TÉCNICO LISBOA

Tackling class imbalance in medical imaging via conditional generation

Miguel Santos Coelho Ferreira Neves

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Pedro Tomás

October 31, 2022

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Abstract

Despite the increasing availability of large amounts of data, the datasets used in medical imaging are still small and have a high degree of class imbalance. To mitigate this problem, researchers often augment the training data with methods based on transformations that introduce little variability and can change the semantics of the images. An alternative approach involves using Conditional Generative Adversarial Networks (cGANs) to generate high-quality and diverse synthetic images. To this end, this work studies the suitability of this generative method, by considering different cGAN architectures, and comparing them to traditional augmentations, in a chest X-ray classification task. It is found that, despite promising, cGANs are limited by their ability to generate images from the correct distribution, and that a relevant subset of the generated images is associated with the incorrect label. Moreover, the correctly labeled images are often easily classifiable and do not improve the robustness of classifiers. This work also shows that, despite being useful, GAN metrics based on ImageNet classifiers do not provide an accurate assessment of the image generation process, and may hide important details and potentially lead to wrong conclusions.

Keywords

Conditional Generative Adversarial Networks; Data Augmentation; Class Imbalance; Medical Imaging;

Resumo

Apesar de atualmente existir uma grande disponibilidade de dados, os datasets utilizados em imagiologia médica são normalmente pequenos e têm um elevado grau de desbalanceamento de classes. Para mitigar este problema, os investigadores geralmente aumentam a quantidade de dados no dataset recorrendo a métodos baseados em transformações, que introduzem pouca variabilidade e podem mudar a semântica das imagens. Uma abordagem alternativa envolve a utilização de Redes Adversárias Gerativas Condicionais (RAGCs) para gerar imagens sintéticas diversas e de alta qualidade. Este trabalho estuda o potencial deste método, considerando diferentes arquitecturas, e comparando-o com métodos baseados em transformações, numa tarefa de classificação de raios-X ao tórax. Ao longo do trabalho apurou-se que, apesar de promissores, as RAGCs são limitadas na capacidade de gerar imagens pertencentes à distribuição correta, e que um subconjunto relevante das imagens está associado à classe errada. Além disso, descobriu-se que as imagens correctamente classificadas são facilmente classificáveis e não melhoram a robustez dos classificadores. Este trabalho mostra também que, apesar de úteis, as métricas para a avaliação de RAGCs baseadas em classificadores treinados no ImageNet não fornecem uma avaliação precisa do processo de geração de imagem, e podem esconder detalhes importantes e levar a conclusões inválidas.

Palavras Chave

Redes Adversárias Gerativas Condicionais; Aumento de Dados; Desbalanceamento de Classes; Imagiologia Médica

Contents

List of Figures

List of Tables

1

Introduction

Contents

Deep Learning methods have been a key instrument for the advancement of computer vision, often outperforming traditional methods by a large margin, while also being successful in novel applications where traditional methods failed. The major catalyst for their development was the introduction of Deep Neural Networks, allowing them to obtain state-of-the-art results in image classification, segmentation, and object detection. Developments in computing capabilities and the availability of large data volumes are essential to the successful use of these methods, as the size of the models imposes a very large number of parameters that need to be trained and tuned. If there is a lack of data, the quality of the models suffers, and so does their performance. Even though the availability of large amounts of data has increased, in many fields, e.g., medical imaging, it is still difficult to gain access to large datasets.

Applying deep learning methods in medical imaging poses a unique set of challenges. Firstly, collecting medical data requires dealing with patient privacy and clinical data management requirements, which leads to small centralized and open-source datasets. In addition, diseases follow a long-tailed distribution, as most are infrequent, with a small subset having a significant number of observed cases. Finally, the ratio between positive and negative samples is often extremely uneven, as samples from healthy individuals are far more common.

Researchers often deal with these problems by augmenting the dataset. Data augmentation includes techniques where, traditionally, transformations are applied to existing images, to artificially increase the size of the dataset. These methods are often limited by the lack of variability they introduce (for example, feature size, shape, and location) and by the risk of invalidating the associated label (for example, for digit recognition, rotating an image of the number 6 may result in the number 9). Recently, a new form of augmentation based on generative models, namely Generative Adversarial Networks (GANs), has risen in popularity, due to their incredible ability to produce never before seen synthetic images.

Augmenting a dataset with GANs can be difficult since, by default, it is not possible to control the class of the generated images. Naturally, images from non-intended classes can be filtered, but this requires a non-trivial process to distinguish between them. One solution involves training a number of GANs equal to the number of classes in the dataset (with each GAN trained solely on images from one class). However, medical datasets rarely have enough samples in the classes of interest to successfully train a GAN without incurring a severe risk of collapse or overfitting. Moreover, with an increasingly larger number of classes, this method may become prohibitively expensive. A better solution is to train a GAN on the entire dataset and later condition it to only generate images from the intended class. The conditional variant of the vanilla GAN is known as the Conditional Generative Adversarial Network (cGAN).

1.1 Objectives and Contributions

The main objective of this work is to leverage the capabilities of cGANs to generate realistic synthetic images and use them to reduce the impact of class imbalance and improve the performance of the automatic diagnosis of diseases in highly imbalanced chest X-ray datasets. This work also aims to provide a fair and extensive comparison between generative and transformation-based methods, by considering 8 conditional architectures, 16 transformations, and 2 state-of-the-art augmentations. In addition, an extensive evaluation of the generated synthetic images is presented, considering both quality and diversity. This work shows that:

1. Despite the favorable theoretical properties, the effectiveness of the conditioning mechanism limits the viability of generative augmentation with cGANs.
2. The frequently used GAN evaluation metrics calculated over features extracted by classifiers trained on ImageNet may not be the most adequate for chest X-ray datasets, hiding important details and potentially leading to incorrect conclusions.

1.2 Document Organization

This work starts by analyzing the fundamentals of data augmentation, starting with traditional methods and following with generative ones, focusing on GANs, and then looking into their application in medical imaging (Chapter 2). Chapter 3 focuses on image generation architectures, with both non-conditional and conditional versions. The following Chapter 4 describes the evaluation process, the augmentation methods employed in this work, and the results from the experiments. Chapter 5 discusses the results presented in the previous section and provides possible explanations as to why augmentation with cGANs is not as viable as traditional methods are. Chapter 6 concludes this work.

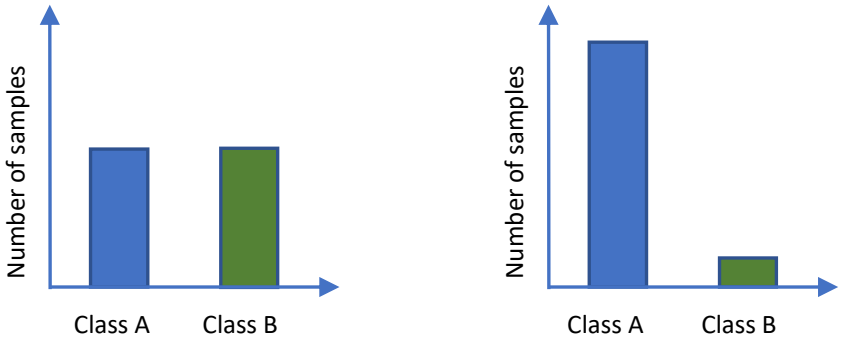
2

Related Work

Contents

2.1 Class Imbalance and Data Augmentation

A dataset is considered balanced if the number of data samples is approximately the same for every class (see Figure 2.1(a)). In contrast, an imbalanced dataset contains one or more classes with a considerably greater number of samples than the remaining (see Figure 2.1(b)). Class imbalance is widely present in medical datasets, as most diseases do not have a significant number of observed cases, and healthy individuals are far more common. When trained on imbalanced data, deep learning models often assume minority classes as rare occurrences (and over-classify the majority ones), resulting in increased misclassification of these data points. This is particularly damaging for medical applications, as the minority classes are the classes of interest.



(a) Balanced Dataset. **(b)** Imbalanced Dataset.

Figure 2.1: Class balance/Imbalance in a binary classification problem.

Specialized techniques have been developed to deal with class imbalance, which can be divided into three main approaches: data-level, algorithm-level, and hybrid [20]. Data-level approaches focus on sampling techniques, i.e., by increasing the number of samples in the minority classes (oversampling and data augmentation), reducing the number of samples in the majority classes (undersampling), or a mix of both. Algorithm-level approaches, such as cost-sensitive learning, transfer-learning, and threshold moving, aim to reduce the impact of class imbalance by adapting the architecture, changing the learning process, or modifying the objective function. Hybrid approaches mix techniques from the two.

Data-level techniques are considered the most straightforward approach as they deal exclusively with the dataset, without changing the model and/or the learning algorithm. However, not every sampling technique can be used in every situation. Undersampling is only applicable when the majority classes are represented well enough after the sample reduction, which is usually not the case for medical datasets. In these cases, oversampling techniques are preferable. Pure oversampling, ie, techniques that repeat samples in the dataset, tend to increase the risk of overfitting, and, as such, data augmentation is the preferable option.

Data Augmentation is a set of techniques used to create new synthetic images from existing ones, in order to artificially increase the size of the dataset. These techniques aim to solve, or at least to reduce the impact of class imbalance, lack of training data, and overfitting. Traditionally, data augmentation has been based on simple image transformations. Recently, techniques based on deep learning methods have emerged, particularly with the use of generative models.

2.2 Transformations

Transformation-based methods are widely used in the computer vision field as a simple, cost and time-efficient method of performing augmentation. These methods include geometric transformations, photometric transformations (transformations that modify the pixel values), kernel filters, random erasing, and image mixing. An overview and in-depth analysis of transformation-based methods is presented in [49].

One important factor to consider when transforming medical images is the safety of said transformations, i.e., if the resulting image preserves the original label. Since most distinguishing features in medical images are fine-grain details that can be easily damaged or omitted, transformations that remove portions of the image (ex: random erasing) and transformations that mix portions of images (image mixing) should not be applied. In addition, photometric transformations can be damaging or result in the same image (since images are black and white).

2.2.1 Simple Transformations

This section highlights and defines commonly used transformations. Figure 2.2 illustrates them, across 3 representative magnitude levels (strength at which the transformation is applied).

- **Auto Contrast**: Maximizes the image's contrast by remapping its pixel values so that the lowest becomes black and the lightest becomes white. Magnitude has no effect on this transformation.
- **Brightness**: Increases/Decreases the brightness of the image by a factor sampled from $[0, a]$. a is larger for larger magnitudes.
- **Contrast**: Increases/Decreases the contrast of the image by a factor randomly sampled from $[0, a]$. a is larger for larger magnitudes.
- **Equalize**: Equalizes the histogram of the image by applying a non-linear mapping to the input in order to create a uniform distribution of gray-scale values in the output. Magnitude has no effect on this transformation.
- **Gaussian Blur**: Blurs the image by passing it through a gaussian kernel of size 5×5 with σ randomly sampled from $[0, a]$. a is larger for larger magnitudes.

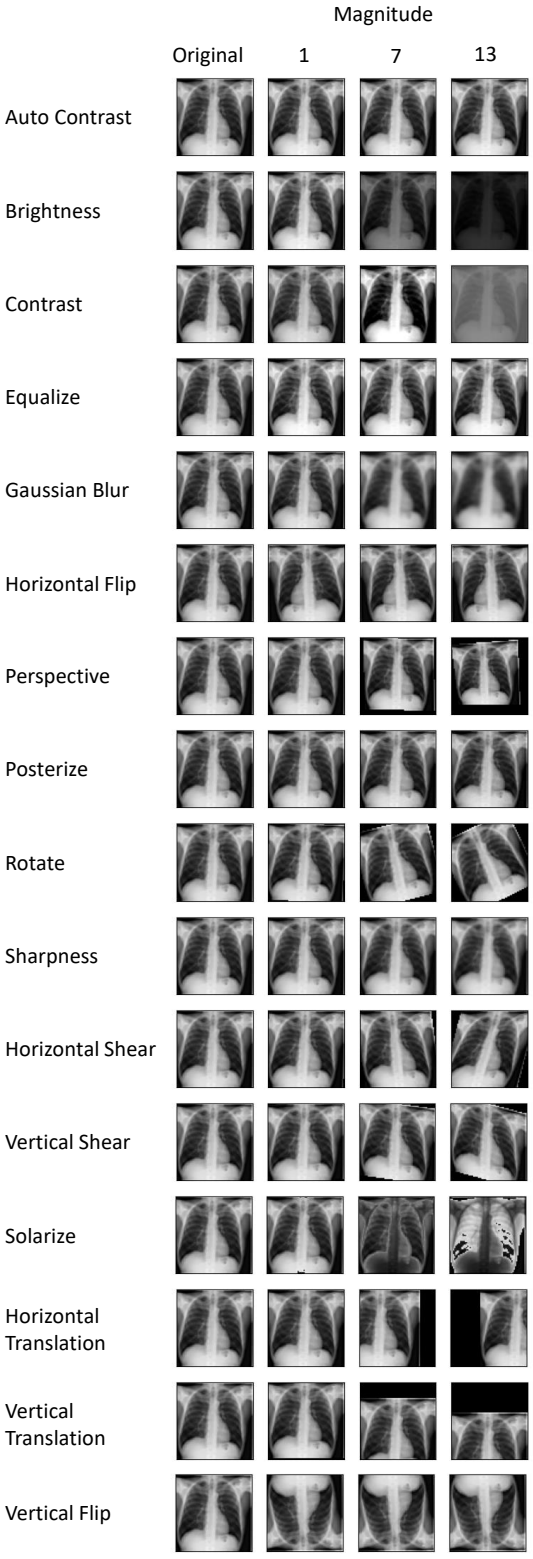


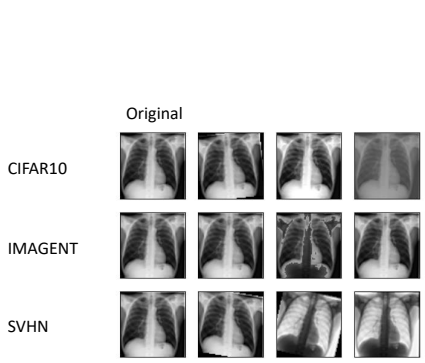
Figure 2.2: Simple transformations with varying magnitude levels.

- **Horizontal Flip**: Horizontally flips the image. Magnitude has no effect on this transformation.
- **Perspective**: Changes the perspective of the image by a factor randomly sampled from $[0, a]$. a is larger for larger magnitudes.
- **Posterize**: Reduces the number of bits in each color channel by a factor randomly sampled from $[0, a]$. a is larger for larger magnitudes.
- **Rotate**: Rotates the image by an angle randomly sampled from $[-a, a]$. a is larger for larger magnitudes.
- **Sharpness**: Increases/Decreases the sharpness of the image by a factor randomly sampled from $[0, a]$. a is larger for larger magnitudes.
- **Horizontal Shear**: Applies a function to every pixel so that a pixel with coordinates (x, y) has coordinates $(x + my, y)$, where m is randomly sampled from $[-a, a]$. a is larger for larger magnitudes.
- **Vertical Shear**: Applies a function to every pixel so that a pixel with coordinates (x, y) has coordinates $(x, y + mx)$, where m is randomly sampled from $[-a, a]$. a is larger for larger magnitudes.
- **Solarize**: Inverts all pixel values over a threshold randomly sampled from $[0, a]$. a is larger for larger magnitudes.
- **Horizontal Translation**: Translates the image horizontally by a distance randomly sampled from $[-a, a]$. a is larger for larger magnitudes.
- **Vertical Translation**: Translates the image vertically by a distance randomly sampled from $[-a, a]$. a is larger for larger magnitudes.
- **Vertical Flip**: Vertically flips the image. Magnitude has no effect on this transformation.

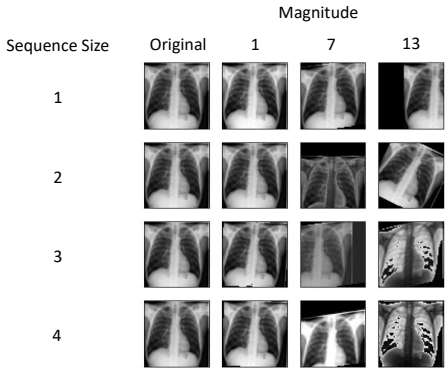
2.2.1.A Complex transformations

To complement the aforementioned transformations, other more complex ones can be applied. Two state-of-the-art techniques, AutoAugment [5] and RandAugment [6], are highlighted.

In AutoAugment, the best augmentation policy, i.e. 25 sets of 2 transformations, is automatically learned by optimizing an RNN that generates policies from a search space of transformations, probabilities, and magnitude. AutoAugment is shown to transfer well to datasets other than the one it was trained on. Figure 2.3(a) illustrates this augmentation with three policies learned on widely used benchmark datasets, Cifar10, ImageNet, and SVHN. RandAugment dramatically reduces the computational cost by removing the search phase. With RandAugment, a sequence of N equiprobable transformations of magnitude M is randomly chosen, from an augmentation space with 14 transformations. Despite being simpler, this method has comparable results to AutoAugment. Figure 2.3(b) illustrates this augmentation.



(a) AutoAugment with different policies.



(b) RandAugment with varying sequence and magnitude.

Figure 2.3: Example of complex transformations.

2.3 Generative Adversarial Networks

Generative data augmentation is based on deep generative models, a subclass of deep learning models that learn the data distribution, and generate new images from the learned distribution. Generative models differ in how the distribution is modeled and learned and include Autoregressive Models, Latent Variable Models, such as Variational Autoencoders, and Generative Adversarial Networks. Generative Adversarial Networks generate higher-quality images and are the predominant model in generative data augmentation.

Generative Adversarial Networks (GANs), first introduced in [10], consist of two models, the generator (G) and the discriminator (D). These two models are trained via an adversarial process, to create realistic synthetic images. The generator is trained to model the underlying distribution of the training data, and output synthetic images from noise vectors (z). The discriminator is trained to classify images as fake, if they were generated by the generator, or real, if they originate from the training data. The generator is usually implemented by stacking layers that progressively upsample the noise vector until an image with the desired resolution is generated. In contrast, the discriminator progressively downsamples the input image to make a decision. Figure 2.4 illustrates the general architecture.

The training process can be seen as a minimax game where D and G compete with the value function V(D, G):

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

where x is the training data.

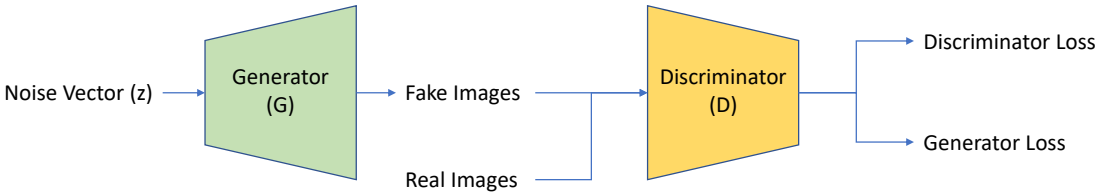


Figure 2.4: High-level architecture of a Generative Adversarial Network.

In practice, training is done iteratively with alternating optimization steps, where D is optimized for k steps for every step of G. Every D optimization step requires ascending its stochastic gradient:

$$[h!]\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(x^{(i)}\right)+\log\left(1-D\left(G\left(z^{(i)}\right)\right)\right)\right] \tag{2.2}$$

where m is the batch size, x is a sample of real images, and z is a sample of noise vectors. Every G optimization step requires descending its stochastic gradient:

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(z^{(i)}\right)\right)\right) \tag{2.3}$$

where m is the batch size and z is a sample of noise vectors.

A conditional GAN (cGAN) [38] is a variant of the vanilla GAN that performs conditional generation by encoding additional information y in the generator and the discriminator (see Figure 2.5). The minimax game takes the form:

$$\min_G\max_D V(D,G)=E_{x\sim p_{data}(x)}[\log D(x\mid y)]+E_{z\sim p_z(z)}[\log(1-D(G(z\mid y)))] \tag{2.4}$$

where y is, for example, a class label. The conditional portion may be implemented in various ways and will be explored in detail in Section 3.2.

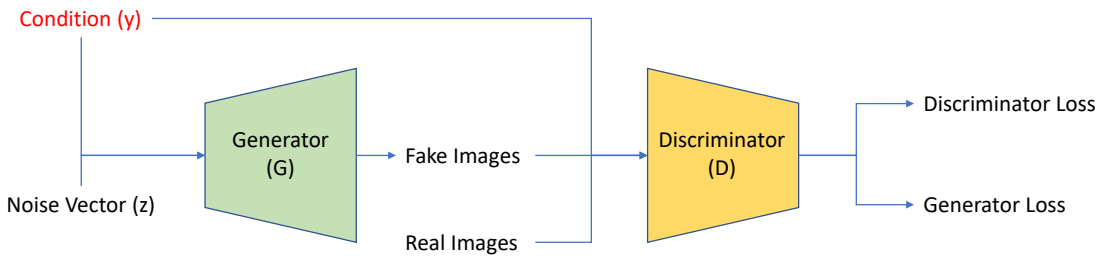


Figure 2.5: High-level architecture of a Conditional Generative Adversarial Network.

2.3.1 Objective Functions

Training stability is a desirable property of generative models that is especially hard to obtain with Generative Adversarial Networks. Some approaches for stabilizing training involve changing the sigmoid cross entropy loss function with a more stable and meaningful objective. This section presents loss functions that are/were commonly used in Generative Adversarial Networks.

