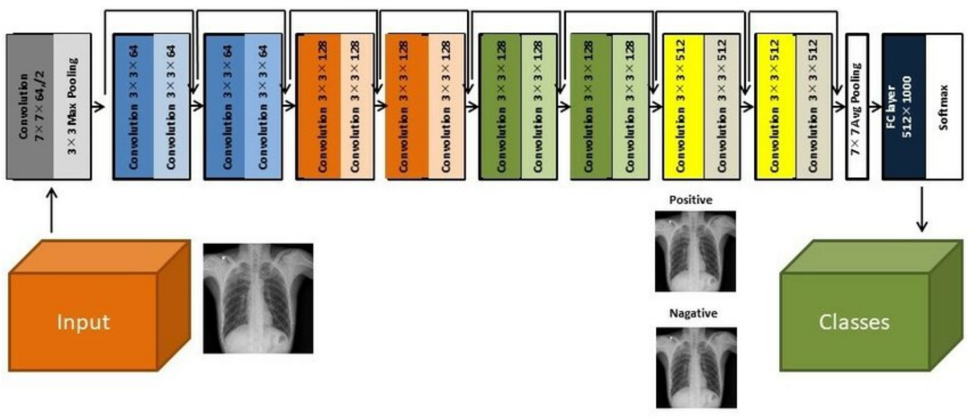


Figure 9: Architecture of ResNet18 model

- The loss from the output layers, goes back through the skip connections, for the change of weights and biases, making the gradient paths shorter, resolving the vanishing gradient problem.

- As the training process becomes faster for the above reasons, it is easy to add blocks in the residual networks, increasing the modularity of the network.

The ResNet18 architecture is a variant of this network, where there are 18 learnable layers, including the convolutional and the fully connected layer, which is the cause of the naming convention. We have used the ResNet18 architecture for feature extraction to feed the features to the machine learning models like SVM, Decision Tree, etc. We have trained the model on the dataset with a small configuration. The mechanism is shown in the Algorithm 7.

Algorithm 7 Feature Extraction using ResNet18

```
Require: Pretrained ResNet18 model, input image $I$ (size 128×128)
Ensure: Feature vector $f \in R^{512}$
  Step 1: Load pretrained ResNet-18 model
  Step 2: Remove final fully connected classification layer
  Step 3: Resize input image $I$ to 128×128, normalize to [0, 1]
  Step 4: Forward propagate $I$ through the truncated model
  Step 5: Extract output of the last hidden layer as feature vector $f$
  Step 6: return $f$
```

We have opted for this process because the early and intermediate layers capture the underlying features according to the image dataset and propagate through the layers to the last fully connected layer, where all the features have been captured. That's why we have extracted the fully connected layer as features of the images. Exemplary extracted features are shown in Figure 10 and 11.

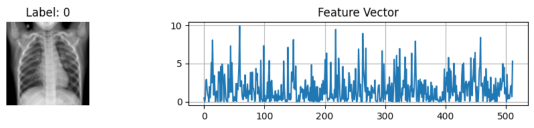


Figure 10: Extracted feature vector of a Normal category image

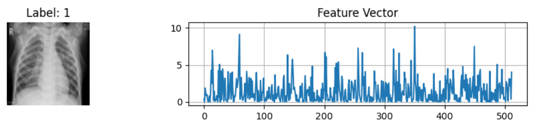


Figure 11: Extracted feature vector of a Pneumonia category image

Synthetic Image Generation

To address this class imbalance problem in the dataset, fake samples were generated by DC-GANs. The goal was to enrich the minority class with realistic, structurally coherent samples that align with the distribution of the original data.