Mao et al. [36] show that the cross entropy loss function leads to vanishing gradients when updating the generator with fake images located on the real side of the decision boundary. These images result in almost no error even when far from the real data distribution. The adversarial least squares loss function [36] moves these images closer to the decision boundary as it penalizes images lying far from it. The loss takes the form:

$$L_D = \frac{1}{2}E_{x\sim p_{data}(x)}\left[(D(x)-b)^2\right] + \frac{1}{2}E_{z\sim p_z(z)}\left[(D(G(z))-a)^2\right] \tag{2.5}$$

$$L_G = \frac{1}{2}E_{z\sim p_z(z)}\left[(D(G(z))-c)^2\right] \tag{2.6}$$

where a and b are the labels for fake and real data, respectively, and c is the value that the generator wants the discriminator to believe for fake data. However, parameter selection must fulfill one of two constraints. The first possible constraint enforces that $b-c=1$ and $b-a=2$, minimizing the Pearson χ^2 divergence between $p_{data}+p_{generator}$ and $2p_{generator}$. The second is to set $c=b$, forcing the generator to produce images as real as possible. In practice, both formulations achieve similar results.

Arjovsky et al. [1] prove that the traditionally minimized divergences/distances, like the Jensen-Shannon divergence used in the original GAN, are not continuous when learning distributions over low dimensional manifolds. The work attributes the source of training instability to this non-continuity and introduces the Wasserstein loss, based on the Earth-Mover/Wasserstein-1 distance, that is continuous everywhere and differentiable almost everywhere, under mild assumptions. The Kantorovich-Rubinstein duality used to construct the loss requires the function approximated by the discriminator to be 1-Lipschitz, which is enforced with weight clipping. Gulrajani et al. [11] argue that enforcing the Lipschitz constraint with weight clipping leads to capacity underuse and exploding/vanishing gradients and proposes to instead use a gradient penalty that enforces the discriminator gradients to have norm at most 1 everywhere. The final loss formulation takes the form:

$$\begin{aligned} L_D = &\underbrace{E_{x\sim p_{data}(x)}[D(x)] - E_{z\sim p_z(z)}[D(G(z))]}_{\text{loss}} \\ &+ \underbrace{\lambda E_{\hat{x}\sim p_{interp}(\hat{x})}[(||\nabla_{\hat{x}}D(\hat{x})||_2 - 1)^2]}_{\text{gradient penalty}} \end{aligned} \tag{2.7}$$

$$L_G = -E_{z \sim p_z(z)} D(G(z)) \tag{2.8}$$

where p_{interp} is the distribution obtained by interpolating between real and fake images and λ is the penalty coefficient, usually set to 10. The Wasserstein loss has the added benefit of calculating values that directly correlate with sample quality and generator convergence.

The adversarial hinge loss [33] improves training stability, reduces mode collapse, and has been widely used in recent architectures (e.g., [16, 3, 21, 28, 39, 22]). With this formulation, the discriminator is seen as a soft margin Support Vector Machine (SVM) trying to maximize the distance between real and fake images and the SVM hyperplane. The generator tries to move the fake images toward the normal vector direction of the SVM hyperplane in order to be classified as real. The adversarial loss takes the form:

$$L_D = -E_{x \sim p_{data}(x)}[\min(0, -1 + D(x))] - E_{z \sim p_z(z)}[\min(0, -1 - D(G(z)))] \tag{2.9}$$

$$L_G = -E_{z \sim p_z(z)} D(G(z)) \tag{2.10}$$

Lucic et al. [35] perform a large study on different loss functions and finds that all studied formulations achieve comparable results when given enough hyperparameter optimizations and random restarts. However, in practice, some losses outperform others in certain scenarios, forcing a trial-and-error method for loss selection.

2.3.2 Training a Generative Adversarial Network

Training GANs is notoriously difficult as the adversarial nature of the training process imposes different training instabilities, and often does not converge. Instead of changing the loss function to improve convergence and reduce instability, some works focus on developing techniques that can be easily applied to most GAN architectures. Such techniques include modifying the training process or applying regularization terms. Specialized augmentation methods have been developed to enable GAN training with small datasets. Key works are presented below, classified under the following topics: techniques, regularization, and dealing with limited data.

2.3.2.A Techniques

Salimans et al. [44] focus on training methodologies and introduce several heuristically-motivated techniques to encourage convergence and sample variability. In particular:

- *Feature Matching* changes the objective function to improve training stability by reducing overtraining on the current discriminator. Hence, instead of simply distinguishing between real and fake images, the discriminator ensures that the features extracted from the fake images are closer to the ones extracted from the real images. As such, the generator is trained with the new objective function $||E_{x \sim p_{data}(x)} f(x) - E_{z \sim p_z(z)} f(G(z))||_2^2$, where f is an intermediate layer of the discriminator.

- *Minibatch Discrimination* is based on the idea that a discriminator that decides on multiple images instead of images in isolation can reduce mode collapse. With Minibatch Discrimination, the features extracted from one sample depend on every other sample in the training batch.

- *Historical Averaging* is particularly useful to prevent the failure mode where gradient descent does not approach an equilibrium point. The technique modifies the objective function to include a new term $||\theta - \frac{1}{t} \sum_{i=1}^{t} \theta[i]||^2$ where $\theta[i]$ is the value of the parameters at time i and t the number of timesteps to be considered.

- *One-sided label smoothing* is a simple technique shown to reduce the vulnerability of neural networks to adversarial examples. The technique consists in replacing the binary labels for fake and real images (0 and 1, respectively) with smoothed values, such as 0.1 or 0.9.

- *Virtual Batch Normalization* is an alternative technique to Batch Normalization [19]. The work argues that having an input sample that depends on several other input images in the same batch can be problematic. Hence, Virtual Batch Normalization computes the sample statistics on a reference batch that is chosen once and fixed at the start of training.

Two Time-Scale Update Rule (TTUR) [15] is a training technique where the discriminator is optimized with a greater learning rate than the generator. In principle, TTUR is very similar to the k-step method referenced in Section 2.3, as both methods encourage the discriminator to learn faster and identify new features that can be transferred to the generator. However, training with TTUR is shown to not oscillate as much. Brock et al. [3] propose using the two methods in conjunction for better performance.

Yadav et al. [56] argue that the optimal solution to the GAN objective function is a saddle point and not a minimizer, which leads to training instability. In the standard training formulation, the optimization alternates between a maximization step and a minimization step, which incurs the risk of the minimization step overpowering the maximization step and the optimization process deviating from the saddle point. The work introduces a prediction mechanism for the maximization step based on the last minimization step that aims to match its strength. If the last minimization step was weak/strong, the prediction mechanism predicts a following weak/strong minimization step, resulting in a weak/strong maximization step to compensate.

2.3.2.B Regularization

Mescheder et al. [37] show that unregularized gradient descent GAN optimization does not always converge when data distributions are not completely continuous, as is often the case. The study prompted the development of new regularization terms, gradient penalties, and normalization techniques to improve convergence. Lee and Seok [32] provide an in-depth overview of their application in GANs. The following paragraphs do not intend to serve that purpose and only present methods that have been widely adopted in state-of-the-art architectures.

Internal covariate shift is defined as the change in the distribution of network activations due to the change in network parameters during training. Ioffe and Szegedy [19] argue that this property is undesirable as the network layers need to continuously adapt to the new distribution. The work introduces Batch Normalization (BN), a technique that normalizes layer outputs to have zero mean and unit variance. This is achieved by calculating layer activation statistics (mean and variance), normalizing them, and scaling and shifting the normalized values with learnable parameters γ and β. With BN, convergence speed is improved greatly. Conditional Batch Normalization (CBN) [7] incorporates linguistic information in the normalization process. Linguistic information is able to manipulate layer activations, scaling them up or down, negating them, or shutting them off, by predicting the parameters γ and β with a multi-layer perceptron that takes a word embedding as input. CBN is especially useful for conditional generation as it can be used as a conditioning mechanism if the parameters are predicted with the class label.

Spectral Normalization (SN) [40] is a normalization technique that has seen heavy use in recent GAN architectures. It works by normalizing the discriminator weights with their spectral norm, defined as the l_2 matrix norm, to enforce the discriminator to be K-Lipschitz continuous. The authors show that, when compared with other regularization approaches, SN provides faster training and produces images with higher quality.

2.3.2.C Limited Data

Of interest to medical imaging is another subset of training techniques based on data augmentation methods to enable GAN training on small datasets. When training with small datasets, the discriminator often overfits on the training examples resulting in a diverging training process. Data augmentation is usually the solution to this problem. Tran et al. [53] show that transformation-based augmentation methods (such as rotations, flips, crops, etc) can not be easily applied to datasets intended for GAN training as the generator may learn the distribution of the augmented images. To discourage the generator to match the augmented distribution, augmentation should be applied to both real and fake images, but doing so often leads to poor convergence. The following paragraphs present methods to tackle these problems.

Karras et al. [26] argue that transformations can be used as long as they are non-leaking, meaning transformations that are invertible when applied to probability distributions. Non-leaking transformations allow the discriminator to implicitly find the correct data distribution during the training process. The work proves that most traditionally used transformations are non-leaking if they are executed with a probability <1, and introduces a discriminator augmentation framework based on 18 different augmentation methods that adapts to the level of discriminator overfit. The goal is to adjust the augmentation probability p if, heuristically, the discriminator is overfitting. If the heuristic indicates much/little overfitting, p is incremented/decremented by a fixed amount.

Zhao et al. [59] defend that transformations can be used as long as they are differentiable functions that allow the gradients to be propagated through the augmentation back to the generator. The introduced DiffAugment applies a combination of differentiable transformations, such as translation, cutout, and color space transformations, to improve the quality of generated images and reduce the number of training images needed to converge.

2.4 Generative Data Augmentation in Medical Imaging

Works on GAN-based data augmentation differ in the target domain, the problem being tackled (data scarcity or class imbalance), and the approach to image generation. Evaluation usually compares classification/segmentation metrics, such as accuracy, specificity, and sensitivity, with and without augmentation. GAN augmentation has been mainly used in medical imaging as it is the field of study where the aforementioned problems are more prevalent, and, as such, will be the focus of this section. The following works provide possible approaches to GAN augmentation when dealing with data scarcity and class imbalance.

Frid-Adar et al. [9] is one of the first works to empirically show that GAN augmentation is a valid augmentation method that can improve classification in medical imaging tasks. The work applies the procedure to liver lesion classification, with a dataset consisting of 182 images from 3 different balanced classes. Two image generation methods were tested: non-conditional generation with a GAN trained on each class separately, and class conditional generation. The work reports a baseline accuracy of 57% with no augmentation, 78.6% with transformation-based augmentation, and 85.7% with transformation-based and GAN augmentation, resulting in a 7.1% increase with GAN augmentation. The non-conditional generation outperformed the conditional variant.

Bowles et al. [2] propose using GAN augmentation in a segmentation task. A GAN conditioned on the segmentation mask is used to generate images from two imbalanced datasets, containing CT and MRI images of the brain, with 500 and 147 images respectively. The work varies the percentage of real data and augmented data in the training dataset, and empirically shows that GAN augmenta-

tion is most effective when keeping 100% real data and adding an extra 100% synthetic data. Mixing transformation-based augmentation methods and GAN augmentation proved to be beneficial, resulting in greater improvements than the sum of the improvements given by using the two methods separately. Mixed augmentation provides an improvement of 1.2% and 0.4% in the Dice score (F1 score for segmentation) when compared with no augmentation and GAN augmentation, respectively.

Han et al. [12] use a mixed approach to augmentation, combining GAN augmentation with transformations in equal parts. The work applies the augmentation method to a medium-sized balanced dataset ($\sim$13000 images) containing healthy and unhealthy brain MRI images. The mixed augmentation combines random horizontal/vertical flips, rotations, width/height shifts, shearing, and zooms with synthetic images generated by a non-conditional GAN. GAN augmentation is shown to degrade performance when used in isolation. In conjunction with transformation-based methods, the work reports a 1% and 0.3% increase in classification accuracy over no augmentation and transformation-based augmentation.

The aforementioned works give clear indications that GAN augmentation alone may not be enough to boost accuracy. However, mixing GAN augmentation with transformation-based augmentation is often beneficial as the GAN tries to fill in holes in the data distribution, while transformation-based augmentation improves robustness.

A different approach to GAN augmentation is domain transfer using the CycleGAN architecture (see Section 3.1). Sandfort et al. [45] observe that most CT scan datasets contain mainly images of CT scans performed with intravenous injection of iodine contrast. In the clinical setting, a relevant percentage of CT scans do not use contrast, constituting a distribution shift between training data and real-world data. The work proposes using domain transfer from contrast CT scans to non-contrast CT scans to alleviate the class imbalance in training datasets. The process is evaluated with a segmentation task, resulting in a substantial increase of 60% in the DICE score over no augmentation.

Research on GAN augmentation for chest X-ray classification has been limited until recently. The COVID-19 pandemic has spiked interest in the automatic diagnosis of COVID-19 induced pneumonia from chest X-ray images, which prompted the aggregation of new datasets and new research in the area. GAN-based augmentation for chest X-ray datasets has consequently risen in popularity but to a far lesser extent. Even so, this domain of application has already seen some degree of success, with some key works presented below.

Kora Venu and Ravula [30] use a GAN to generate chest X-ray images from two imbalanced classes. The work compares transformation-based augmentation methods, such as random flips, crops, and color space transformations, with GAN augmentation and shows, empirically, that the latter consistently outperforms the transformation-based methods. The work compares the two augmentation approaches against a baseline classifier with 92.8% accuracy and reports an accuracy of 93.5% and 95.5% with transformation-based and GAN augmentation, respectively.

Waheed et al. [55] leverage conditional image generation to augment a small chest X-ray dataset with 403 COVID images and 721 normal images. The work reports an accuracy of 85% with no augmentation and 95% with GAN augmentation. An increase of 7% and 21% in precision and recall is observed for the COVID class, improving the detection rate of this pathology.

Sundaram and Hulkund [50] leverage a cGAN [13] pretrained on a highly imbalanced chest X-ray dataset with 14 different pathologies. Experiments involve two steps: selecting an increasingly smaller percentage of the available training data and then balancing the resulting dataset with synthetic images. GAN augmentation is evaluated against a baseline of no augmentation and transformation-based augmentation methods. The work reports a consistent increase in accuracy when compared with no augmentation and transformation-based augmentation, with diminishing returns on larger datasets. However, it is important to note that the cGAN was never trained on the smaller datasets, which may skew the results in favor of GAN augmentation.

In contrast to what is shown in [2, 9, 12], GAN augmentation when applied to chest X-ray imaging tends to outperform transformation-based methods, even when used in isolation.

2.5 Summary

A dataset is considered imbalanced when there is one or more classes with a greater number of samples than the remaining. When trained on imbalanced data, deep learning models often assume minority classes as rare occurrences (and over-classify the majority ones), resulting in increased misclassification of these data points. This is particularly damaging for medical applications, as the minority classes are the classes of interest. Data augmentation is a technique developed to mitigate this problem, where the number of samples in the minority classes is increased through image generation methods. Traditionally, these methods have been based on transformations. Recently, techniques based on Generative Adversarial Networks (GANs) have emerged.

Transformation-based methods are a simple, cost and time-efficient method of performing augmentation. Frequently used transformations include Auto Contrast, Brightness, Contrast, Equalize, Gaussian Blur, Horizontal Flip, Perspective, Posterize, Rotate, Sharpness, Horizontal Shear, Vertical Shear, Solarize, Horizontal Translation, and Vertical Flip. More complex methods built from these transformations can be used, such as AutoAugment and RandAugment.

Generative Adversarial Networks (GAN) are composed of two models, the generator and the discriminator. These models are trained via an adversarial process, to create realistic synthetic images. The generator is trained to model the underlying distribution of the training data and output synthetic images from noise vectors. The discriminator is trained to classify images as fake, if they were produced by the generator, or real, if they originate from the training data.

Training GANs is particularly difficult as they often collapse and do not converge, due to instabilities that stem from the adversarial nature of the training process. One line of work involves replacing the adversarial sigmoid cross entropy loss with a more stable objective. The Wasserstein loss took important steps toward understanding the source of the training instability and greatly improved the convergence of GANs. Recently, the hinge loss function has seen wide adoption in state-of-the-art architectures. Another line of work focuses on developing training techniques that can be easily applied to most architectures, namely the Two Time-Scale Update Rule (TTUR). Some works focus on regularization, such as Batch Normalization (BN), Conditional Batch Normalization (cBN), and Spectral Normalization (SN), while others focus on the limited data problem, developing transformations-based methods that do not leak in the generated images, such as Adaptive Discriminator Augmentation (ADA) and Differential Augmentation (DiffAugment).

Generative augmentation when applied to medical imaging tasks is usually not enough to boost accuracy, but is beneficial when used in conjunction with transformations. For the chest X-ray case, it tends to outperform transformation-based methods, even when used in isolation.

3

Generating Synthetic X-ray Images

Contents

For the successful application of generative augmentation, high-quality images must be produced. As such, this work focuses heavily on GAN architectures, both conditional and non-conditional. Simply choosing the best-performing GAN, according to some arbitrary measure, is not sufficient to produce realistic chest X-ray images, as different GANs perform better in certain scenarios and datasets. As such, this chapter aims to analyze a wide range of architectures.

3.1 Non-Conditional Generative Adversarial Networks

The first non-conditional GAN architecture was proposed in [10]. Since then, multiple architectural variants have been developed for different purposes with various degrees of success. Non-conditional GANs introduce interesting architectural details and ideas that can be leveraged to improve on the conditional variant. In addition, non-conditional GANs can be conditioned with the methods presented in Section 3.2. This section highlights some key works and developments.

DCGAN [42] was the first GAN using only convolution layers, replacing the traditionally used pooling layers. The work also standardized the use of the Adam optimizer [29], ReLu activation for all hidden layers of the generator, tanh activation for the output of the generator, and LeakyRelu activation for all layers of the discriminator.

PGGAN [24] is one of the first works focused on generating high-resolution images. The authors argue that memory constraints and easier discrimination between fake and real images affect the quality of high-resolution images. As such, the work proposed training a GAN to generate low-resolution images, and then progressively increase the resolution by adding upsample/downsample blocks to the generator/discriminator. This way of training allows the generator to focus on large structural components at the beginning of training and progressively add more detail to the image. However, doubling the image resolution by adding a new upsample block to the generator does not occur abruptly. During the transition period from resolution $\frac{N}{2} * \frac{N}{2}$ to $N * N$ the generated image I_{N*N} is obtained with:

$$I_{N*N} = \mathrm{up}(I_{\frac{N}{2}*\frac{N}{2}}) * (1-\alpha) + \mathrm{conv}(\mathrm{up}(I_{\frac{N}{2}*\frac{N}{2}})) * \alpha \tag{3.1}$$

where up is the upsample operation, conv a stack of convolution layers, and α a variable that increases linearly from 0 to 1 over k training steps.

CycleGAN [60] focuses on the image-to-image translation problem. In image-to-image translation, the GAN learns a mapping from an input image to an output image using a training set of aligned image pairs. CycleGAN learns a mapping between two image domains without the need for paired images by changing the objective function and introducing the notion of cycle consistency. Cycle consistency is based on the idea that translating something, be it a sentence or an image, from domain X to domain Y and then back to domain X should result in the original sentence/image. The work implements the

mapping from one domain X/Y to the other Y/X with a generator/discriminator pair, for a total of two generator/discriminator pairs. To enforce cycle consistency, the work adds a cycle consistency loss to minimize the l_1 norm between the original image and the translated back image.

SinGAN [48] is of special interest for medical imaging as it learns a generative model from a single image. The architecture consists of a stack of generators and discriminators that increasingly generate higher-resolution images. The generator at layer N takes as input a noise vector z^N and the generated image from generator $N-1$. The discriminator at layer N learns to distinguish between fake images generated by generator N and real images downsampled by a factor of r^N, for some $r > 1$. As the layer number decreases, so does the number of overlapping patches in the fake images as the real images are downsampled by a smaller factor, resulting in higher-resolution images.

StyleGAN [25] proposes a generator architecture based on style transfer literature, namely Adaptive Instance Normalization (AdaIN) [18]. Instead of feeding the noise vector z to the first layer of the generator, z is instead first mapped by a function f, approximated by a multi-layer perceptron, to create an intermediate representation w that is far less entangled than z. w is then passed to a learned affine transform to produce *styles* used for AdaIN in every style block. To encourage stochastic variation, Gaussian noise is also fed to every block. With this architecture, the mapping network and affine transformations are seen as a way of drawing samples for each style from a learned distribution, and the generator network as a way of generating synthetic images from the sampled styles. StyleGAN2 [27] adds weight demodulation, lazy regularization, and path length regularization, and redesigns the network to be based on skip connections [14]. With these modifications, StyleGAN2 achieves state-of-the-art image quality.

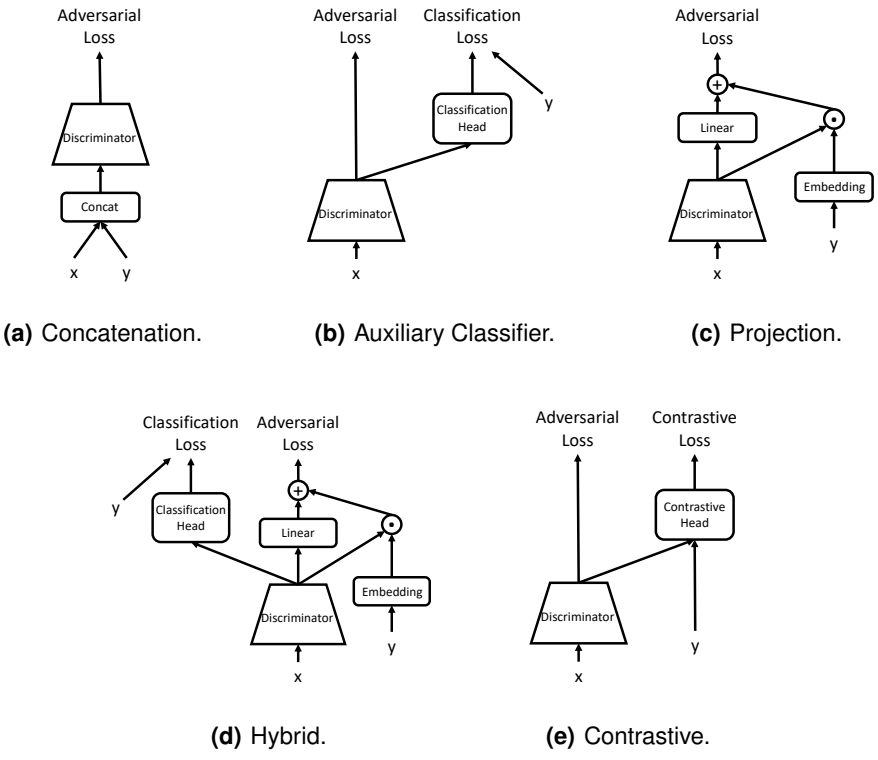
U-NET GAN [47] proposes a new discriminator based on the U-NET [43]. The U-NET is an encoder/decoder architecture where the encoder/decoder progressively downsample/upsample the input image with skip connections between matching resolutions of the two modules, improving the ability of the network to accurately detect fine-grain details. The U-NET discriminator is composed of an encoder/decoder pair, matching the U-NET architecture. The encoder behaves like a traditional discriminator, classifying images as real or fake while the decoder outputs the classification on a per-pixel level. The discriminator loss is obtained by summing the encoder loss with the decoder loss computed as the mean decision over all pixels. The decoder architecture allows for a regularization technique based on CutMix transformations [57], where fake and real images are mixed and the discriminator is penalized for inconsistent pixel-level classifications. This regularization encourages the discriminator to focus on semantic and structural changes between real and fake images.

FastGAN [34] proposes a GAN structure that reduces the time to convergence and the number of training images, while achieving competitive results on small benchmark datasets. Generating images with low computational resources enforces smaller batch and model sizes, increasing the risk of model overfit and mode collapse. To address these two challenges, the work proposes adding the skip-layer

excitation (SLE) block to the generator and a discriminator based on the encoder/decoder architecture. The SLE block reformulates skip-connections [14] to allow for longer-range dependencies between resolution levels, resulting in a more robust gradient flow. The discriminator is an encoder with decoders at various resolution levels, forcing the discriminator to extract features that can be used for reconstruction. With this technique, the discriminator learns better features, yielding more comprehensive signals to the generator.

3.2 Conditional Generative Adversarial Networks

Conditional GANs can be categorized according to the conditioning mechanism used in the discriminator. The following sections highlight the architectural details and ideas introduced by various conditional architectures, grouped by the conditioning mechanism. Figure 3.1 illustrates these mechanisms.



(a) Concatenation. **(b)** Auxiliary Classifier. **(c)** Projection.

(d) Hybrid. **(e)** Contrastive.

Figure 3.1: Discriminator conditioning mechanisms.

3.2.1 Concatenation

In concatenation-based conditioning, the noise vector z is concatenated with the conditional vector y at the input layer or some hidden layer of the discriminator (see Figure 3.1(a)). Despite being simple, this

method was successfully used by the first cGAN [38] to generate MNIST digits conditioned on the class label.

LAPGAN [8] proposes a method of training GANs based on the Laplacian pyramid representation, where an image can be reconstructed from a downsampled version with a set of residuals. The image I at resolution $N * N$ is obtained with:

$$I_{N*N} = \mathrm{up}(I_{\frac{N}{2}*\frac{N}{2}}) + h_{N*N} \tag{3.2}$$

where up is the upsample operation and the residual h_{N*N} is the difference between resolution levels (this value is calculated when downsampling the original image). The LAPGAN consists of a stack of generator/discriminator pairs that produce h_{N*N}, conditioned on $\mathrm{up}(I_{\frac{N}{2}*\frac{N}{2}})$. The generator at the lowest resolution is conditioned on the class label. The major contribution is to make each step of the upsampling process as plausible as possible instead of focusing on a global notion of image fidelity.

3.2.2 Auxiliary Classifier

Classifier-based architectures (see Figure 3.1(b)), first introduced in ACGAN [41], append an auxiliary classification head to the discriminator. In addition to distinguishing between real and fake images, the discriminator must also label the image with a softmax classifier. Despite not proposing, structurally speaking, a particularly new architecture, this simple addition to the discriminator produces higher-quality images and appears to stabilize training for a small number of classes. Apart from the standard adversarial loss, the cross entropy loss is added, forcing both the generator and discriminator to maximize the probability of the discriminator correctly predicting the class label for real and fake images.

The ACGAN has been shown to have poor image diversity [41, 39] as the auxiliary classifier may encourage the generator to produce easily classifiable images when the number of classes increases. The ADCGAN [16] argues that the reason for poor diversity is due to the optimal classifier being agnostic to the generated data distribution. This way, the classifier cannot correctly distinguish between the real and generated distribution, resulting in a biased generator. To correct this problem, the ADCGAN proposes using a classifier that classifies real and fake images from the same class differently, effectively doubling the number of labels.

3.2.3 Projection

Projection-based discriminators (see Figure 3.1(c)) were introduced in ProjGAN [39] as an alternative conditioning mechanism that mitigates the problems of the ACGAN. It is possible to rewrite the optimal adversarial loss function [38] as the sum of two log-likelihood ratios:

$$f^*(x,y) = \log \frac{q(x \mid y)q(y)}{p(x \mid y)p(y)} = \log \frac{q(y \mid x)}{p(y \mid x)} + \log \frac{q(x)}{p(x)} := r(y \mid x) + r(x) \tag{3.3}$$

where x is the sample, y is the class label, q is the real data distribution, and p is the generator distribution. By assuming the conditional distributions to follow discrete or uni-modal continuous distributions, like log-linear or Gaussian, the expression can be further simplified:

$$f(x,y;\theta) := f_1(x,y;\theta) + f_2(x;\theta) = y^{\mathrm{T}} V \phi\left(x;\theta_\Phi\right) + \psi\left(\phi\left(x;\theta_\Phi\right);\theta_\Psi\right) \tag{3.4}$$

where V is an embedding of y and ϕ and ψ are functions approximated by neural networks. The parameters V, θ_Φ, and θ_Ψ can be learned by optimizing the adversarial loss. This change in conditioning allows ProjGAN to outperform previously established state-of-the-art cGANs in image synthesis and image super-resolution.

SAGAN [58] adopts the projection-based discriminator and introduces the self-attention mechanism [54] in image generation. Most convolution-based generative models have trouble modeling long-range dependencies in images, since the convolution operator has a local receptive field, resulting in poor structural or geometric patterns when images can not be easily distinguished by texture alone. One possible solution is to increase the convolution kernel size, but doing so would be at the cost of computational and statistical efficiency. The self-attention mechanism offers a better balance between long-range dependency modeling and computational and statistical efficiency.

BigGAN [3] is very similar to SAGAN but massively increases model and batch size to improve image quality. BigGAN refers to a collection of cGAN configurations optimized to generate images at 128x128, 256x256, and 512x512 resolutions. BigGan-deep consists of even deeper and larger models that outperform BigGAN models to obtain state-of-the-art image quality on high-resolution images. Even though increasing model size improves image quality, this improvement comes at the cost of training stability. After analyzing possible sources of instability, the work experiments with different regularization and gradient penalty methods but finds them to be either ineffective when applied to the generator or to degrade performance when applied to the discriminator, suggesting that this instability does not come solely from the generator or discriminator but from the adversarial training process and the interactions between them. To improve image quality at large-scale generation, BigGAN leverages several network-level techniques, such as orthogonal weight initialization, spectral normalization, shared class embeddings, skip connections from the latent space to multiple layers, self-attention, and orthogonal regularization. Orthogonal regularization allows for a truncation trick where different noise distributions are used during training and inference mode (usually Gaussian and truncated Gaussian, respectively). With this trick, there is a trade-off between sample quality and variety, with truncated sampling ranges having better quality and larger sampling ranges having better variety.

3.2.4 Hybrid

Hybrid architectures (see Figure 3.1(d)) leverage both classifier-based and projection-based conditioning and were introduced to improve the image quality of the ACGAN. The MHGAN [28] proposes using hybrid conditioning with a Crammer-Singer multi-hinge loss, and shows that it outperforms both conditioning methods when used separately.

3.2.5 Contrastive

Contrastive discrimination (see Figure 3.1(e)) aims to improve conditioning over classifier-based and projection-based architectures by borrowing concepts from contrastive learning literature. The objective of contrastive learning is to learn an image embedding by minimizing a contrastive loss such that similar images are close together and dissimilar ones are far apart in the embedding space. With this idea in mind, ContraGAN [21] introduces the notion of data-to-class and data-to-data relations. Data-to-class refers to the relations between the image embedding and the label embedding, while data-to-data refers to the relations between image embeddings. The authors argue that the ACGAN only explores data-to-class relations and that data-to-data relations can provide strong conditioning signals. As such, the ContraGAN replaces the original classification loss with a contrastive loss (2C loss) that considers both types of relations. With this formulation, not only the distance between image embeddings and class embeddings is minimized but also the distance between image embeddings in the same class. The discriminator minimizes the contrastive objective and learns the image embeddings and the generator generates images that are similar and have a small contrastive loss.

As mentioned previously, the ACGAN has poor image diversity and is prone to collapse at early stages of training [41, 39]. The ReACGAN [22] attributes the source of early training collapse to exploding gradients in the cross-entropy loss, affecting the balance between adversarial learning and classifier training. The authors found that normalizing the feature map onto a unit hypersphere is enough to mitigate this phenomenon. To increase diversity, the work replaces the cross entropy loss with D2D-CE (Data-to-data cross-entropy), a contrastive loss function that encourages data-to-class and data-to-data relations and guarantees inter-class separability and intra-class diversity.

ContraGAN and ReACGAN take inspiration from contrastive learning for the loss formulation, resulting in two conceptually similar architectures. However, the D2D-CE loss can be seen as an improvement over 2C loss as it fixes two issues that result in the generation of images from unintended classes: bias towards only minimizing the distance between images from the same class and lack of separation between images from different classes.

3.2.6 Network Architecture and Generator Conditioning

In conjunction with the aforementioned conditional architectures, a conditional version of the StyleGAN2, entitled cStyleGAN2, with projection discrimination was also proposed. This architecture achieves state-of-the-art image quality in the non-conditional case, and as such, is a promising alternative to strictly conditional architectures.

Regarding the architecture of the neural networks, the ACGAN implements a stack of convolutional layers, the ProjGAN follows a simple ResNet [14], the ADCGAN adopts the layers architecture of the SAGAN, and the ContraGAN, MHGAN, and ReACGAN adopt the layer architecture of the BigGAN. Regarding generator conditioning, cGANs end up being very homogeneous, with only one method, Conditional Batch Normalization [7], used in all variants (except the cStyleGAN2 which does not employ Batch Normalization).

Table 3.1 summarizes the conditional architectures. Some implementations deviate slightly from the original configuration (such as the one used in this work), namely the ACGAN, usually implemented with a ResNet architecture, and the MHGAN, usually implemented with the BigGAN layer architecture. These deviations do not negatively affect the performance of these cGANs, as they are simple improvements over the original configuration.

Architecture	Network	Conditioning		Loss	
		Generator	Discriminator	Adversarial	Classification/Contrastive
ACGAN	Conv	cBN	AC	SCE	Cross-Entropy
ADCGAN	BigGAN	cBN	AC	Hinge	Cross-Entropy
BigGAN	BigGAN	cBN	PD	Hinge	-
ContraGAN	BigGAN	cBN	Contrastive	Hinge	2C
cStyleGAN2	StyleGAN2	cAdaIn	PD	Logistic	-
MHGAN	SAGAN	cBN	Hybrid	Hinge	Crammer-Singer
ProjGAN	ResNet	cBN	PD	Hinge	-
ReACGAN	BigGAN	cBN	Contrastive	Hinge	D2D-CE

Table 3.1: Summary of conditional architectures. Conv corresponds to a stack of convolutional layers. cBN stands for conditional Batch Normalization. AC stands for Auxiliary Classifier. PD stands for Projection Discrimination. cAdaIn stands for conditional Adaptive Instance normalization. SCE stands for Sigmoid Cross-Entropy.

3.3 Summary

GAN architectures can be subdivided into two variants, non-conditional and conditional. The first non-conditional variant was the DCGAN, which normalized the use of convolutional layers. PGGAN proposes starting the training process by generating low-resolution images and then progressively increase their

resolution. CycleGAN proposes an architecture that translates images between domains. SinGAN is able to build a generative model from just a single image. StyleGAN2 proposes an architecture that draws styles from the noise vector and then generates an image from the sampled styles. UNET-GAN proposes a discriminator based on the U-NET architecture, with an encoder followed by a decoder. FastGAN massively reduces the time and the number of training images needed for convergence, while achieving competitive results.

Conditional architectures can be further subdivided into five subcategories, depending on the conditioning mechanism used in the discriminator. The first cGAN followed the concatenation method, where the noise vector is concatenated with the class label. LAPGAN uses this mechanism and proposes a method of training GANs based on the Laplacian pyramid representation. cGANs with an auxiliary classifier, introduced in the ACGAN, maximize the probability of the discriminator correctly predicting the class label for real and fake images. The ADCGAN proposes classifying real and fake images differently, effectively doubling the number of labels. With projection-based discrimination, introduced in the ProjGAN, the inner product between an intermediate layer and a class embedding is added to the final output of the discriminator. SAGAN introduces the self-attention mechanism in image generation. BigGAN proposes an architecture that massively increases model and batch sizes to improve image quality. Hybrid conditioning, introduced in the MHGAN, utilizes both an auxiliary classifier and projection discrimination for improved image quality. ContraGAN proposes contrastive discrimination, arguing that contrastive losses provide strong conditioning signals. The ReACGAN tackles some problems present in the ACGAN, namely early training collapse and poor image diversity.

While a discriminator can be conditioned in many ways, conditioning a generator is often limited to one technique, Conditional Batch Normalization.

4

Evaluation

Contents

This chapter aims to provide a fair and consistent evaluation of the data augmentation techniques, both transformation-based and generative, across two augmentation procedures. To this end, for each method, a classifier was trained on the augmented dataset and compared against a baseline classifier with no augmentation. This section starts by describing the dataset and classifier used in the experiments, followed by a description of the two augmentation procedures and the image generation methods. It then presents and analyses the results of the experiments.

4.1 Dataset

The dataset used in this work, entitled COVID-19 Radiography Database [4], contains 21165 chest X-rays images aggregated from multiple sources, including images from the Italian Society of Medical and Interventional Radiology. The images are labeled according to the pathology, namely covid, lung opacity, viral pneumonia, and normal. A random sample of 5% was taken, to simulate data scarcity, and the remaining 95% was discarded. The reduced dataset contains 1056 images, of which 180 are classified as covid, 300 as lung opacity, 67 as viral pneumonia, and 509 as normal. The sample was taken in a way that preserves the original class balance. The original images had a resolution of 299x299, which was downsampled (with Lanczos resampling) to 64x64 to lower the computational cost of the experiments. Finally, the dataset was split into 5 train/validation pairs, following the stratified K-fold cross-validation method. Each fold has approximately 844 training images, of which 144 are classified as covid, 240 as lung opacity, 407 as normal, and 53 as viral pneumonia, and 212 validation images, of which 36 are classified as covid, 60 as lung opacity, 102 as normal, and 14 as viral pneumonia. Figure 4.1 contains a sample of 40 images, 10 from each class.

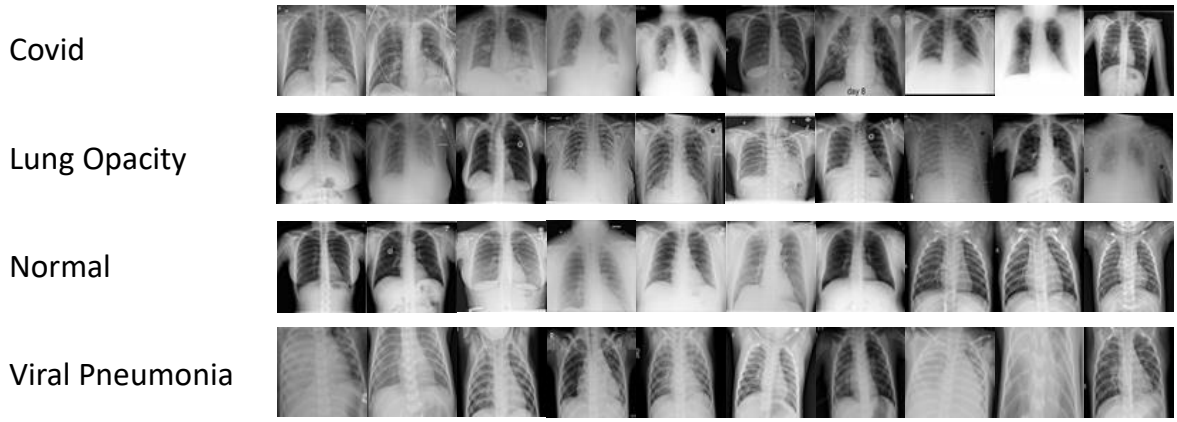


Figure 4.1: Sample of images from the dataset.

4.2 Classifier

To evaluate the augmentation procedures described in Section 4.3, this work proposes training a classifier and assessing its performance with and without data augmentation. The choice of the classifier is not particularly important given that this work focuses primarily on evaluating augmentations, and not on achieving the best possible performance. As such, small and fast classifiers are ideal for this use case. The EfficientNet [52] is a family of machine learning models with a compound scaling method that achieves competitive performance on ImageNet with a fraction of the parameters. The main building block is the mobile inverted bottleneck [46] with squeeze-and-excitation optimization [17]. The base model (EfficientNet-B0) was built with a neural network search that optimizes accuracy and FLOPS. The EfficientNet-B0 is then uniformly scaled up across all network dimensions (depth, width, and resolution) by a scaling coefficient ϕ, to obtain models B1 to B7. This work uses the B4 model as it is the most efficient classifier when considering the number of parameters and the accuracy obtained on ImageNet [52].

Even though the ADAM optimizer [29] is often regarded as the most efficient, this work uses the RMSProp optimizer, to not deviate from the configuration presented in [52]. The optimizer was configured with a learning rate of 1^{-3}, alpha equal to 0.9, and weight decay equal to 5^{-6}. To improve stability, the learning rate was decayed by 0.05 every 4 epochs. For the loss function, the cross entropy loss was used in conjunction with a softmax activation. The classifiers were trained for 70 epochs with a batch size of 8, accumulated over 8 steps.

No modifications were done to the neural network, except the last layer (the classification layer), which was swapped from a linear transformation with output size 1000 (the number of classes in ImageNet), to a linear transformation with output size 4 (the number of classes in the dataset). The weights pretrained on ImageNet were downloaded from the Nvidia repository.

4.3 Data Augmentation Methods

There are several ways to augment a dataset, depending on which problem is present. For class imbalance problems, one possible approach is to insert samples in the minority classes until every class has the same number of samples. For other problems, such as lack of data and overfitting, showing more samples to the classifier is usually sufficient. As such, two approaches to augmentation were tested, one that tackles class imbalance directly, and a more generalized procedure:

1. **Tackling class imbalance:** In the first approach, for every class c that has fewer images than a predefined threshold t, $t-n_c$ images of class c are inserted into the dataset, where n_c is the number of images in the class. The classifier is then trained on the resulting dataset. At every training

epoch, the $t - n_c$ images are generated again to increase variety. The value t was progressively increased to evaluate the impact of class imbalance. The dataset is considered balanced when t is at least 400, i.e. when the number of images in every class is the number of images in the normal class. Figure 4.2 illustrates this approach.

2. **Generalized Augmentation:** In the second approach, every image shown to the classifier is modified, mimicking the procedure commonly used in natural-image problems. Since this method does not solve the class imbalance problem, transfer learning, an algorithmic-level approach, is also used. With transfer learning, the classifier is loaded with pretrained weights (in this case, on ImageNet) before training starts. By starting with a classifier that extracts high-level features that are not biased towards a majority class, the class imbalance problem can be mitigated.

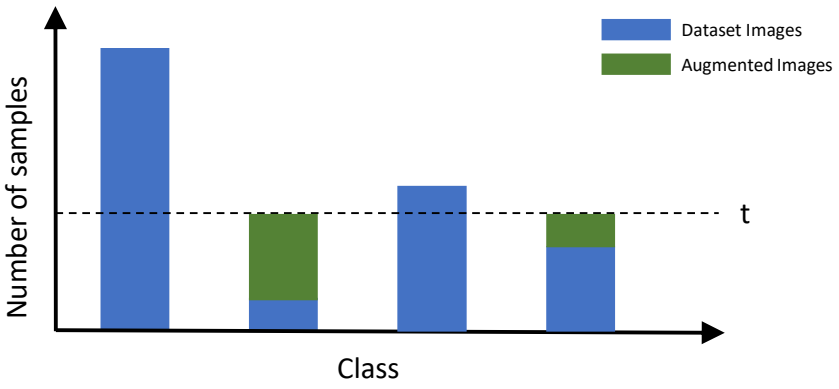


Figure 4.2: Illustration of the augmentation procedure to tackle class imbalance.

Regarding image generation, two types of methods were evaluated, one based on generative models, with the conditional GANs described in Section 3.2 (entitled cGANAugment), and one based on transformations, with simple transformations, AutoAugment, and RandAugment (Section 2.2).

The magnitude M of the simple transformations was chosen from the set $\{1, 7, 13\}$, simulating weak, medium, and strong deformations. For AutoAugment, three policies were chosen from three widely used benchmark datasets, Cifar10, ImageNet, and SVHN. Each policy considers a different collection of 25 sets of 2 transformations (with each transformation having an associated magnitude and probability p of being applied). For an image to be transformed by AutoAugment, one of the transformation sets is randomly chosen, and the two transformations are sequentially applied, with probabilities p_1 and p_2. RandAugment proposes performing a grid search over magnitude and sequence to optimize the transformation set for a particular dataset. As such, this work explores values of magnitude M in $\{1, 5, 9, 13\}$, and sequence N in $\{1, 2, 3, 4\}$. For an image to be transformed by RandAugment, N transformations of magnitude M are randomly chosen and sequentially applied.