The DCGAN, above mentioned, was trained for 100 epochs using the Adam optimizer with learning rate = 0.0002 and $\beta_1 = 0.5$ with Binary Cross Entropy loss functions for both models. These parameters are tuned based on experimental intuitions. Real images were normalized to $[-1, 1]$ to match the generator's tanh output. After completion of the training and generation, the generator was used to produce 1000 synthetic images for the minority. We made this generation several times, so we could match the number of images with the majority classes. Visual inspection ensured that these preserved anatomical consistency and did not suffer from mode collapse or saturation artefacts. These synthetic images were appended to the minority class in the training set, forming a balanced dataset for model training and evaluation. Some of the samples of the synthetic images are shown in Figure 12. The distribution classes of images are shown in Figure 13. The classification of new images along with previous images is done

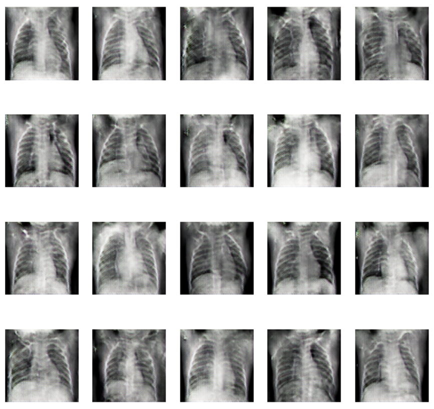


Figure 12: Sample of Normal category images generated by GAN

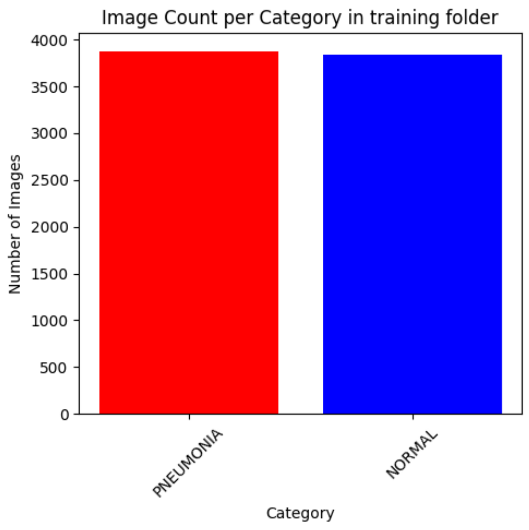


Figure 13: Class Distribution in train folder after addition of synthetic images

using the mentioned algorithms in Section . The ANN, CNN and DenseNet121 models are trained in 50, 30, and 30 epochs, respectively.

Result & Discussion

To evaluate the impact of DCGAN-based oversampling on classification performance, we employed four widely used evaluation metrics: precision, recall, F1-score, and ROC-AUC score. These metrics are particularly effective for assessing model performance under class imbalance, offering insights into both the quality and completeness of predictions.

- **Precision** measures the proportion of true positive predictions among all positive predictions:

$$Precision = \frac{TP}{TP+FP}$$

- **Recall** (also known as sensitivity) quantifies the proportion of actual positives that were correctly identified:

$$Recall = \frac{TP}{TP+FN}$$

- **F1-score** is the harmonic mean of precision and recall, providing a single measure that balances both:

$$F1\text{-}Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- **ROC-AUC score** represents the area under the Receiver Operating Characteristic (ROC) curve, and reflects the model's ability to distinguish between classes across different thresholds. It ranges from 0.5 (random guessing) to 1.0 (perfect classification). ROC Curves of models before and after synthetic image addition in the dataset are shown in Figure 18 and 19

These metrics (refer to Table 1) were computed for each model under both baseline and GAN-augmented training conditions, with a specific focus on their performance with respect to the minority class. Comparison of individual metrics is done in Figure 14, 15, 16 and 17.

Configuration	**Precision**	**Recall**	**F1-Score**	**ROC-AUC Score**
Decision Tree + DCGAN	0.7439	0.7372	0.7151	0.6752
Decision Tree	0.7539	0.7324	0.7004	0.6594
Random Forest + DCGAN	0.7858	0.75	0.7182	0.6761
Random Forest	0.7882	0.7388	0.6987	0.6577
SVM + DCGAN	0.8333	0.7885	0.7642	0.7205
SVM	0.8338	0.7933	0.7711	0.7278
ANN + DCGAN	0.8135	0.7804	0.7575	0.8569
ANN	0.8123	0.7484	0.7079	0.8418
CNN + DCGAN	0.8472	0.8317	0.822	0.8728
CNN	0.8102	0.7372	0.6902	0.8031
DenseNet121 + DCGAN	0.7613	0.9974	0.8635	0.9568
DenseNet121	0.7647	1	0.8667	0.9074