Implementing a GAN from scratch often leads to sub-optimal results, since most of the time the difference in performance stems from implementation details that are not well specified in papers and documentation. As such, this work uses StudioGAN, a library that provides high-quality implementations of cGANs and a consistent evaluation platform, allowing for a fair benchmark across architectures. The cGANs were configured in such a way that matches the original configuration as closely as possible while keeping the network configuration the same across architectures that share the same backbone (ACGAN and ProjGAN; ADCGAN, BigGAN, ContraGAN, MHGAN, and ReACGAN). For better image quality, the networks (except the ACGAN and ProjGAN, which do not consider these techniques) included spectral normalization, on both the generator and the discriminator, and self-attention, on the third layer of the generator and the first layer of the discriminator. The batch sizes were set to 128, accumulated over 3 steps, in accord with the finding that large batch sizes improve image quality [3]. The channel multiplier was set to 32, following the observation that larger networks improve image quality [3]. The ADAM optimizer [29] was configured with beta values of 0.0 and 0.999. For improved stability, the cGANs were trained with TTUR (with a generator learning rate of 1^{-4} and a discriminator learning rate of 4^{-4}), 2 discriminator updates per generator update, and an exponential moving average of weights (starting at epoch 20k with a decay of 0.9999). Even with such precautions, some architectures trained on some dataset folds collapsed before converging, and several restarts were needed.

Training a GAN with limited data is arduous, as there are not enough training images to correctly learn and model the data distribution. With this type of data regime, the training procedure often diverges or converges to a collapsed GAN (where a GAN only generates a tiny subset of images). To mitigate this problem, the training dataset is augmented with specific procedures that ensure the generated images do not reflect the transformations. Two augmentations were tested, DiffAugment [59] and ADA [26], with the former producing better results.

In medical imaging, it is particularly important that no transformation is leaking in the generated images, as they may result in medical phenomena that, even though possible, are incredibly rare. In natural image problems, the Horizontal Flip transformation is often used as a simple way of increasing the size of the dataset and improving the synthetic image quality. However, this transformation is not differentiable and is reflected in a considerable number of generated images. While this is usually inconsequential (for example, it is completely plausible for a dog or a car to be facing either horizontal direction), for chest X-rays, this means that anatomical structures, such as the heart, are located on the right side of the chest, instead of the left. These images increase the frequency of rare medical occurrences and, when used for augmentation, might affect the quality of the final classifier.

4.4 Metrics

This work uses accuracy, precision, and recall to evaluate the quality of the trained classifiers. These metrics are calculated with:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

To evaluate the cGANs, this work uses five different metrics, four to assess image quality (FID [15], Intra FID, IS [44] and Improved Precision [31]) and one to assess the coverage of the learned distribution (Improved Recall [31]). Out of the four quality assessing metrics, FID is regarded as being the most reliable. FID (Fréchet Inception Distance) is calculated with:

$$\text{FID} = ||m_f - m_r||_2^2 + \text{Tr}(C_f + C_r - 2 * (C_f * C_r)^{\frac{1}{2}}) \tag{4.4}$$

where m and C are the mean and covariance of the feature vectors extracted from the last activation layer of the InceptionV3 model [51], obtained by feeding real (subscript r) or fake (subscript f) samples to the model. TR stands for the trace linear algebra operation. Intra FID is calculated by only considering images from a particular class. IS (Inception Score) can be estimated from N generated samples x and K classes with labels y with:

$$\text{IS}(G) \approx \exp\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} p(y_k|x_i)\log\frac{p(y_k|x_i)}{\hat{p}(y_k)}\right) \tag{4.5}$$

$$\hat{p}(y_k) = \frac{1}{N}\sum_{i=1}^{N} p(y_k|x_i) \tag{4.6}$$

where $p(y|x)$ is obtained by applying the InceptionV3 model to every sample in x. Improved Precision and Recall are calculated by first generating two image sets (one with real and one with fake images), which are then fed to the InceptionV3 model to obtain the corresponding sets of feature vectors Φ_r and Φ_f. The metrics are then calculated with:

$$\text{Improved Precision}\left(\Phi_r, \Phi_f\right) = \frac{1}{|\Phi_f|} \sum_{\phi_f \in \Phi_f} f\left(\phi_f, \Phi_r\right) \tag{4.7}$$

$$\text{Improved Recall}\left(\Phi_r, \Phi_f\right) = \frac{1}{|\Phi_r|} \sum_{\phi_r \in \Phi_r} f\left(\phi_r, \Phi_f\right) \tag{4.8}$$

$$f(\phi, \Phi) = \begin{cases} 1, \text{ if } \|\phi - \phi'\|_2 \leq \|\phi' - \text{NN}_k\left(\phi', \Phi\right)\|_2 \text{ for at least one } \phi' \in \Phi \\ 0, \text{ otherwise} \end{cases} \tag{4.9}$$

where ϕ is the feature vector of an image and NN_k are the k nearest feature vectors, determined by the Euclidean distance.

The FID, Intra FID, and IS were calculated over a set of 10000 fake images and the entire train dataset. For Improved Precision and Recall, this work uses two equal-sized sets, with sizes equal to the number of images in the training dataset, and 3 nearest neighbors (following the author's recommendations). Since these two metrics have high variance (due to small sample sizes), the evaluation was repeated 10 times and the results were averaged. The InceptionV3 model [51] pretrained on ImageNet was used as the feature extractor (following the standard practice).

One could argue that using the validation datasets to calculate the metrics would provide a better estimate of the quality of the learned distribution. However, using that portion of the data would result in biased classifiers, as we are optimizing the cGANs to generate images in the validation set. In a perfect scenario, the training set would be split into training and validation, but doing so would result in an even smaller dataset, possibly prohibiting the learning of the data distribution. Despite potentially affecting the reliability of the results, evaluating GANs on the training set is not unheard of in the field, with the StyleGAN2 using this method.

4.5 Large Scale Computing Infrastructure

Training classifiers and, especially, GANs is a lengthy process that requires a lot of computational resources. One way to accelerate training is by using hardware that supports the parallel processing of large blocks of data, namely GPUs. The RNL is a service offered by IST where a set of machines is available for remote access, corresponding to the computers used in the laboratories during school activities. Out of the 7 laboratories, two have 10 computers with GPUs, LAB1 and LAB3. The GPUs in LAB3 have large memories and clock speeds, perfect for training GANs. The machines in LAB1 were mainly used to train the classifiers. RNL leverages the cluster management system Slurm for scalability, fault-tolerance, and job scheduling. Slurm runs the submitted jobs from a batch file with the desired configuration, namely the type of machines and computational resources required. During this work,

hundreds of experiments were executed, which meant that launching jobs had to be an automated process. As such, python scripts were developed that automatically built a configuration file and submitted it to Slurm from a small set of input variables. Despite a large number of computers, the machines at RNL are utilized by the entirety of the IST, which meant that a job could be waiting for a long time (up to two weeks).

As mentioned previously, a large set of metrics needs to be computed and tracked during training. Most of the time, a simple text file is sufficient, but not when there is a need for real-time evaluation. Weights & Biases (wandb for short) is an online logging tool that facilitates real-time assessment of the training process. It works by building interactive visualizations from a set of epoch/metric pairs (continuously submitted to the platform), allowing one to easily observe and interpret their evolution, and quickly compare training runs and configurations.

4.6 Image Generation Results

Figure 4.3 illustrates the evolution of the evaluation metrics during training. Some cGANs converged in 60 thousand epochs, while others needed to be trained for longer, up to a maximum of 100 thousand, mainly waiting for Improved Recall to stabilize. Notably, the ProjGAN failed to train successfully, even after numerous random restarts and configurations.

As previously mentioned, GANs tend to train stably for an unknown number of epochs, at which point the training collapses and the overall FID increases, resulting in poor image quality. This phenomenon is especially noticeable in the ACGAN, where from epoch 20 thousand onward the FID progressively increases and never recovers. The cStyleGAN2 trained on the fifth data fold also exhibits this diverging behavior, but to a lesser extent. Curiously, for this data fold, there appears to be a tradeoff between image quality and diversity. Even though image quality decreases, as measured by the FID, the Improved Recall keeps increasing, suggesting that the discriminator is favoring diversity over quality. The IS curve for most cGANs follows an interesting path, where there is a peak in its value at the beginning of training, followed by a slump, and a gradual recovery towards the original peak. There is no clear indication as to why this is happening, but, as mentioned earlier, IS is not considered to be as reliable as FID, and these types of inaccuracies are to be expected. This unreliability is particularly obvious in the ACGAN, where the IS is similar to the remaining gans, with a FID several times larger. Out of the quality assessing metrics, the Improved precision seems to be the most unstable, especially during the early stages of training, where the value is remarkably inconsistent. These observations seem to meet the consensus that, for comparing architectures, FID is the preferable metric, while IS and Improved Precision are mostly used to assess the training process. After completing the training, the best checkpoint, measured by the lowest FID, was taken to perform a final evaluation.

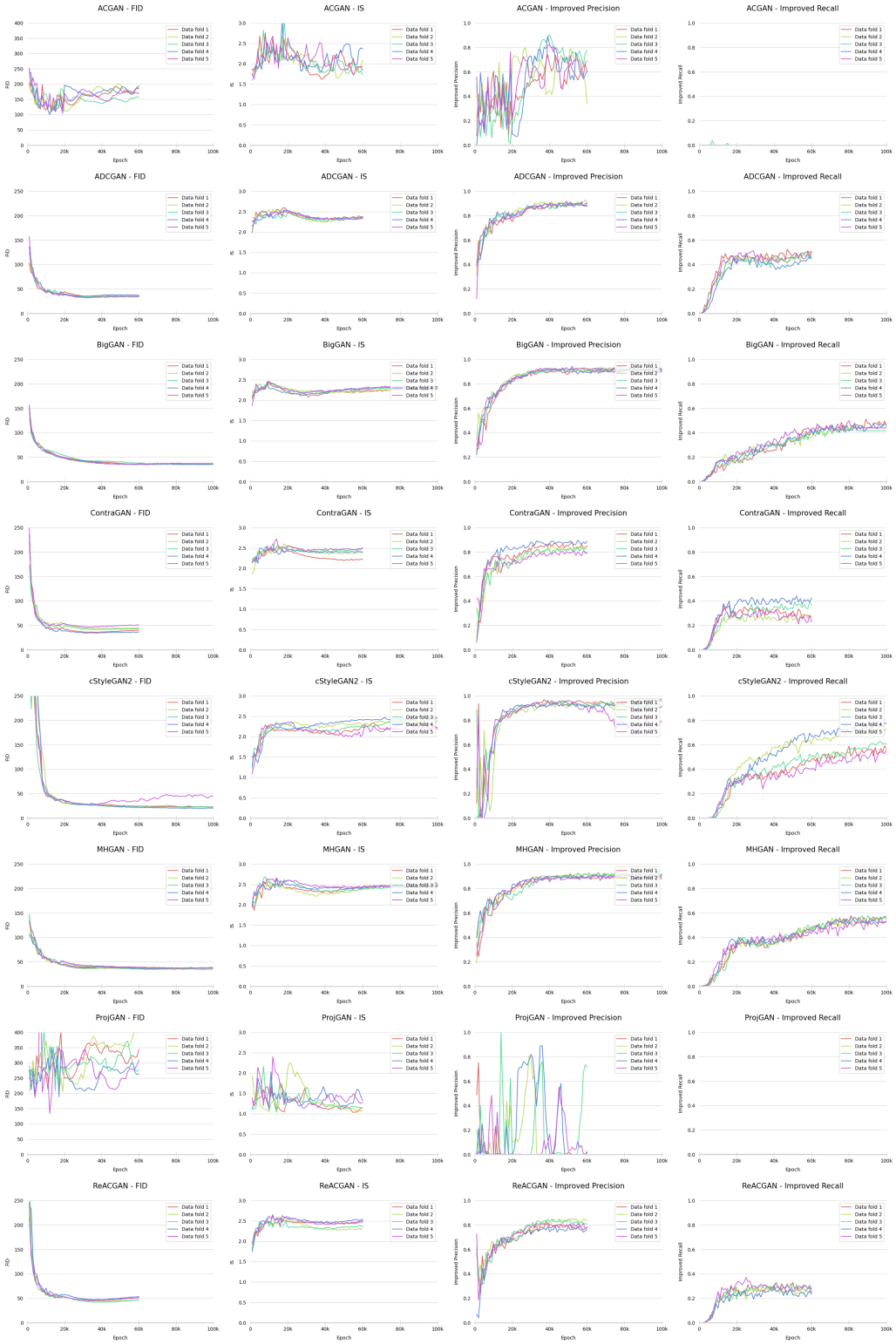


Figure 4.3: Evolution of FID, IS, Improved Precision, and Improved Recall of the images generated by the ACGAN, ADCGAN, BigGAN, ContraGAN, cStyleGAN2, MHGAN, ProjGAN, and ReACGAN.

The metrics in Table 4.1 were calculated by averaging the results from 10 evaluations with different seeds. The cStyleGAN2 generated the highest quality images, while the ProjGAN generated the lowest. Notably, the BigGAN-based architectures (ADCGAN, BigGAN, ContraGAN, MHGAN, ReACGAN) perform very similarly concerning image quality, except the two contrastive architectures, ContraGAN and ReACGAN, that perform slightly worse. Regarding image diversity, the ACGAN and the ProjGAN are clear outliers, with recall values of 0. The two contrastive architectures perform, once again, worse than the remaining BigGAN-based cGANs.

Architecture	**FID** ↓	**IS** ↑	**Improved Precision** ↑	**Improved Recall** ↑
ACGAN	112.44	2.2	0.58	0.00
ADCGAN	36.47	2.33	0.88	0.44
BigGAN	37.35	2.27	0.92	0.42
ContraGAN	46.50	2.38	0.83	0.32
cStyleGAN2	**24.86**	2.33	**0.94**	**0.59**
MHGAN	38.50	**2.44**	0.91	0.52
ProjGAN	193.32	1.75	0.34	0.0
ReACGAN	47.65	2.40	0.80	0.28

Table 4.1: Evaluation of the images generated by the cGANs.

Table 4.2 shows the per-class assessment of image quality. The cStyleGAN2 has the lowest Intra FID for every class, which correlates with having the lowest overall FID. As expected, the class with the highest number of images (Normal) has the lowest Intra FID, while the class with the lowest number of images (Viral Pneumonia) has the highest Intra FID. Surprisingly, the Intra FID for most classes is considerably higher than the overall FID, particularly in the ContraGAN, which may suggest some difficulty in generating images from the correct distribution.

Architecture	**Intra FID** ↓			
	Covid	**Lung Opacity**	**Normal**	**Viral Pneumonia**
ACGAN	170.65	153.41	127.99	207.92
ADCGAN	64.09	55.45	33.76	88.48
BigGAN	65.03	56.10	35.79	78.45
ContraGAN	103.45	93.01	71.21	147.94
cStyleGAN2	**53.71**	**42.29**	**27.44**	**73.86**
MHGAN	62.82	59.65	33.78	86.62
ProjGAN	214.44	195.11	218.07	265.18
ReACGAN	73.55	69.62	41.97	107.42
average	100.97	90.21	73.71	131.98

Table 4.2: Intra FID of the images generated by the cGANs.

Another powerful tool in GAN evaluation is a simple visual inspection. Figures B.1 and B.2 show 10 images from each class, per architecture. It is immediately obvious the differences in image quality between the ProjGAN and the remaining cGANs. The lack of variety in the ACGAN is also noticeable, especially when considering that out of 10, only 3 or 4 different images are present. The differences between the remaining cGANs are not detectable by the human eye, highlighting the importance of conducting a quantitative evaluation.

Nonetheless, we can conclude, by considering the previous observations, that the cGANs (except the ProjGAN and the ACGAN) were largely successful in generating high-fidelity chest x-ray images with a reasonable degree of variety. Since the results with ProjGAN were so poor, this architecture will be left out of the cGANAugment architectures.

Generator overfit

The Improved Recall metric only accounts for image diversity, disregarding what kinds of images are being produced. In some cases, and especially with limited data, cGANs, and GANs in general, simply learn to repeat the images in the training data. These images are still varied, as they cover the entirety of the real data distribution, but defeat the purpose of image generation. In these failure states, the Improved Recall is very high. As such, a different evaluation method needs to be conducted to test for overfitting. Usually, simple visual inspection, with a KNN method, is sufficient. For this, a set of 4000 images, 1000 of each class, were produced by the generator. This set of images was then used to find the k nearest neighbors of 4 real images, 1 for each class. The results are displayed in Figures B.3 and B.4. The first set of images corresponds to real data, while the remaining are the 10 nearest images, sorted by distance. The distance between images was calculated with the Euclidean distance over feature vectors extracted by InceptionV3. The only architecture that seems to be repeating images from the dataset is the MHGAN, especially in the Viral Pneumonia class. However, this is not enough to assume that the MHGAN generator is overfitted since, in a sample of 1000 images, it is expected that some are very similar to real data points.

On the impact of Augmentation, and transformations leaking in the generated images

It can be argued that applying transformations to the images intended for GAN training may invalidate the comparison between transformation-based augmentation and generative augmentation, since, in a way, transformations are implicitly used to generate the synthetic images. However, one could also argue that such an argument is only valid if the generated images reflect the transformations. Either way, there is a practical reason why transformations are usually used and are sometimes strictly necessary. Training GANs is a particularly complex affair due to the unstable nature of the process, which may result in collapsed GAN states, where a GAN produces a tiny subset of images, or diverge just after

a few training epochs. These problems are exacerbated by a lack of training data, which is usually the norm in medical applications. With very small datasets, i.e., datasets with less than 1000 images, training GANs is practically impossible. Figure 4.4 illustrates this very problem. For example, with a BigGAN trained without DiffAugment, the FID increases exponentially just after 20 thousand training epochs, without ever recovering. Apart from the obvious impact on stability, applying DiffAugment also immensely affects the image quality, which can be verified by the FID stabilizing at around 100.

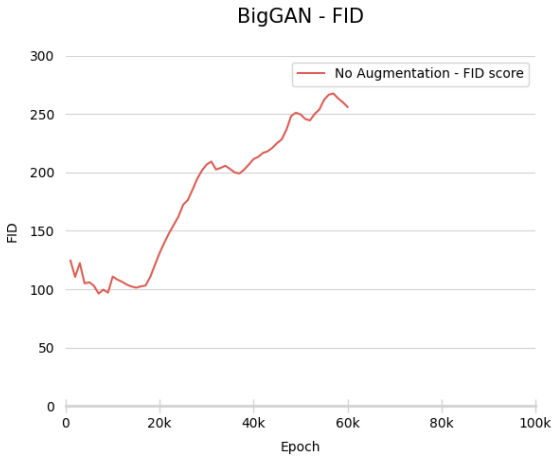


Figure 4.4: Impact of not augmenting the training images in the BigGAN architecture.

For reference, DiffAugment is constructed by sequentially applying 6 transformations: Brightness, Saturation (which does not affect black and white images), Contrast, Horizontal Translation, Vertical Translation, and Cutout. Figure 4.5 illustrates the effect of DiffAugment on the training images. By observing the images in Figures 4.5, B.1 and B.2, it appears that some images from the Viral Pneumonia class generated by the MHGAN may reflect the Brightness transformation. However, no other transformation is leaking in the remaining architectures.

4.7 Classification Results

This section presents the results with the augmentation methods proposed in Section 4.3 and the 4 proposed image generation methods, simple transformations, AutoAugment, RandAugment, and cGANAugment. For each experiment, 5 classifiers were trained, each on a different data fold. After training, the metrics (accuracy, precision, and recall) from the checkpoint with the highest validation accuracy were stored, per classifier. These metrics were then averaged across the 5 classifiers, corresponding to the results presented here.

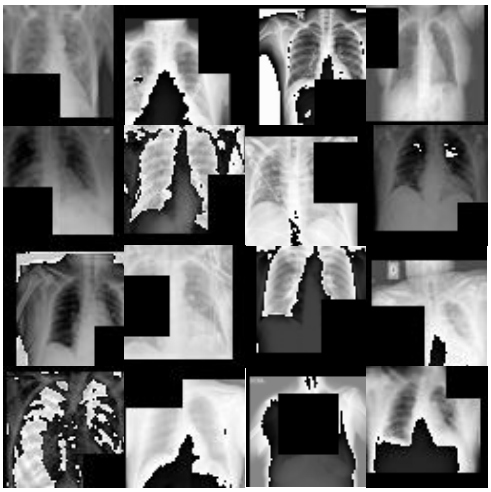


Figure 4.5: Sample of images transformed by DiffAugment.

4.7.1 Tackling class Imbalance

This section explores and analyses the results obtained with the augmentation method for tackling class imbalance, where every class has at least t images.

Baseline

Table 4.3 summarizes the results obtained with no augmentation, i.e. with $t = 0$. The class imbalance problem is clearly illustrated. While the Normal class, i.e. the majority class, has a high recall, with a value of 87.04%, the remaining minority classes, especially Covid and Viral Pneumonia, have a recall value far lower, of 42.22% and 47.58%, respectively. The Lung Opacity class has a recall value higher than Covid and Viral Pneumonia, which is to be expected, given that it has a far greater number of images. Surprisingly, the Covid class has a lower recall than Viral Pneumonia, despite having almost three times as many images. This observation may indicate that the Covid class is especially hard to classify, or that the Viral Pneumonia class is particularly easy. The class imbalance problem heavily affects the overall accuracy of the classifier, with a low value of 69.79%.

Accuracy (%)	Recall (%)			
	Covid	Lung Opacity	Normal	Viral Pneumonia
69.79	42.22	62.00	87.04	47.58

Table 4.3: Evaluation of baseline classifier.

Transformation-based Augmentation:

Table A.1 summarizes the results with simple transformations, while Table 4.4 highlights the 3 best and worst performing classifiers, according to accuracy. It is immediately clear that the best-performing transformation was the perspective shift, especially with larger threshold values. The best result was obtained with a perspective shift of magnitude 13 and a threshold of 1000 (with an accuracy score of 78.03%), while, curiously, the worst result was also obtained with a perspective shift, but with a magnitude of 7 and a threshold of 100.

t	Transformation	M	Accuracy (%)	Recall (%)			
				Covid	Lung Opacity	Normal	Viral Pneumonia
1000	Perspective	13	78.03	58.89	76.00	85.67	80.44
600	Perspective	7	77.94	65.56	71.33	86.05	79.12
1000	Perspective	7	77.94	57.78	76.67	85.08	83.52
400	Auto Contrast	13	69.13	30.56	65.67	85.85	61.21
400	Auto Contrast	7	69.13	30.56	65.67	85.85	61.21
100	Perspective	7	68.84	47.22	63.33	80.94	60.22

Table 4.4: 3 best and worst performing classifiers with simple transformations. t stands for threshold and M for magnitude.

It is not immediately clear the impact of magnitude on the resulting classifiers. Some transformations may benefit from larger magnitudes while others may deform the images to the point where they are no longer label-preserving. Figure 4.6 illustrates how magnitude influences the transformations and affects the resulting classifier performance. Some transformations, namely Auto Contrast, Equalize, Horizontal Flip, Posterize, and Vertical Flip, behave independently of the magnitude, while others vary heavily. Three photometric transformations, Brightness, Contrast, and Sharpness, tend to degrade performance at higher magnitudes, highlighting the fragility of x-ray images to color-space transformations. Gaussian blur, Rotate, Horizontal Shear, and Vertical Shear, benefit from larger magnitudes, but there is little to no improvement from 7 onwards. Solarize obtains similar results with low and medium magnitudes but sees a big improvement with larger values. Three transformations, Perspective, Horizontal Translation, and Vertical Translation, experience an increase in performance from magnitude 1 to magnitude 7, followed by a decrease when the magnitude is 13. At greater magnitudes, the range of these transformations is so large, that they may either distort the image heavily, in the case of Perspective, or remove relevant features, in the case of Horizontal and Vertical translations. While it would be expected for accuracy to increase with larger magnitudes and thresholds (since bigger ranges increase the image variety in the dataset), most transformations experience no benefit, or even suffer from, large magnitudes, suggesting that medium-level deformations are enough to obtain optimal performance.

Table A.2 summarizes the results with AutoAugment, while Table 4.5 highlights the 3 best and worst

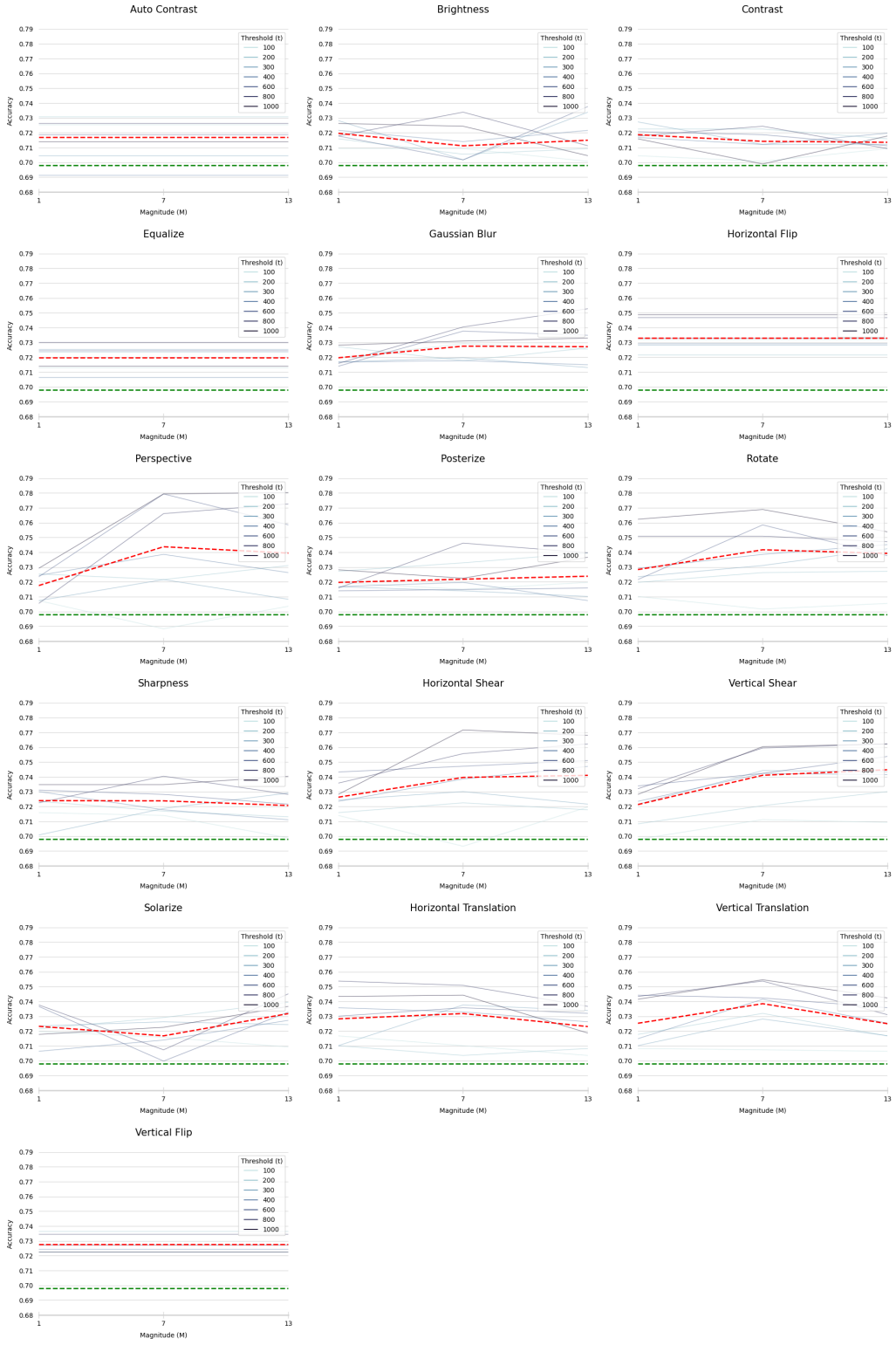


Figure 4.6: Accuracy with different transformations per magnitude value. Each blue line corresponds to one threshold value. The dashed green line represents the baseline value, while the dashed red line represents the trendline obtained with a locally weighted linear regression.

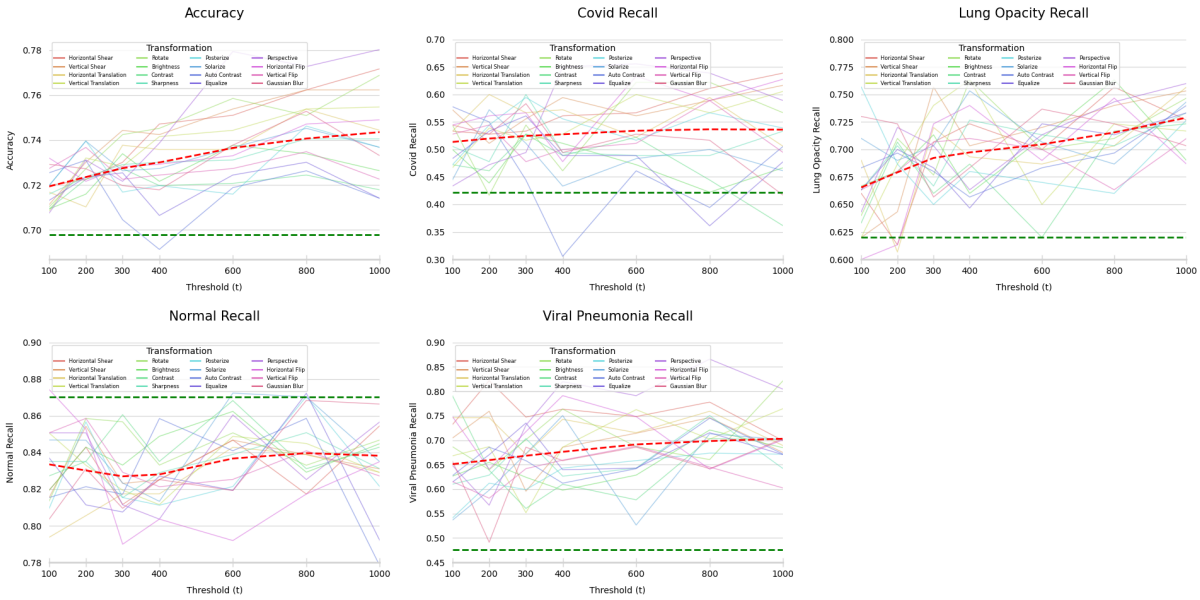


Figure 4.7: Accuracy and Recall for the best performing magnitude per transformation, per threshold value. The dashed green line represents the baseline value, while the dashed red line represents the trendline obtained with a locally weighted linear regression.

performing classifiers. Notably, the SVHN policy comprises most of the best and worst results, giving the impression that this policy scales better with larger threshold values. This observation is particularly clear in Figure 4.8, where SVHN starts by performing worse than the baseline (mainly because of the tradeoff between Normal and the remaining classes' recall) but is far better than the competing policies after the dataset is balanced, i.e. $t = 400$. This is due to the Cifar10 and ImageNet policies being mainly constructed from photometric transformations [5], which have been confirmed to be damaging at medium and large magnitudes, while the SVHN is mainly composed of geometric transformations, namely Rotation, Translation, Horizontal Shear, and Vertical Shear. AutoAugment performs better than the best-performing transformation, with an accuracy of 78.79%.

t	Policy	Accuracy (%)	Recall (%)			
			Covid	Lung Opacity	Normal	Viral Pneumonia
1000	SVHN	78.79	56.67	77.67	86.06	88.13
800	SVHN	76.89	64.44	78.33	79.77	82.09
600	SVHN	76.70	56.67	77.33	83.70	74.95
100	SVHN	69.70	48.33	63.67	81.53	64.12
100	ImageNet	69.32	41.67	64.33	84.87	47.58
200	SVHN	68.94	55.56	62.00	76.24	80.55

Table 4.5: 3 best and worst performing classifiers with AutoAugment. t stands for threshold.

Table A.3 summarizes the results with RandAugment, while Table 4.6 highlights the 3 best and worst

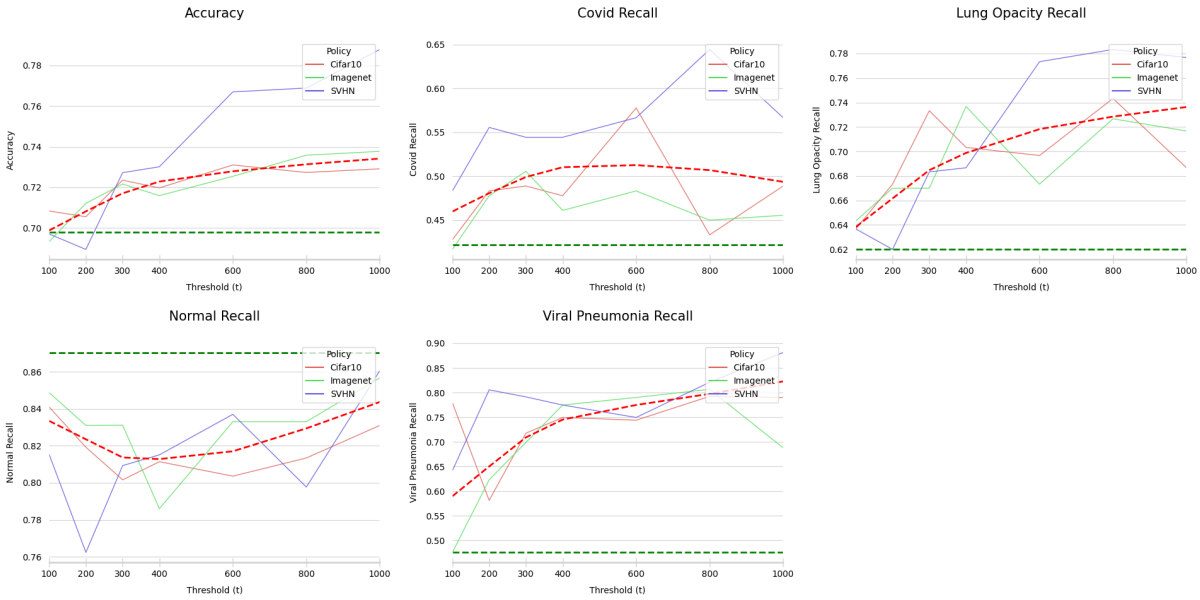


Figure 4.8: Accuracy and Recall for AutoAugment configured with different policies, per threshold value. The dashed green line represents the baseline value, while the dashed red line represents the trendline obtained with a locally weighted linear regression.

performing classifiers. It is not clear the correlation between the number of transformations and the resulting performance, but having larger sequence values seems to be beneficial at larger thresholds and damaging at lower ones. Figure 4.9 clearly illustrates this. At lower thresholds, lower sequence values outperform higher ones, while at larger thresholds the increased image variety provided by larger sequences improves performance. The best classifier with RandAugment, configured with a sequence and magnitude value of 3 and 9, respectively, achieves an accuracy score similar to the best-performing classifier with AutoAugment (with the differences being most likely due to rounding errors). These results follow the observations in [6], where RandAugment is comparable to AutoAugment but is much faster to compute.

t	N	M	Accuracy (%)	Recall (%)			
				Covid	Lung Opacity	Normal	Viral Pneumonia
1000	3	9	78.78	53.89	74.67	89.80	80.85
1000	4	9	78.60	58.89	76.00	86.45	83.63
800	4	5	78.50	58.89	81.00	82.92	86.37
100	4	9	67.42	43.33	59.33	80.56	69.12
100	4	5	67.23	37.22	57.67	82.50	74.73
100	2	13	66.86	35.00	62.00	45.82	86.37

Table 4.6: 3 best and worst performing classifiers with RandAugment. t stands for threshold, N for sequence, and M for magnitude.

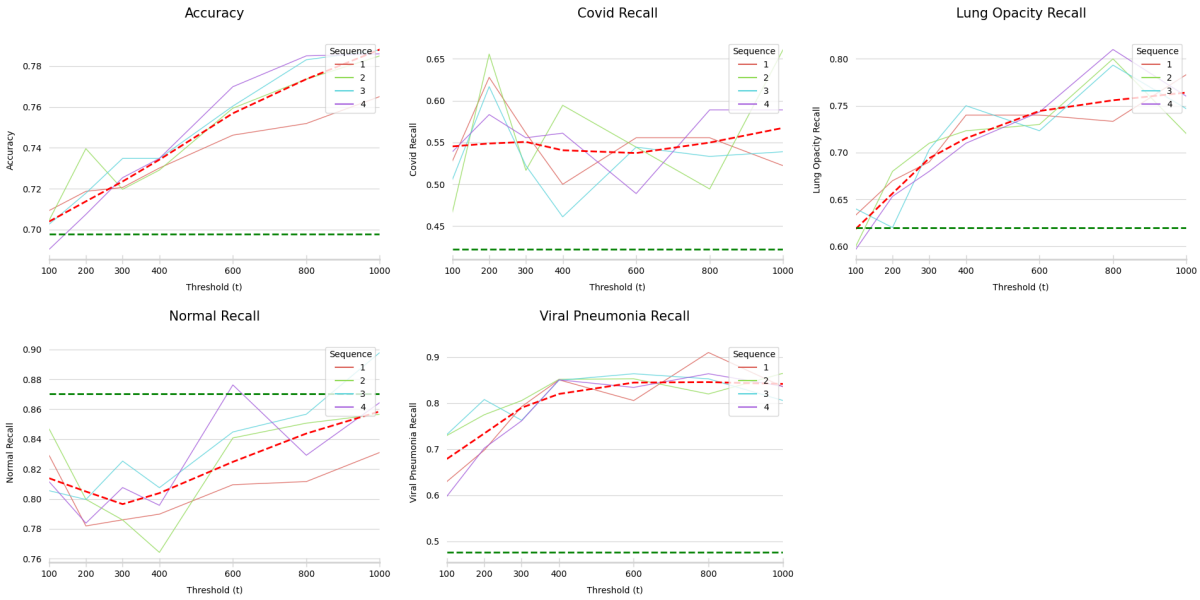


Figure 4.9: Accuracy and Recall for the best performing configuration of RandAugment, per sequence and threshold values. The dashed green line represents the baseline value, while the dashed red line represents the trendline obtained with a locally weighted linear regression.

Regarding the impact of the threshold value in the resulting classifier performance, four main conclusions can be drawn, based on Figures 4.7 to 4.9:

1. The accuracy increases with increasingly larger threshold values. For simple transformations, this behavior is transformation-specific, but an overall trend can be verified. For this type of image generation and AutoAugment, there appear to be diminishing returns with threshold values larger than 400, while with RandAugment the increase in performance is almost linear.

2. The accuracy increases even after the dataset is balanced, i.e. with $t > 400$, as the increased image variety improves generalizability and reduces overfitting. For $t <= 400$, the main improvement stems from the balancing mechanism.

3. The Normal recall tends to decrease until the dataset is balanced, as the classifier becomes less biased towards the majority class. While the Lung Opacity and Viral Pneumonia recall increase with larger thresholds, this is not verified for the Covid recall, which corroborates the hypothesis that the Covid class is particularly hard to classify.

4. Setting the threshold value to 100 provides immediate benefits to recall for most classes, suggesting that slightly reducing the class imbalance is enough to improve detection rates.

The best performance with transformation-based image generation methods was obtained with AutoAugment configured with the SVHN policy and a threshold of 1000, achieving 78.79% accuracy, which

corresponds to an improvement of 9% over the baseline classifier. The recall for Covid, Lung Opacity, and Viral Pneumonia increased by 14.45%, 15.67%, and 40.55%, respectively. These improvements indicate that adding transformed images to the dataset was an effective way of dealing with class imbalance.

Generative Augmentation:

With cGANAugment, the dataset is augmented with images generated by cGAN architecture. cGANAugment was configured with 7 architectures, ACGAN, ADCGAN, BigGAN, ContraGAN, cStyleGAN2, MHGAN, and ReACGAN. Table A.4 summarizes the results with this augmentation procedure, while Table 4.7 highlights the 3 best and worst performing classifiers. In contrast with transformation-based augmentation, the best results are not obtained with the largest thresholds, and the worst results with smaller ones. In fact, larger thresholds seem to negatively impact the accuracy score. This is confirmed in Figure 4.10, where the trend curve for accuracy increases until $t = 600$, where it stabilizes, followed by a decrease at $t = 1000$. Two architectures perform particularly poorly, ContraGAN and cStyleGAN2, where the accuracy is below the baseline value independently of the threshold. This observation is reflected in the remaining recall plots. None of the transformation-based methods exhibited this behavior, suggesting that a substantial problem is present in the images generated by these architectures. While with transformations the Lung Opacity and Viral Pneumonia recall values tended to increase with larger thresholds, with cGANAugment this is only verified for Lung Opacity, where it increases until $t = 400$, stabilizing after. Even though the Covid recall tended to behave independently of threshold, with cGANAugment there is a clear decrease at larger values. Once again, the Normal recall reflects the class imbalance problem, where it decreases until the dataset is balanced. This leads to the conclusion that the main benefit of using cGANAugment comes from reducing the bias toward the Normal class, and not from improving the detection rate of the pathologies after the dataset is balanced.

t	Architecture	Accuracy (%)	Recall (%)			
			Covid	Lung Opacity	Normal	Viral Pneumonia
600	ReACGAN	73.96	51.11	65.33	87.42	71.87
800	ADCGAN	73.87	47.78	70.67	86.06	65.60
600	ADCGAN	73.68	55.00	72.00	80.95	76.26
1000	cStyleGAN2	60.42	27.22	58.33	80.55	5.93
600	cStyleGAN2	58.14	23.89	56.00	78.38	6.04
800	cStyleGAN2	58.05	25.56	52.00	80.16	4.40

Table 4.7: 3 best and worst performing classifiers with cGANAugment. t stands for threshold.

The results obtained with cGANAugment are far inferior when compared to the transformation-based counterpart, achieving an accuracy score 4.83% lower. However, generative augmentation still improved

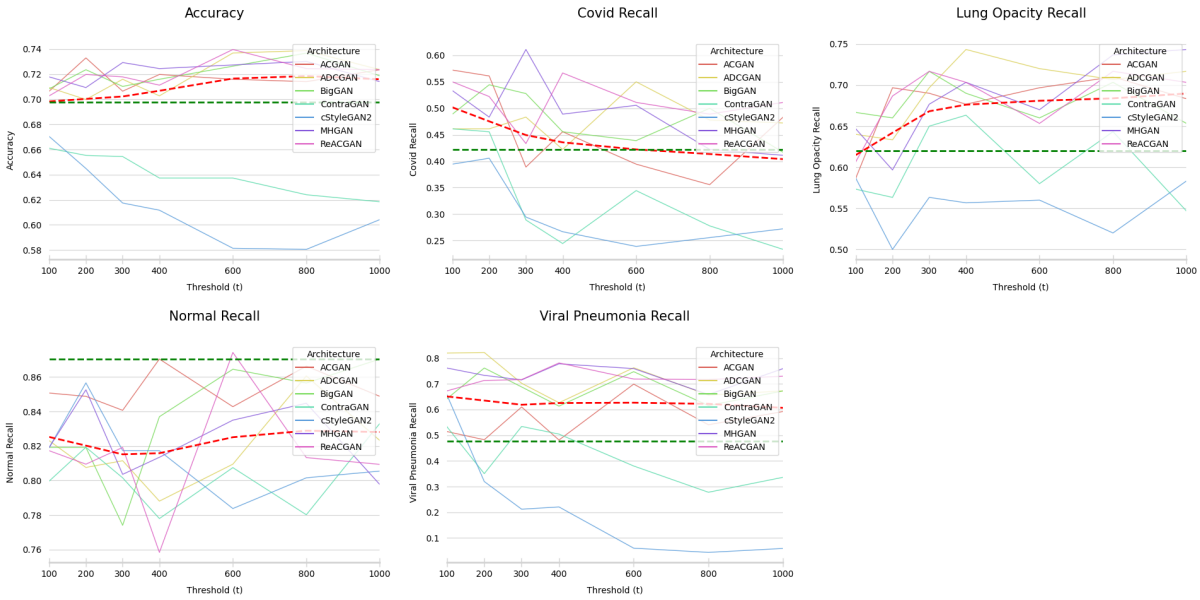


Figure 4.10: Accuracy and Recall for cGANAugment configured with different architectures, per threshold value. The dashed green line represents the baseline value, while the dashed red line represents the trend-line obtained with a locally weighted linear regression.

over the baseline, with an accuracy 4.17% higher. It was also effective in fixing the class imbalance problem, reflected in the higher recall values for Covid, Lung Opacity, and Viral Pneumonia.