Table 1: Model Performance on Balanced vs. Imbalanced Chest X-ray Datasets

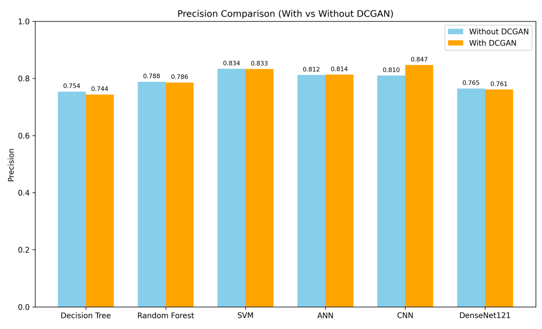


Figure 14: Precision Comparison

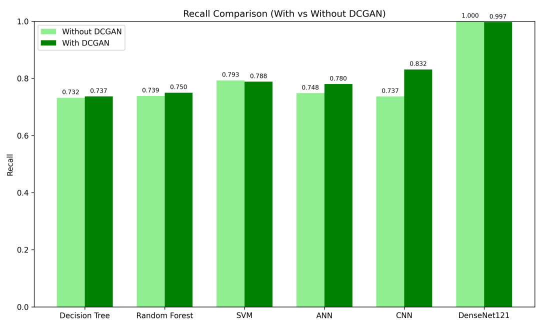


Figure 15: Recall Comparison

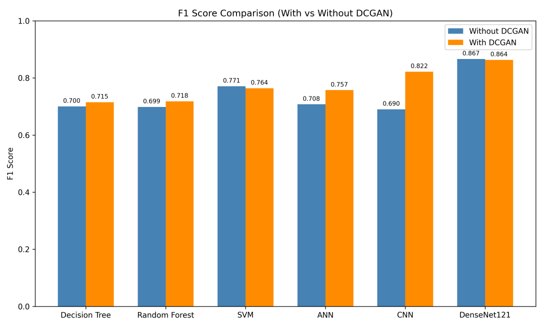


Figure 16: F1-Score Comparison

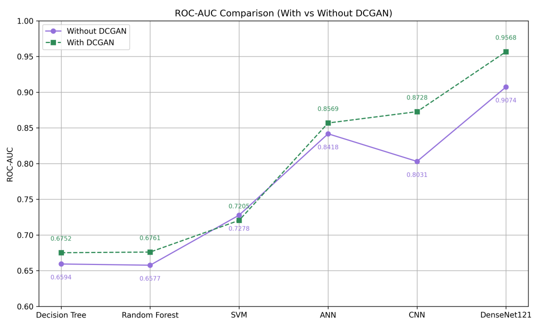
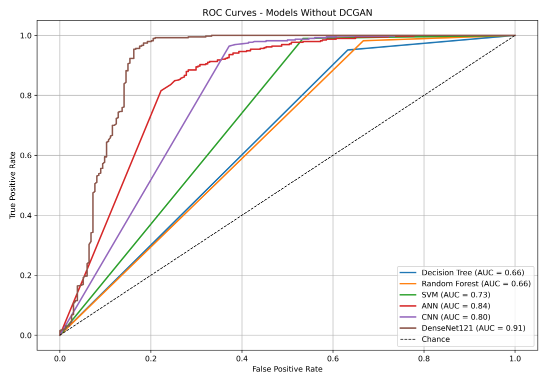


Figure 17: ROC-AUC Score Comparison



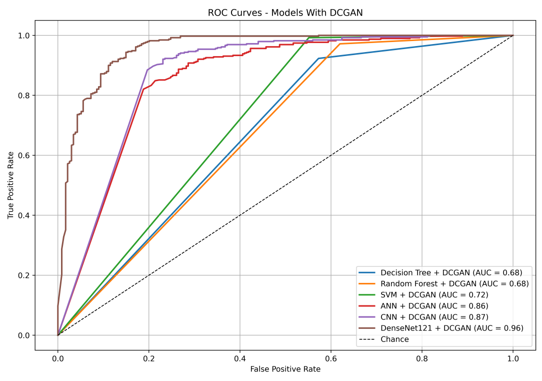


Figure 18: ROC Curves without DCGAN

Figure 19: ROC Curves with DCGAN

This study evaluated multiple classification models on chest X-ray images under both imbalanced and balanced (via DCGAN) data conditions. The balanced dataset was generated by synthetically oversampling the "normal" class using a DCGAN model.