4.7.2 Generalized augmentation

This section explores and analyses the results obtained with the generalized augmentation method, where every image shown to the classifier has a chance of being modified.

Baseline

It is immediately clear from Table 4.8 that starting from pretrained weights mitigates the class imbalance problem. Even though the majority class has the highest recall, the recall of the remaining classes is close to it (which was not verified in Table 4.3), indicating that no class imbalance is present. In addition to remedying class imbalance, transfer learning heavily improves the accuracy of the classifier, achieving an accuracy score, with no augmentation, 8.9% higher than the best-performing augmentation method in Section 4.7.1. Transfer learning cements itself as a powerful alternative when pretrained weights are available.

Data augmentation when no class imbalance is present aims to tackle data scarcity and improve the generalization ability of the classifiers. In addition to mitigating class imbalance, transfer learning remedies data scarcity, as the classifier starts from a point where it has seen a larger number of sam-

Accuracy (%)	Recall (%)			
	Covid	Lung Opacity	Normal	Viral Pneumonia
87.69	82.78	83.33	91.55	91.10

Table 4.8: Evaluation of baseline classifier.

ples than the ones in the training dataset. In this case, data augmentation is solely used to improve generalizability.

Transformation-based Augmentation

In contrast to the procedure to tackle class imbalance, this method does not simply insert new samples in the dataset. Instead, every image shown to the classifier has a chance of being modified. For simple transformations, this means that every image is deformed with probability p (in this work, p is set to 0.5). The way AutoAugment and RandAugment function stays the same, as they already account for probability.

Table A.5 summarizes the results with simple transformations and varying magnitude, while Table 4.9 highlights the 3 best and worst performing classifiers. The Perspective and Rotation transformations with a magnitude value of 13 outperform the remaining combinations. Notably, the best and worst results are obtained with high magnitude values, suggesting that some transformations scale well with larger values, while others are too deforming.

Transformation	M	Accuracy (%)	Recall (%)			
			Covid	Lung Opacity	Normal	Viral Pneumonia
Perspective	13	89.58	86.11	87.33	92.34	88.02
Rotate	13	89.58	88.89	86.00	91.95	89.56
Horizontal Translation	13	89.01	85.00	86.33	91.56	92.64
Sharpness	13	86.65	78.89	82.33	91.75	88.02
Posterize	13	86.55	78.89	82.00	91.56	89.56
Gaussian Blur	13	85.23	76.11	81.67	89.98	89.78

Table 4.9: 3 best and worst performing classifiers with simple transformations. M stands for magnitude.

Without the need to vary the threshold value, a more accurate evaluation of transformations can be conducted (see Figure 4.11). Here, only the impact of the magnitude affects performance, instead of the magnitude-threshold interaction and the impact of class imbalance. Most trends observed in Section 4.7.1 do not hold with this augmentation procedure. Firstly, most photometric transformations (Equalize, Gaussian Blur, Posterize, Sharpness, and Solarize) degrade performance, especially at larger magnitudes. This corroborates the preposition that medical images, and in particular, x-rays, are very sensitive to color-space deformations. Surprisingly, the Brightness and Contrast transformations im-

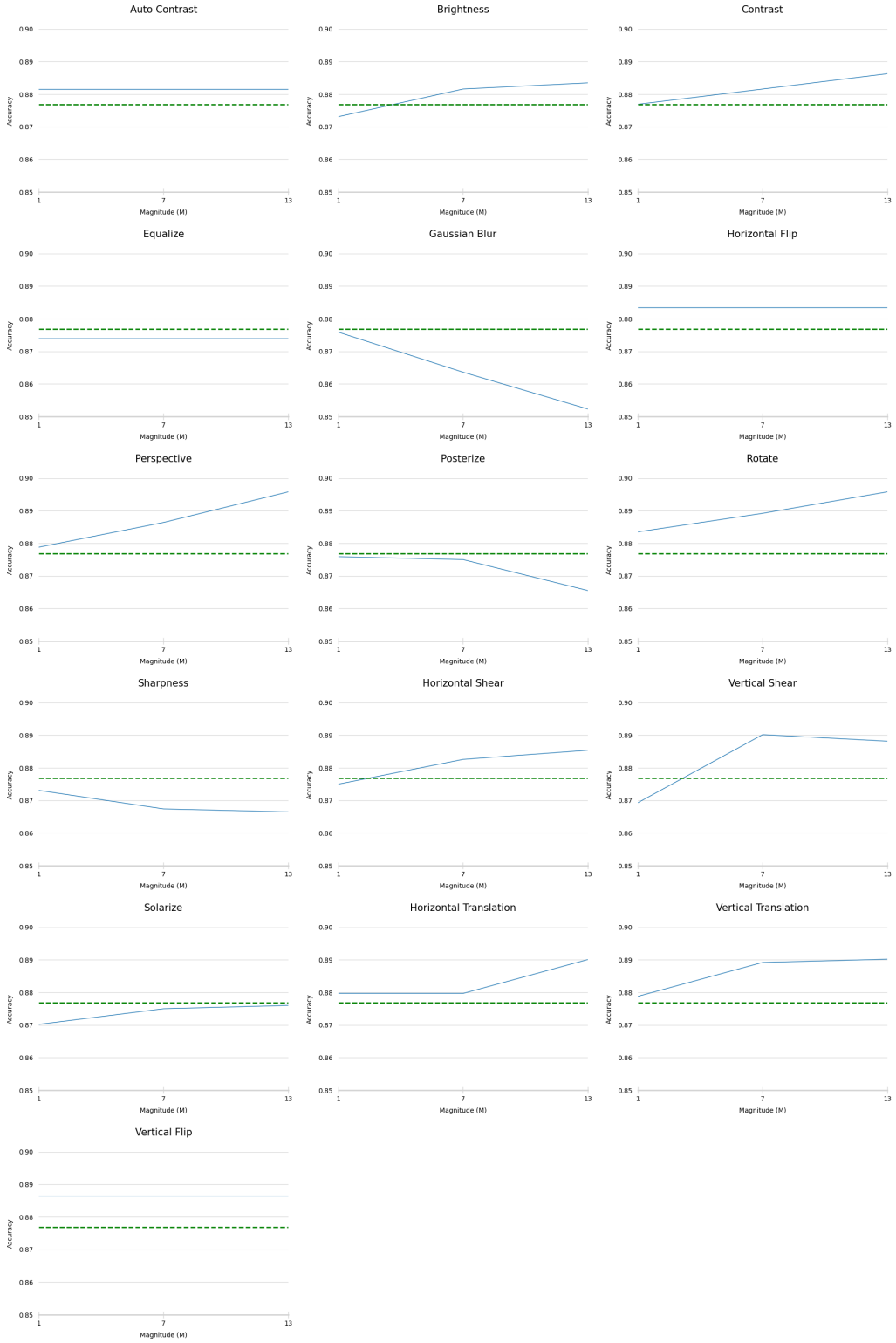


Figure 4.11: Accuracy of transformations with varying magnitude. The dashed green line represents the baseline value.

prove the accuracy score, especially at larger magnitudes. The Perspective, Horizontal Translation, and Vertical Translation transformations are no longer too deforming at larger magnitudes, increasing the accuracy score instead. A general trend can be observed, where geometric transformations improve performance with increasingly larger magnitudes, while most photometric transformations are consistently damaging.

The results with AutoAugment (see Table 4.10) differ heavily from the ones in Section 4.7.1. The SVHN policy ranks last in performance, while the ImageNet ranks first. It is not clear why this is happening, especially since the results for photometric transformations are even worse. One could hypothesize that a classifier pretrained on ImageNet is biased toward policies optimized for this dataset. The fact that most Cifar10 classes are also present in the ImageNet dataset could justify the Cifar10 policy ranking second. However, these are only hypotheses that have no experimental or theoretical basis and should be exclusively taken as an attempt to explain the conflicting results.

Policy	**Accuracy (%)**	**Recall (%)**			
		Covid	**Lung Opacity**	**Normal**	**Viral Pneumonia**
ImageNet	89.20	85.56	84.33	92.94	92.42
Cifar10	88.64	83.89	83.67	93.13	89.56
SVHN	88.54	80.00	86.33	92.34	92.64

Table 4.10: Evaluation of classifiers with AutoAugment.

In contrast to what is shown in Section 4.7.1, RandAugment outperforms AutoAugment. This may be explained by the fact that none of the AutoAugment policies are optimized for chest x-ray images. The amount of information that can be learned from suboptimal augmentation policies is limited, which is highlighted by the trend curve in Figure 4.8. With high accuracy levels, the benefits from AutoAugment are few, as the classifier may already know what AutoAugment introduces. The main benefit of RandAugment comes from the possibility of optimizing an image generation policy towards a particular dataset. Table A.6 summarizes the results with RandAugment, while Table 4.11 highlights the 3 best and worst performing classifiers. The best classification accuracy was obtained with 1 transformation of magnitude 5, with an accuracy of 89.96%, while the worst was obtained with 4 transformations of magnitude 13.

It is expected that applying a large number of high-magnitude transformations sequentially may deform the image heavily, and end up affecting performance. Figure 4.12 shows the relationship between sequence and magnitude. For larger sequences (3 and 4), the dip in performance is noticeable for magnitudes larger than 5, and, in the extreme case, may even degrade the baseline performance. Surprisingly, a drop in performance is also observed for smaller sequences, indicating an overall trend that performance is not optimal at large magnitudes, accentuated by longer sequences. This happens because the RandAugment augmentation space includes transformations such as Sharpness, Solarize,

N	M	Accuracy (%)	Recall (%)			
			Covid	Lung Opacity	Normal	Viral Pneumonia
1	5	89.96	86.67	86.67	93.32	87.91
1	9	89.93	88.33	84.00	92.74	91.10
2	9	89.30	82.22	87.33	92.74	90.99
3	13	87.88	76.11	87.33	92.15	89.45
1	1	87.40	81.67	84.00	91.16	89.45
4	13	86.55	68.33	83.00	94.90	88.02

Table 4.11: 3 best and worst performing classifiers with RandAugment. N stands for sequence and M for magnitude.

Posterize, and Equalize, that degrade performance at higher magnitudes.

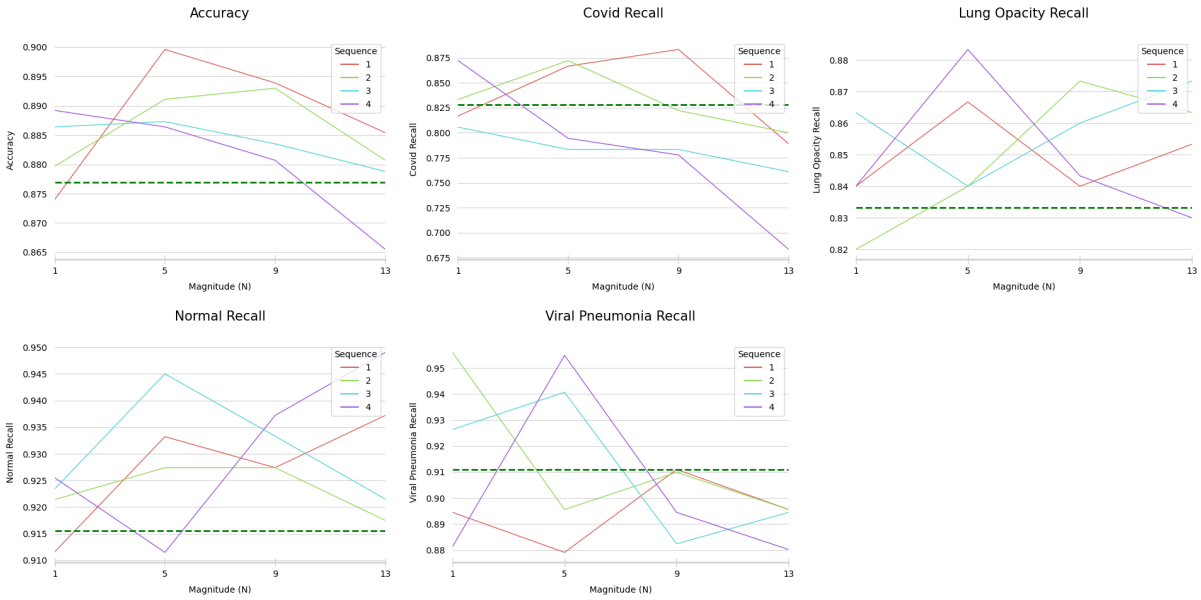


Figure 4.12: Accuracy and Recall for RandAugment, per sequence and magnitude values. The dashed green line represents the baseline value.

Applying transformation-based augmentation improved the accuracy score of a classifier pretrained on ImageNet. The best result was obtained with RandAugment, configured with a sequence of 1 and a magnitude of 5. This augmentation resulted in an improvement of 2.27% over the baseline classifier.

Generative Augmentation

Similarly to the transformation-based augmentations, an image shown to the classifier has a chance of being modified. In this case, the modification corresponds to swapping the image to one produced by cGANAugment with probability p, chosen from the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. The results of this procedure are summarized in Table A.7, while Table 4.12 highlights the 3 best and worst performing classifiers.

The best performing classifier was obtained with the ReACGAN architecture and $p = 0.5$ with an accuracy score of 89.11%, while the worst was obtained with the ContraGAN and $p = 0.9$.

Architecture	p	Accuracy (%)	Recall (%)			
			Covid	Lung Opacity	Normal	Viral Pneumonia
ReACGAN	0.5	89.11	83.33	87.00	92.14	91.10
ADCGAN	0.3	88.73	82.78	83.67	93.13	94.07
ReACGAN	0.1	88.73	84.44	86.00	91.16	94.07
ACGAN	0.9	82.86	65.56	80.67	89.60	88.13
cStyleGAN2	0.9	76.99	47.78	76.67	88.99	65.49
ContraGAN	0.9	75.09	56.67	66.67	86.64	74.07

Table 4.12: 3 best and worst performing classifiers with cGANAugment. p stands for probability.

The impact of p is clear in Figure 4.13. In the ACGAN, the accuracy is slightly above the baseline for every value of p but 0.9, where there is a steep drop. Since the ACGAN produces many similar images, little to no additional information is added at lower probabilities, while for larger values of p the real images are lost in a sea of similar-looking synthetic images. The ADCGAN, MHGAN, and ReACGAN were the most successful architectures, clearly improving the accuracy score over the baseline. The BigGAN also achieved higher accuracy scores, but to a lesser extent. Once again, the ContraGAN and cStyleGAN2 are damaging irrespective of p, following the results obtained in Section 4.7.1. An overall trend can be observed where the accuracy declines when the value of p is large, suggesting that there is a substantial problem in the images generated by the cGANs.

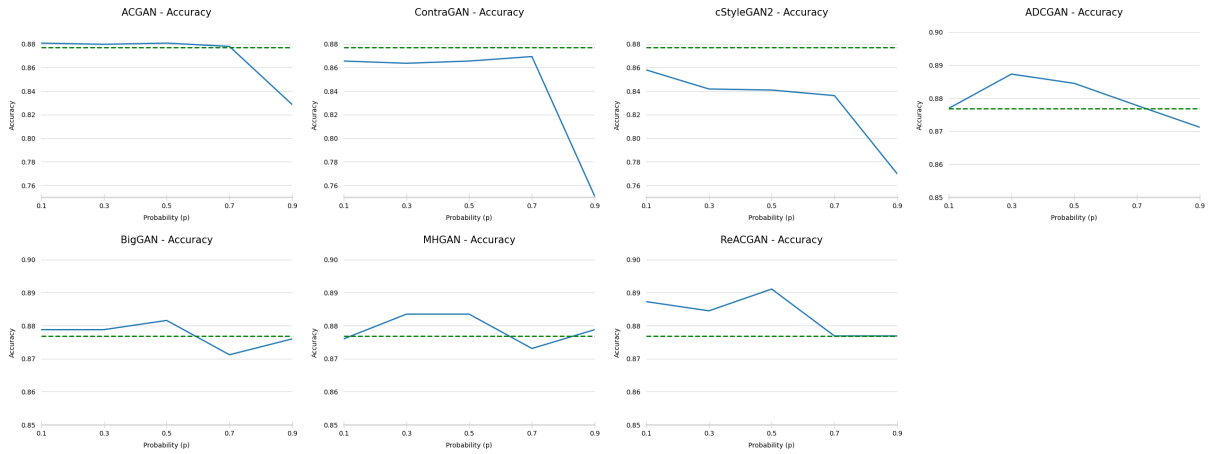


Figure 4.13: Evaluation of cGANAugment with varying architecture and probability. The dashed green line represents the baseline value.

Despite this, cGANAugment when configured with the correct image generation architecture still outperforms the baseline accuracy by 1.42%. Similarly to Section 4.7.1, cGANAugment does not outperform the best transformation-based augmentation, with an accuracy value 0.85% inferior. The reasoning

behind the suboptimal results is explained in detail in the following chapter.

4.8 Summary

Two image generation methods were tested, one with simple transformations, AutoAugment, and RandAugment, and one with the conditional architectures described in Section 2.2. Two augmentation procedures were considered, one that tackles class imbalance directly, by making sure that every class has at least t images (with increasingly larger values of t), and a more generalized method, where every image shown to the classifier has a chance of being modified.

The results from the cGAN evaluation indicated that the architectures, except the ProjGAN and ACGAN, were successful in generating high-fidelity images with a reasonable degree of diversity. Moreover, no overfit was detected.

With the first augmentation procedure, the baseline classifier clearly highlighted the class imbalance problem, where the Normal class had a very high recall, while the remaining classes had recall values far lower. The best performing classifier with transformation-based methods was obtained with AutoAugment configured with the SVHN policy and a threshold of 1000, with an accuracy score 9% higher than the baseline. With images generated by cGANs, the best-performing classifier achieved an accuracy score 4.17% higher than the baseline.

By starting from pretrained weights, not only was the class imbalance problem largely mitigated, but the accuracy of the baseline classifier was far greater. By applying RandAugment with a sequence of 1 and a magnitude of 5, the classifier achieved an accuracy score 2.27% higher than the baseline. With the ReACGAN architecture and a probability of 0.5, generative methods reached an accuracy score 1.42% higher than the baseline.

One thing was certain, independent of the augmentation procedure. cGANAugment did not outperform transformation-based methods.

5

Discussion

Contents

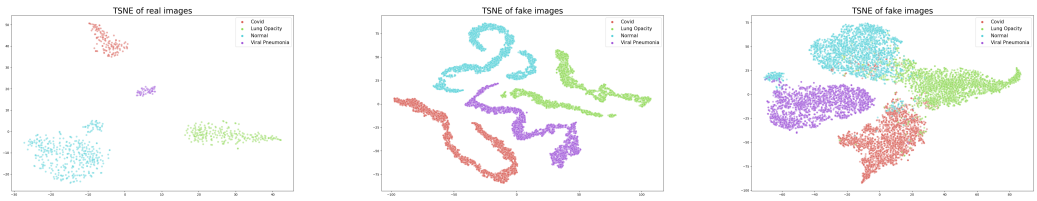
This section focuses on explaining the reasons for the suboptimal results with cGANAugment. It focuses mainly on the generalized augmentation procedure, as it is simpler and not affected by the threshold value t. Nonetheless, the reached conclusions can be extrapolated to the augmentation procedure for tackling class imbalance. The baseline classifier often referenced in this section is the one used for generalized augmentation.

5.1 The conditional generation problem

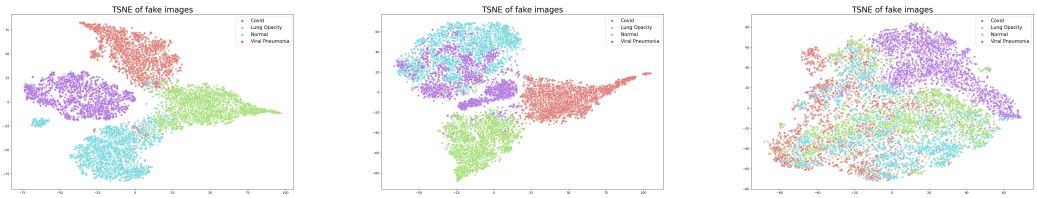
Conditional image generation is only as viable as its ability to correctly match the conditional data distribution. The development of cGANs has been mainly focused on improving image quality, and not on image diversity and the reliability of the conditional mechanism. Most cGANs perform very poorly on the benchmark dataset they were developed for (ImageNet), generating less than 50% of images correctly labeled [23]. When used exclusively for image generation, this problem is not as relevant since the incorrectly labeled images can be filtered out with a simple visual inspection. In medical imaging, this is not possible without the help of an expert. In addition, using incorrectly labeled images for augmentation is catastrophic, as we are continuously misleading the classifier.