In the case of Support Vector Machine, Random Forest and Decision Tree, we have used the feature extraction method instead of direct training on the pixel values of the images. We have used the ResNet18[5] architecture for the feature extraction of the images by extracting the last sequential fully connected layer. The features were then passed as training data for these models for classification with their corresponding labels.

Key findings:

- **Random Forest**, **Decision Tree**, and **CNN** showed significant performance drops under imbalance, especially in **F1-score** and **ROC-AUC Score**, which improved notably after balancing.
- **ANN** and **SVM** were more robust to imbalance, showing comparable or slightly improved performance on the imbalanced data.
- **DenseNet121** [6] maintained top performance in both cases, with perfect or near-perfect recall and high F1-scores.

These results confirm that DCGAN-based augmentation effectively mitigates class imbalance, particularly for models sensitive to data distribution. Therefore, we conclude that our DCGAN successfully generated high-quality synthetic chest X-ray images and that GAN-based oversampling is a valuable strategy in real-world medical image classification.

Future Scope and Improvement

While this study demonstrates the potential of DCGAN-based oversampling for improving minority class performance, future work could focus on integrating more advanced GAN models, domain-specific constraints, and evaluation metrics to further enhance the clinical utility and generalizability of synthetic data augmentation. More types of GANs, like Conditional GANs[10], StyleGAN[7], or CycleGAN[14], can be used for the generation of more realistic, class-controlled, domain-adapted synthetic images. We can add domain knowledge of medical images in GANs to improve the performance or perform attention mechanisms, segmentation masks to preserve anatomical consistency in the images. To make this work more useful in the case of real-life diagnosis, one can package the system into a real-time augmentation tool integrated into hospital workflows or AI training pipelines or enable on-the-fly augmentation to balance streaming image data during training. One can also make a comparative study with the other oversampling techniques like SMOTE, ADASYN (Adaptive Synthetic Sampling Approach), and classical augmentations for a comprehensive benchmark to analyse computational trade-offs between GANs and traditional methods.

Conclusion

Now, we can hereby conclude that we have successfully oversampled the dataset to make it well-suited for image-related tasks without biases towards any particular category. We have used the DCGAN to generate synthetic images of the minority class and added those to the dataset, and re-evaluated the dataset by performing classification using different kinds of machine learning and deep learning models for the assurance of the removal of the bias factor of the dataset. Though it would be more prominent and realistic sample generation if we used more complex and efficient variants of GANs, we have used DCGAN to simplify the task. Moreover, we believe that we have overcome the class imbalance problem as mentioned in Section Introduction, and this will be very helpful in the case of different medical-related tasks.

Further Reading

Here are some useful resources one can read for further interest in these fields.

- LANGR, T. AND BROCKHOFF, W. 2019. GANs in Action: Deep Learning with Generative Adversarial Networks. Manning Publications, Shelter Island, NY.[9]
- CHOLLET, F. 2018. Deep Learning with Python. Manning Publications, Shelter Island, NY.[2]
- Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press. [3]

Bibliography

[1] Djiby Balde. Dcgan keras chest x-ray images. `https://www.kaggle.com/code/djibybalde/dcgan-keras-chest-x-ray-images`, 2022. Accessed: 2025-07-18.

[2] François Chollet. *Deep Learning with Python.* Manning Publications, Shelter Island, NY, 2018.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, Cambridge, MA, 2016.

[4] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. corr abs/1512.03385 (2015), 2015.

[6] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[7] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[8] Saurabh Kumar, Shweta Mishra, and Sunil Kumar Singh. Deep transfer learning-based covid-19 prediction using chest x-rays. *Journal of Health Management*, 23(4):730–746, 2021.

[9] Jakub Langr and Vladimir Bok. *GANs in Action: Deep learning with Generative Adversarial Networks.* Manning Publications, Shelter Island, NY, 2019.

[10] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[11] Paul Timothy Mooney. Chest x-ray images (pneumonia). `https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia`, 2018. Accessed: 2025-07-18.

[12] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[13] Parth Sharma. A brief history of generative adversarial networks.

[14] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.