One way to evaluate the ability of a conditional generator to produce images from the correct distribution is by analyzing their feature vectors on lower dimensions. For this, a dimensionality reduction algorithm, such as T-distributed Stochastic Neighbor Embedding (TSNE), can be used. Figure 5.1 illustrates the TSNE analysis of the conditional architectures, with the data points colored by class. The points were calculated over a set of feature vectors obtained by feeding 10000 synthetic images (2500 from each class) to the baseline classifier. As per Figure 5.1(a), it is clear by the degree of separation that the extracted features are representative of the various classes. The remaining plots present a far more entangled feature space. Most cGANs have some trouble correctly separating classes, with points in clusters they don't belong in, or parts of the clusters overlapping in certain areas. Two architectures, ContraGAN and cStyleGAN2, perform notoriously poorly in this assessment, with clusters overlapping or with no distinguishable cluster, respectively. This is the primary reason for the poor performance of cGANAugment with these architectures. The ACGAN is an interesting case where there is no overlap between clusters, but the data points are organized in band-like structures, and clusters from the same class present some degree of separation. It is not entirely clear as to why this happens, but one thing is certain. The lack of intra-class dispersion is due to the low image variety this cGAN produces, which may be linked to the peculiar shape of the clusters.

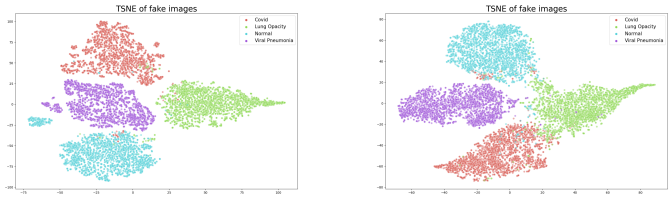
To help quantify the ratio of incorrectly labeled images, 10000 images from each class were labeled with the baseline classifier. This classifier obtains an accuracy score near 100% on the training dataset. Since the cGANs were trained on this data, the classifier serves as an accurate assessment of the



(a) Real. **(b)** ACGAN. **(c)** ADCGAN.



(d) BigGAN. **(e)** ContraGAN. **(f)** cStyleGAN2.



(g) MHGAN. **(h)** ReACGAN.

Figure 5.1: TSNE analysis of cGANs.

conditional mechanism. As per Table 5.1, it is immediately obvious that the ContraGAN and cStyleGAN2 cannot reliably generate images from the correct distribution. The ACGAN presents a high level of recall for the Normal class, with very low recall for Viral Pneumonia. The remaining architectures seem to be able to frequently generate correctly labeled images, with accuracy over 90%.

To highlight the impact of introducing incorrectly labeled images in the baseline classifier, the training dataset was modified so that p% of all images are incorrectly classified, with a random label. The value of p progressively increased from 5 to 20. As per Table 5.2, by just having $p = 5\%$ of the dataset incorrectly labeled, the accuracy of the baseline decreases by more than 1%. For higher values of p, the drop in performance is even more accentuated. This indicates that the number of incorrectly labeled images introduced in the dataset enforces an upper bound on the performance of cGANAugment.

It is expected for the baseline classifier to not correctly predict the label of the generated images every time. The purpose of cGANAugment is to introduce images in the dataset that the classifier has trouble labeling. However, it is nearly impossible to determine a priori how many generated images

Architecture	Accuracy (%)	Recall (%)			
		Covid	Lung Opacity	Normal	Viral Pneumonia
ACGAN	64.44	58.21	83.14	94.6	21.83
ADCGAN	89.81	85.78	82.76	98.56	92.12
BigGAN	94.24	96.83	86.65	94.43	99.03
ContraGAN	25.84	17.7	1.45	62.87	21.33
cStyleGAN2	4.35	9.51	5.82	2.02	0.04
MHGAN	95.93	98.15	91.91	96.64	97.02
ReACGAN	92.95	91.19	87.14	98.91	94.58

Table 5.1: Baseline classifier evaluated on generated images.

p (%)	Accuracy (%)	Recall (%)			
		Covid	Lung Opacity	Normal	Viral Pneumonia
0	87.69	82.78	83.33	91.55	91.10
5	86.08	81.67	81.67	90.18	86.59
10	84.85	73.89	82.33	90.57	81.87
15	82.48	76.67	77.67	88.21	76.04
20	78.41	61.67	77.33	87.63	58.79

Table 5.2: Evaluation of a classifier trained on a dataset with p% of images labeled incorrectly.

the baseline classifier should correctly label for cGANAugment to be optimal, as it depends on the images being generated. For a generator that produces hard-to-classify images, the accuracy will be low, but the images may be from the incorrect distribution. On the other hand, for a generator that produces easily classifiable images, the accuracy will be very high. This type of image is not particularly useful for augmentation, as they do not introduce information the classifier does not already know. One way to discover if easy-to-classify images are being generated is by examining the probability of the corresponding classification. As such, for each cGAN and class, a set of 10000 images were generated, classified, and the probability of the attributed label (defined by the softmax activation) was tracked. Table 5.3 indicates that for the cGANs with high accuracy scores, i.e. the ADCGAN, BigGAN, MHGAN, and ReACGAN, the vast majority of images being generated are already easily labeled by the classifier, as about 95% of all images are classified with a confidence value of over 90%. Surprisingly, the results for the ContraGAN and cStyleGAN2 are quite different, suggesting two different failure modes. The features generated by the ContraGAN are indicative of a class with high confidence values (even if the images are not generated according to the supposed label), while for the cStyleGAN2, the generated pathologies may be ambiguous, or not have enough quality to make an accurate decision.

In a perfect scenario, the cGANs would generate hard-to-classify images, with high-quality and varied features, and always respect the class label. In this case, the baseline classifier would be able to

Architecture	Class	Probability			
		[0.0, 0.7[	[0.7, 0.8[	[0.8, 0.9[	[0.9, 1.0]
ACGAN	Covid	7.44	4.17	7.26	81.14
	Lung Opacity	6.80	3.76	6.24	83.2
	Normal	5.35	2.64	3.82	88.2
	Viral Pneumonia	11.87	5.25	7.64	75.24
ADCGAN	Covid	2.5	1.46	2.18	93.86
	Lung Opacity	2.72	1.41	2.36	93.52
	Normal	1.94	1.15	1.9	95.0
	Viral Pneumonia	0.74	0.45	0.78	98.02
BigGAN	Covid	1.90	1.19	1.73	95.19
	Lung Opacity	1.99	1.53	2.16	94.32
	Normal	2.04	1.13	1.71	95.12
	Viral Pneumonia	0.35	0.2	0.36	99.09
ContraGAN	Covid	2.52	1.83	2.18	93.46
	Lung Opacity	2.28	1.46	2.15	94.12
	Normal	1.72	1.05	1.5	95.74
	Viral Pneumonia	0.94	0.55	1.01	97.49
cStyleGAN2	Covid	10.06	5.45	7.25	77.24
	Lung Opacity	11.48	5.73	7.69	75.11
	Normal	8.76	4.85	6.89	79.5
	Viral Pneumonia	6.58	3.59	5.16	84.66
MHGAN	Covid	2.02	1.24	1.82	94.92
	Lung Opacity	1.61	0.87	1.83	95.69
	Normal	1.57	0.79	1.36	96.28
	Viral Pneumonia	0.76	0.52	0.88	97.84
ReACGAN	Covid	2.76	1.55	2.33	93.35
	Lung Opacity	2.67	1.67	2.42	93.23
	Normal	2.21	1.37	2.09	94.32
	Viral Pneumonia	0.95	0.66	1.07	97.32

Table 5.3: Percentage of images, from a set of 10000, classified as class c with probability p, determined by a softmax activation.

learn new information with the produced images and consequently increase the detection rate of the pathologies. From the results above, it appears that this scenario is extremely hard to achieve, if not impossible. As mentioned previously, one of the big critiques of the ACGAN was that it frequently produces easy-to-classify images (and consequently has poor image diversity), as the auxiliary classifier would favor this type of generation [41, 39]. The remaining architectures solved the image variety problem, but not the easy-to-classify problem. The central issue revolves around the fact that the conditioning

mechanisms used in modern architectures, even though better than what is used in the ACGAN, are not as effective as the baseline classifier. Even with classifier-based architectures, the classification portion is not as deep. This means that the cGAN either produces many incorrectly labeled images, in the case of the ACGAN, ContraGAN, and cStyleGAN2, or images that almost always respect the class label, in the case of the ADCGAN, BigGAN, MHGAN, and ReACGAN, but are easy to classify. This is a consequence of the cGAN only modeling the data distribution that the conditioning mechanism correctly identifies, which is almost always smaller than what the baseline classifier can. If a cGAN was able to accurately generate images that the baseline classifier has trouble labeling, then, in some way, the conditioning mechanism of the cGAN would be superior. If this were the case then using the cGAN as a classifier would result in immediate benefits over the baseline classifier.

5.2 GAN evaluation based on features extracted by X-ray classifiers

Evaluating a cGAN in terms of image quality and diversity requires using metrics tailored for that purpose. These metrics are often calculated on feature vectors extracted by classifiers pretrained on ImageNet. For natural image datasets, the feature vectors end up providing a good enough representation of the generated images, resulting in accurate evaluations. In medical imaging, and especially with X-ray datasets, the features extracted by these general-purpose networks are too high-level to result in any meaningful evaluation. The metrics calculated with features extracted from the InceptionV3 model are no better than a visual evaluation by someone with no expertise in the radiology field. For augmentation purposes, it is not particularly interesting that, for example, high-level features, such as the ones captured by the untrained eye, are varied. What is important is that features that aid classification are generated with high quality and variety. As such, this section re-evaluates the cGANs on feature vectors extracted by the baseline classifier. This evaluation provides a more realistic assessment of the architectures given that, for a metric to be valid, it needs to always be calculated the same way, independently of the dataset.

Some metrics, especially FID, are tightly coupled with the value ranges of the feature vectors and changing the feature extractor results in very different scores. This means that no comparison of FID scores can be made with different feature extractors. On ImageNet, it is extremely unlikely that the conditional probability $p(y_k|x_i)$ (probability of assigning class k to image i) ever be 0, due to the sheer number of classes in the dataset. With the baseline classifier, this happens often enough for the IS metric to result in nan values. As such, every value of $p(y_k|x_i) < 1^{-6}$ is set to 1^{-6}. Table 5.4 summarizes the cGAN evaluation with features extracted by the baseline. The only conclusion that can be extracted from the FID scores is a simple ranking between architectures, as there is no way of telling how the

difference in FID is reflected in image quality. The ADCGAN generated the highest quality images, followed by the MHGAN, ContraGAN, ReACGAN, and BigGAN. The highest FID score belongs to the ProjGAN, which is expected given that the images are barely recognized as chest X-rays. The ACGAN and cStyleGAN2 generate images of similar quality. This result is unexpected and extremely interesting, as the cStyleGAN2 architecture generated the highest quality images when the features were extracted by the InceptionV3 network. As mentioned previously, the InceptionV3 network only extracts high-level features, which means that the cStyleGAN2 can accurately generate features captured by the untrained eye, such as general bone structure, shape, and size, but poor or imperceptible pathologies.

Architecture	**FID** ↓	**IS** ↑	**Improved Precision** ↑	**Improved Recall** ↑
ACGAN	464.74	2.79	0.53	0.00
ADCGAN	**156.25**	3.77	0.78	0.48
BigGAN	186.96	3.85	**0.82**	0.45
ContraGAN	170.90	3.45	0.76	0.47
cStyleGAN2	465.10	3.13	0.63	0.30
MHGAN	168.58	**3.88**	**0.82**	**0.50**
ProjGAN	782.10	1.45	0.60	0.01
ReACGAN	184.14	3.79	0.73	0.45

Table 5.4: Evaluation of the images generated by the cGANs, with features extracted by the baseline classifier.

Since the numeric values of the feature vectors are not directly used in the recall metric (the distance between feature vectors is used instead), a comparison between the results presented in Table 5.4 and the ones in Section 4.6 can be made. Immediately, the recall for the cStyleGAN2 is almost half of the value calculated with InceptionV3, and it is no longer the cGAN that generates the most diverse images. The ACGAN and ProjGAN still have a recall of 0, or very near it. The recall for the ReACGAN is now much closer to the remaining architectures. The diversity assessment for the BigGAN-based architectures is similar to what was shown in Section 4.6, with every cGAN achieving similar levels of recall. There does not seem to be a way of exactly predicting which architecture will do better in augmentation from the metrics alone, but lower FID and higher recall seem to correlate well with better results.

The IntraFID assessment presented in Table 5.5 highlights the inability of the ContraGAN and cStyleGAN2 to generate images from the correct distribution. For the most successful cGANs (ADCGAN, BigGAN, MHGAN, and ReACGAN) the values of Intra Fid for the Viral Pneumonia class are considerably lower than the remaining classes, even though having the lowest number of training images, which corroborates the observation that this class is particularly easy to model, and is represented well enough by the images in the dataset.

An overfitted GAN memorizes the dataset and generates simple copies of the training images, ob-

Architecture	Intra FID ↓			
	Covid	Lung Opacity	Normal	Viral Pneumonia
ACGAN	1025.59	718.08	324.57	597.80
ADCGAN	329.21	223.47	**104.91**	76.83
BigGAN	**163.77**	265.84	119.64	**33.70**
ContraGAN	1809.69	1804.90	452.54	866.25
cStyleGAN2	1379.18	1542.34	762.80	934.98
MHGAN	169.31	**204.34**	111.27	43.27
ProjGAN	1059.64	1344.88	656.93	708.64
ReACGAN	374.15	247.75	113.68	71.99

Table 5.5: Intra FID of the images generated by the cGANs, with features extracted by the baseline classifier.

taining very high accuracy scores and confidence values. As per Section 4.6, this did not seem to be the case, as the images were visually diverse, but the pathologies present may be too similar. Performing a visual evaluation (Figures B.5 and B.6) based on KNN with feature vectors extracted by the baseline classifier would indicate that at least the images are not being memorized, but nothing can be said about the features. As such, a different approach must be taken. For this, the Euclidean distance between the images in the dataset and the 10 nearest synthetic images (from a set of 10000 of every class) was calculated and averaged. The average distance between the images in the dataset and the 10 nearest real images can be used as a basis for comparison. As shown in Table 5.6, there does not seem to be any signs of overfitting. The distances for the ACGAN, ContraGAN, cStyleGAN2, and ProjGAN are considerably higher than the baseline, due to the aforementioned issues. For the remaining GANs, the distance is smaller, but not small enough to indicate memorization. The distance being smaller is a good sign, meaning that the cGANs can generate feature vectors that fill the holes in the distribution of the training data. The distance for the Viral Pneumonia class is much lower than the remaining classes, indicating low variance. This could be one of the reasons why the cGANs seem to be able to accurately model the distribution of this class.

5.3 Summary

Most cGANs have trouble correctly separating classes, which means that a considerable number of the generated images has the incorrect label associated. These images continuously mislead the classifier, affecting its performance. Two architectures performed particularly poorly in this assessment, the ContraGAN and cStyleGAN2. The remaining architectures seemed to generate correctly labeled images at a rate of over 90%. However, adding just 5% of incorrectly labeled images to the dataset was enough to degrade the performance of the baseline classifier.

Images	Distance			
	Covid	**Lung Opacity**	**Normal**	**Viral Pneumonia**
Real	15.10	18.44	17.99	9.14
ACGAN	22.46	19.99	18.96	11.39
ADCGAN	13.71	17.05	17.11	7.24
BigGAN	13.30	17.09	17.27	7.07
ContraGAN	26.72	29.44	18.69	16.25
cStyleGAN2	26.42	26.80	22.64	24.73
MHGAN	13.12	16.90	17.10	6.88
ProjGAN	28.82	28.71	23.44	22.13
ReACGAN	13.95	17.11	17.36	7.61

Table 5.6: Average Euclidean distance between the images in the dataset and the 10 nearest feature vectors, with features extracted by the baseline classifier. The distances between the images in the dataset and the 10 nearest real feature vectors serve as the basis for comparison.

High classification accuracy may be a symptom of a cGAN that generates many easy-to-classify images, that do not improve the robustness of the baseline classifier. It was found that 95% of all generated images were classified with a confidence value of over 90%.

GAN evaluation based on features extracted by ImageNet classifiers does not provide an accurate assessment of image quality and diversity, as the features are too high-level to serve as a meaningful representation. Several conflicting results were observed between the evaluation conducted with features extracted by X-ray classifiers and the evaluation conducted with features extracted by ImageNet classifiers.

6

Conclusion

Contents

The automatic detection and diagnosis of diseases from medical images faces a set of challenges not present in other domains. In medical datasets, the number of images is often not sufficient to build high-quality classification models. In addition, the resulting classifiers may be biased towards the classes with no pathologies, as they contain an order of magnitude higher number of images. Usually, this problem is dealt with by leveling the number of images in every class through image generation methods. Traditionally, only simple transformations have been used, but generative models may offer a better solution. Conditional Generative Adversarial Networks (cGANs) are a type of generative model that performs conditional image generation and produces never before seen high-quality synthetic images. With this type of generation, the minority classes can be inflated with synthetic images, even when there is not enough data to train a non-conditional model.

This work set out to explore the applicability of cGANs as an image generation technique, by considering two augmentation methods, 8 architectures, and several traditional augmentations. Across the two augmentation methods, it was clear that generative augmentation was not capable of outperforming existing transformations, independently of the architecture employed. It was found that the conditioning mechanism in the cGANs severely limits the effectiveness of the method. Firstly, the conditioning mechanisms are not reliable, resulting in many incorrectly labeled images. These images continuously mislead the classifier, limiting its performance. Just adding 5% of incorrectly labeled images is enough to decrease the accuracy score substantially. Secondly, the effectiveness of the conditioning mechanism limits the type of images produced. These mechanisms favor a narrow portion of the real image distribution, resulting in many easily classifiable images. Such images are not interesting for augmentation, as they do not introduce additional information and, consequently, do not improve the robustness of classifiers.

Two types of image quality and diversity assessments were performed, one with features extracted by the InceptionV3 network pretrained on ImageNet, and one with features extracted by a classifier pretrained on the dataset. Despite being the norm in natural image problems, the first evaluation method provides inaccurate results, as the features extracted are too high-level to produce any meaningful representation. The second method can reveal details and aspects not previously possible, and often results in conflicting conclusions, especially regarding image quality.

6.1 Future Work

From the presented results, it is clear that conditional image generation with cGANs intended for augmentation purposes is limited in its current form, and is usually outperformed by traditional augmentations. This section provides possible ways to solve the aforementioned issues.

There will always be some number of images that are generated with the incorrect label unless

the dataset is extremely easy to classify. Instead of aiming for perfect accuracy, one should focus on minimizing the number of incorrectly labeled images inserted in the dataset. A possible solution would be to cluster the synthetic images and remove the data points in the incorrect clusters and their boundary so that a predetermined distance between clusters is set. With this method, the separation between clusters would increase, and the number of incorrectly labeled images would hopefully decrease. Regarding easily classifiable images, there could be two possible solutions. Firstly, one could train the cGAN in such a way that it produces images that a baseline classifier has trouble labeling. One way to achieve this would be to add a term to the cGAN loss function that penalized images that the baseline classifier labels with high confidence. Secondly, one could simply use the trained cGANs and filter out every image that is labeled with high confidence values. However, this would result in a lot of images being filtered, which could slow down the training process to unacceptable levels.

Another way of solving the conditional generation problem would be to entirely drop the conditional portion and instead focus on training one GAN per class. As mentioned previously, training a GAN with small datasets could result in overfitting. From preliminary results with the FastGAN, a GAN intended for few-shot generation, this seems to be the case, but other architectures and configurations could be tried.

Bibliography

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[2] C. Bowles, L. Chen, R. Guerrero, P. Bentley, R. Gunn, A. Hammers, D. A. Dickie, M. V. Hernández, J. Wardlaw, and D. Rueckert. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[4] M. E. H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. A. Emadi, M. B. I. Reaz, and M. T. Islam. Can ai help in screening viral and covid-19 pneumonia? *IEEE Access*, 8:132665–132676, 2020. doi: 10.1109/ACCESS.2020.3010287.

[5] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[6] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[7] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. Courville. Modulating early visual processing by language. *arXiv preprint arXiv:1707.00683*, 2017.

[8] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *arXiv preprint arXiv:1506.05751*, 2015.

[9] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, 321:321–331, 2018.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[12] C. Han, L. Rundo, R. Araki, Y. Furukawa, G. Mauri, H. Nakayama, and H. Hayashi. Infinite brain mr images: Pggan-based data augmentation for tumor detection. In *Neural approaches to dynamics of signal exchanges*, pages 291–303. Springer, 2020.

[13] T. Han, S. Nebelung, C. Haarburger, N. Horst, S. Reinartz, D. Merhof, F. Kiessling, V. Schulz, and D. Truhn. Breaking medical data sharing boundaries by employing artificial radiographs. *bioRxiv*, page 841619, 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[16] L. Hou, Q. Cao, H. Shen, S. Pan, X. Li, and X. Cheng. Conditional gans with auxiliary discriminative classifier. In *International Conference on Machine Learning*, pages 8888–8902. PMLR, 2022.

[17] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[18] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[20] J. M. Johnson and T. M. Khoshgoftaar. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019.

[21] M. Kang and J. Park. Contragan: Contrastive learning for conditional image generation. *arXiv preprint arXiv:2006.12681*, 2020.

[22] M. Kang, W. Shim, M. Cho, and J. Park. Rebooting acgan: Auxiliary classifier gans with stable training. *Advances in Neural Information Processing Systems*, 34, 2021.

[23] M. Kang, J. Shin, and J. Park. Studiogan: A taxonomy and benchmark of gans for image synthesis. *arXiv preprint arXiv:2206.09479*, 2022.

[24] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[25] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[26] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020.

[27] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. 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.

[28] I. Kavalerov, W. Czaja, and R. Chellappa. A multi-class hinge loss for conditional gans. 2021.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] S. Kora Venu and S. Ravula. Evaluation of deep convolutional generative adversarial networks for data augmentation of chest x-ray images. *Future Internet*, 13(1):8, 2021.

[31] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

[32] M. Lee and J. Seok. Regularization methods for generative adversarial networks: An overview of recent studies. *arXiv preprint arXiv:2005.09165*, 2020.

[33] J. H. Lim and J. C. Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017.

[34] B. Liu, Y. Zhu, K. Song, and A. Elgammal. Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In *International Conference on Learning Representations*, 2020.

[35] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are gans created equal? a large-scale study. *arXiv preprint arXiv:1711.10337*, 2017.

[36] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[37] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

[38] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[39] T. Miyato and M. Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

[40] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[41] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[42] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[43] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[44] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[45] V. Sandfort, K. Yan, P. J. Pickhardt, and R. M. Summers. Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. *Scientific reports*, 9(1):1–9, 2019.

[46] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[47] E. Schonfeld, B. Schiele, and A. Khoreva. A u-net based discriminator for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8207–8216, 2020.

[48] T. R. Shaham, T. Dekel, and T. Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4570–4580, 2019.

[49] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[50] S. Sundaram and N. Hulkund. Gan-based data augmentation for chest x-ray classification. *arXiv preprint arXiv:2107.02970*, 2021.

[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[52] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[53] N.-T. Tran, V.-H. Tran, N.-B. Nguyen, T.-K. Nguyen, and N.-M. Cheung. On data augmentation for gan training. *IEEE Transactions on Image Processing*, 30:1882–1897, 2021.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[55] A. Waheed, M. Goyal, D. Gupta, A. Khanna, F. Al-Turjman, and P. R. Pinheiro. Covidgan: data augmentation using auxiliary classifier gan for improved covid-19 detection. *Ieee Access*, 8:91916–91923, 2020.

[56] A. Yadav, S. Shah, Z. Xu, D. Jacobs, and T. Goldstein. Stabilizing adversarial nets with prediction methods. *arXiv preprint arXiv:1705.07364*, 2017.

[57] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[58] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[59] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. Differentiable augmentation for data-efficient gan training. *Advances in Neural Information Processing Systems*, 33:7559–7570, 2020.

[60] J.-Y. Zhu, T. Park, P. Isola, and A. 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.

A

Extended Augmentation Results

A.1 Tackling class imbalance

Threshold (t)	Transformation	Magnitude (M)		
		1	7	13
100	Auto Contrast	72.54	72.54	72.54
	Brightness	70.93	70.93	70.07
	Contrast	70.45	70.07	70.93
	Equalize	71.31	71.31	71.31
	Gaussian Blur	71.97	72.92	72.54
	Horizontal Flip	73.20	73.20	73.20
	Perspective	70.74	68.84	70.36
	Posterize	71.97	71.50	71.97
	Rotate	71.02	70.17	70.55
	Sharpness	71.59	71.40	69.88
	Horizontal Shear	71.40	69.32	71.97
	Vertical Shear	69.79	71.12	70.93
	Solarize	71.97	71.59	70.93
	Horizontal Translation	71.68	71.02	70.36
	Vertical Translation	70.83	70.74	70.64
	Vertical Flip	72.73	72.73	72.73
200	Auto Contrast	73.10	73.10	73.10
	Brightness	71.59	70.55	70.93
	Contrast	72.25	72.25	71.59
	Equalize	72.35	72.35	72.35
	Gaussian Blur	72.72	71.78	72.63
	Horizontal Flip	72.16	72.16	72.16
	Perspective	72.54	72.16	73.11
	Posterize	72.72	73.29	73.96
	Rotate	71.97	72.63	72.72
	Sharpness	72.35	71.69	71.31
	Horizontal Shear	71.59	72.25	71.78
	Vertical Shear	70.83	72.06	73.01
	Solarize	72.16	72.92	73.95
	Horizontal Translation	71.02	70.36	70.83
	Vertical Translation	71.78	73.20	71.68
	Vertical Flip	73.67	73.67	73.67
300	Auto Contrast	70.45	70.45	70.45
	Brightness	72.82	70.17	73.39
	Contrast	72.73	71.21	71.21
	Equalize	72.54	72.54	72.54

	Gaussian Blur	71.68	71.97	71.31
	Horizontal Flip	72.82	72.82	72.82
	Perspective	70.74	72.16	70.83
	Posterize	71.68	71.40	71.02
	Rotate	72.35	73.11	74.24
	Sharpness	70.08	71.87	72.91
	Horizontal Shear	72.44	73.01	72.16
	Vertical Shear	72.16	74.43	74.34
	Solarize	72.35	72.63	72.44
	Horizontal Translation	71.02	73.77	73.39
	Vertical Translation	71.02	72.82	71.69
	Vertical Flip	72.25	72.25	72.25
400	Auto Contrast	69.13	69.13	69.13
	Brightness	72.16	71.40	72.16
	Contrast	71.68	71.21	71.97
	Equalize	70.64	70.64	70.64
	Gaussian Blur	71.68	71.78	71.50
	Horizontal Flip	72.92	72.92	72.92
	Perspective	72.44	73.86	72.63
	Posterize	71.68	71.97	70.74
	Rotate	72.92	73.86	74.52
	Sharpness	73.01	71.78	71.11
	Horizontal Shear	72.35	73.86	74.71
	Vertical Shear	72.35	74.24	74.15
	Solarize	70.64	71.40	72.73
	Horizontal Translation	73.58	73.30	72.63
	Vertical Translation	71.49	74.15	72.54
	Vertical Flip	72.44	72.44	72.44
600	Auto Contrast	71.88	71.88	71.88
	Brightness	71.78	70.17	73.77
	Contrast	72.07	71.87	71.12
	Equalize	72.44	72.44	72.44
	Gaussian Blur	71.40	73.77	73.49
	Horizontal Flip	73.30	73.30	73.30
	Perspective	72.35	77.94	75.85
	Posterize	71.40	71.49	71.59
	Rotate	72.16	75.85	73.77
	Sharpness	73.11	72.82	72.16
	Horizontal Shear	74.33	74.72	75.10
	Vertical Shear	73.39	74.24	75.38
	Solarize	73.67	69.98	73.29

	Horizontal Translation	73.01	73.58	73.20
	Vertical Translation	74.43	74.24	73.58
	Vertical Flip	72.73	72.73	72.73
800	Auto Contrast	72.63	72.63	72.63
	Brightness	71.78	73.39	71.12
	Contrast	71.78	72.44	70.93
	Equalize	73.01	73.01	73.01
	Gaussian Blur	71.59	74.05	75.28
	Horizontal Flip	74.71	74.71	74.71
	Perspective	70.55	76.61	77.27
	Posterize	71.59	74.62	73.96
	Rotate	75.09	75.09	74.71
	Sharpness	72.25	74.05	72.82
	Horizontal Shear	73.58	75.57	76.23
	Vertical Shear	73.20	75.95	76.23
	Solarize	73.77	70.74	74.52
	Horizontal Translation	75.38	75.09	73.67
	Vertical Translation	74.34	75.38	73.11
	Vertical Flip	73.48	73.48	73.48
1000	Auto Contrast	71.40	71.40	71.40
	Brightness	72.63	72.44	70.45
	Contrast	71.59	69.89	71.78
	Equalize	71.40	71.40	71.40
	Gaussian Blur	72.82	73.10	73.30
	Horizontal Flip	74.90	74.90	74.90
	Perspective	72.92	77.94	78.03
	Posterize	72.82	72.25	73.67
	Rotate	76.23	76.89	75.38
	Sharpness	73.48	73.48	74.05
	Horizontal Shear	72.82	77.17	76.80
	Vertical Shear	72.82	76.04	76.23
	Solarize	71.78	72.26	73.67
	Horizontal Translation	74.34	74.43	71.87
	Vertical Translation	74.15	75.47	74.24
	Vertical Flip	72.25	72.25	72.25

Table A.1: Classification accuracy (%) with simple transformations and varying threshold and magnitude.

Threshold (t)	Policy	Accuracy (%)
100	Cifar10	70.83
	Imagenet	69.32
	SVHN	69.70
200	Cifar10	70.55
	Imagenet	71.21
	SVHN	68.94
300	Cifar10	72.35
	Imagenet	72.16
	SVHN	72.72
400	Cifar10	71.97
	Imagenet	71.59
	SVHN	73.01
600	Cifar10	73.10
	Imagenet	72.54
	SVHN	76.70
800	Cifar10	72.73
	Imagenet	73.58
	SVHN	76.89
1000	Cifar10	72.91
	Imagenet	73.77
	SVHN	78.79

Table A.2: Classification accuracy (%) with AutoAugment and varying threshold and policy.

Threshold (t)	Sequence (N)	Magnitude (M)			
		1	5	9	13
100	0	69.89	70.93	69.70	68.65
	1	70.45	68.37	68.37	66.86
	2	70.26	68.37	69.32	68.56
	3	69.03	67.23	67.42	68.08
200	0	71.21	70.07	71.59	71.87
	1	73.96	71.69	72.25	72.54
	2	71.78	70.07	71.59	71.40
	3	68.94	69.98	70.74	70.17
300	0	71.78	71.02	72.06	71.12
	1	71.50	71.21	71.97	71.68
	2	71.88	73.48	71.97	73.39
	3	72.06	72.25	72.54	72.06
400	0	71.88	71.12	73.01	72.73
	1	71.21	72.91	72.63	72.73
	2	73.01	73.48	72.06	73.29
	3	71.78	73.48	73.10	72.82
600	0	72.63	73.01	74.62	74.52
	1	75.47	74.81	75.57	75.94
	2	75.66	74.72	76.04	75.85
	3	75.19	75.95	76.99	76.51
800	0	75.00	74.90	75.19	74.34
	1	75.94	76.99	77.36	76.23
	2	76.04	78.31	78.12	76.61
	3	75.94	78.50	76.99	77.75
1000	0	75.00	76.51	75.95	76.23
	1	75.19	78.31	77.84	78.50
	2	75.57	77.37	78.78	76.99
	3	76.80	78.41	78.60	77.84

Table A.3: Classification accuracy (%) with RandAugment and varying threshold, sequence, and magnitude.

Threshold (t)	Architecture	Accuracy (%)
100	ACGAN	70.64
	ADCGAN	70.93
	BigGAN	70.83
	ContraGAN	66.10
	cStyleGAN2	67.04
	MHGAN	71.78

	ReACGAN	70.27
200	ACGAN	73.29
	ADCGAN	69.98
	BigGAN	72.35
	ContraGAN	65.53
	cStyleGAN2	64.49
	MHGAN	70.93
	ReACGAN	71.97
300	ACGAN	70.64
	ADCGAN	71.59
	BigGAN	71.02
	ContraGAN	65.44
	cStyleGAN2	61.74
	MHGAN	72.92
	ReACGAN	71.78
400	ACGAN	71.97
	ADCGAN	70.27
	BigGAN	71.59
	ContraGAN	63.73
	cStyleGAN2	61.17
	MHGAN	72.44
	ReACGAN	71.12
600	ACGAN	71.59
	ADCGAN	73.68
	BigGAN	72.63
	ContraGAN	63.73
	cStyleGAN2	58.14
	MHGAN	72.73
	ReACGAN	73.96
800	ACGAN	71.40
	ADCGAN	73.87
	BigGAN	73.67
	ContraGAN	62.40
	cStyleGAN2	58.05
	MHGAN	73.01
	ReACGAN	72.44
1000	ACGAN	72.35
	ADCGAN	72.35
	BigGAN	71.87
	ContraGAN	61.84
	cStyleGAN2	60.42

MHGAN	71.40
ReACGAN	72.35

Table A.4: Classification accuracy (%) with cGANAugment and varying threshold and architecture.

A.2 Generalized Augmentation

Transformation	Magnitude		
	1	7	13
Auto Contrast	88.16	88.16	88.16
Brightness	87.31	88.16	88.35
Contrast	87.69	88.16	88.63
Equalize	87.40	87.40	87.40
Gaussian Blur	87.59	86.36	85.23
Horizontal Flip	88.35	88.35	88.35
Perspective	87.88	88.64	89.58
Posterize	87.59	87.50	86.55
Rotate	88.35	88.92	89.58
Sharpness	87.31	86.74	86.65
Horizontal Shear	87.50	88.26	88.54
Vertical Shear	86.93	89.02	88.82
Solarize	87.02	87.50	87.60
Horizontal Translation	87.97	87.97	89.01
Vertical Translation	87.88	88.92	89.02
Vertical Flip	88.64	88.60	88.64

Table A.5: Classification accuracy (%) with simple transformations and varying magnitude.

Sequence (N)	Magnitude (M)			
	1	5	9	13
1	87.40	89.96	89.39	88.54
2	87.97	89.11	89.30	88.07
3	88.64	88.73	88.35	87.88
4	88.92	88.64	88.0	86.55

Table A.6: Classification accuracy (%) with RandAugment and varying sequence and magnitude.

Architecture	Probability (p)				
	0.1	**0.3**	**0.5**	**0.7**	**0.9**
ACGAN	88.07	87.97	88.07	87.79	82.86
ADCGAN	87.69	88.73	88.45	87.78	87.12
BigGAN	87.88	87.88	88.16	87.12	87.60
ContraGAN	86.55	86.36	86.55	86.93	75.09
cStyleGAN2	85.80	84.18	84.09	83.62	76.99
MHGAN	87.60	88.35	88.35	87.31	87.88
ReACGAN	88.73	88.45	89.11	87.69	87.69

Table A.7: Classification accuracy (%) with cGANAugment and varying probability.

B

Extended GAN Evaluation

B.1 Sample of generated images

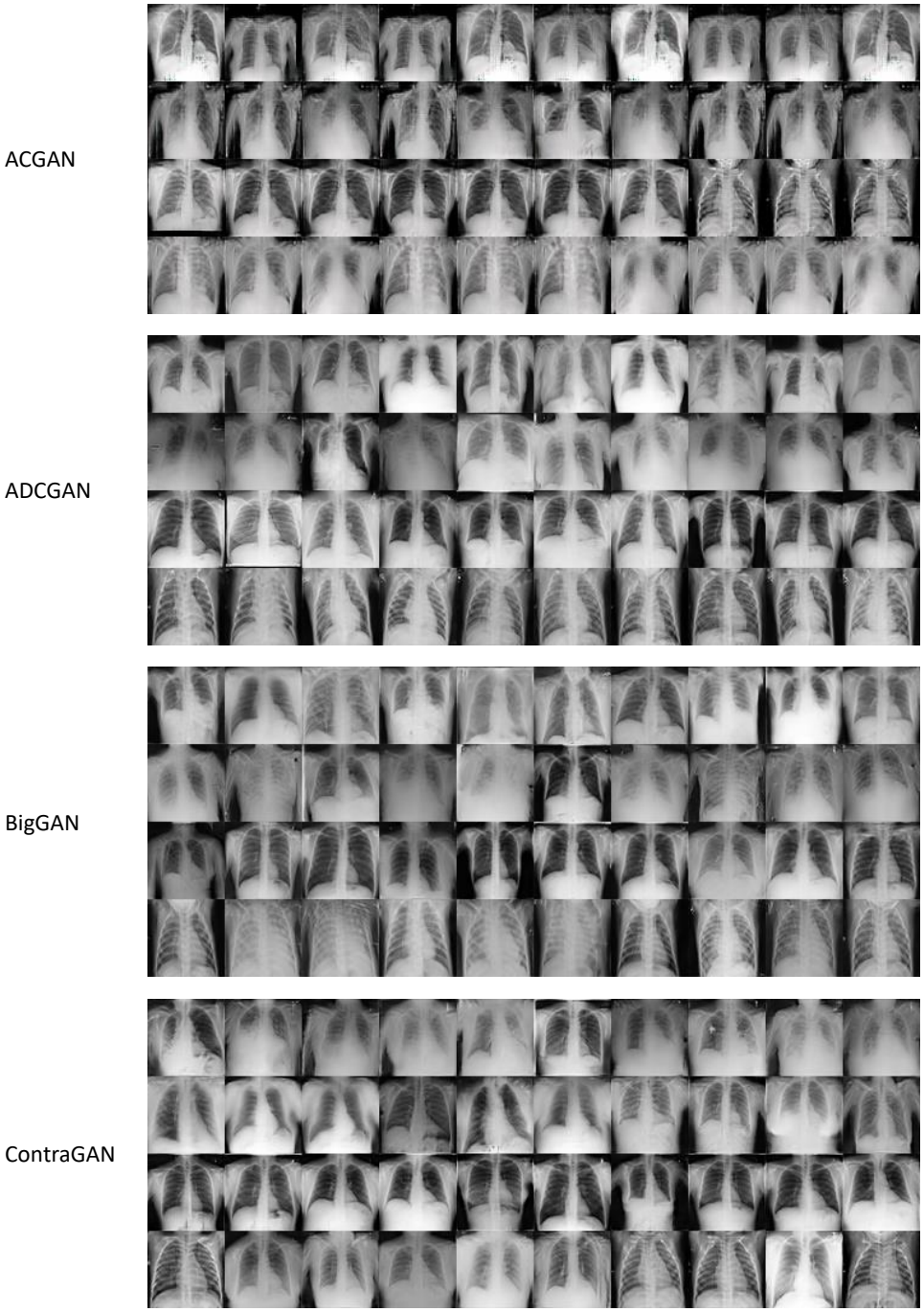


Figure B.1: Sample of images generated by the ACGAN, ADCGAN, BigGAN, and ContraGAN. Each row corresponds to one class, out of Covid, Lung Opacity, Normal, and Viral Pneumonia, respectively.

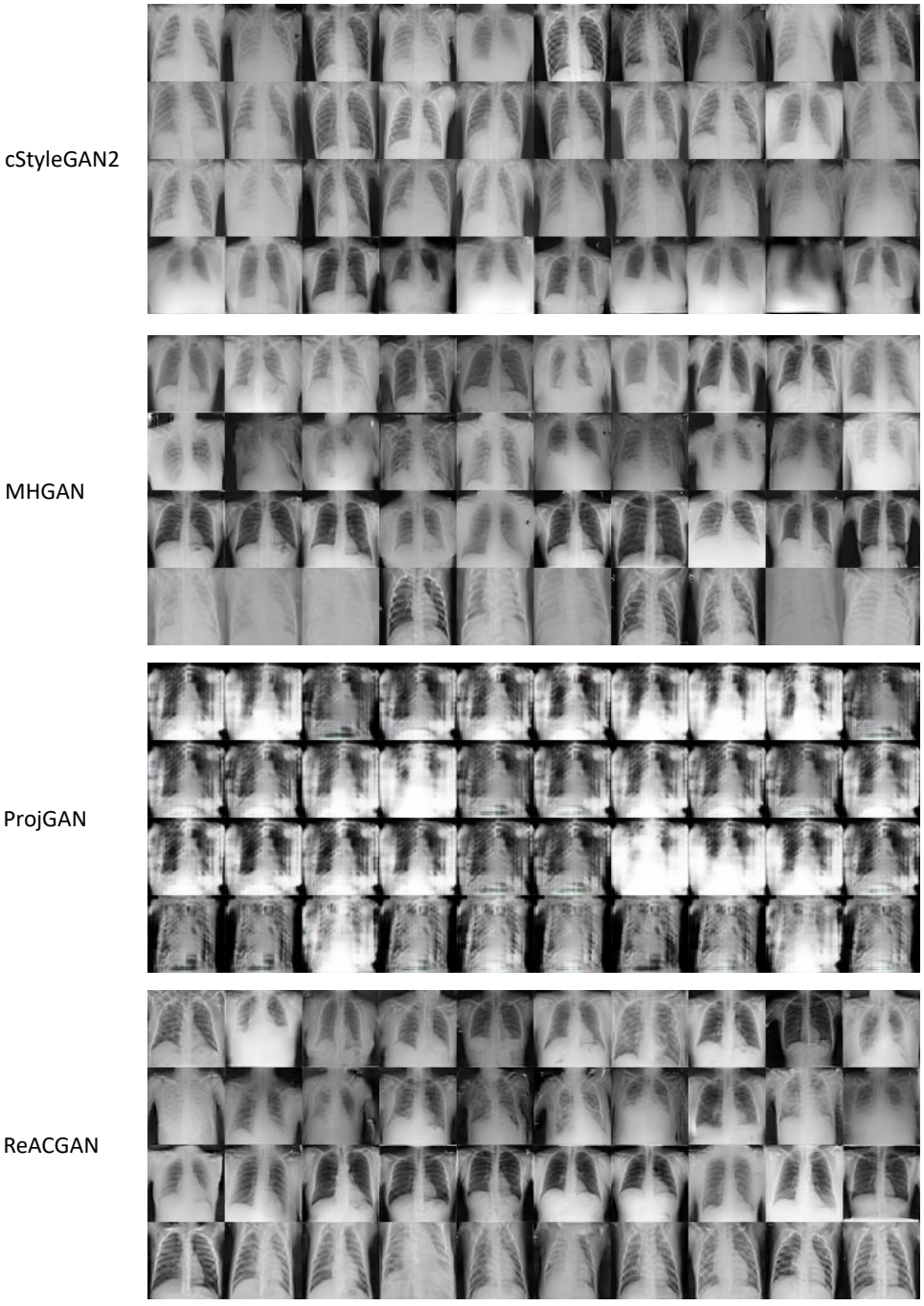


Figure B.2: Sample of images generated by the cStyleGAN2, MHGAN, ProjGAN, and ReACGAN. Each row corresponds to one class, out of Covid, Lung Opacity, Normal, and Viral Pneumonia, respectively.

B.2 KNN anaysis of generated images

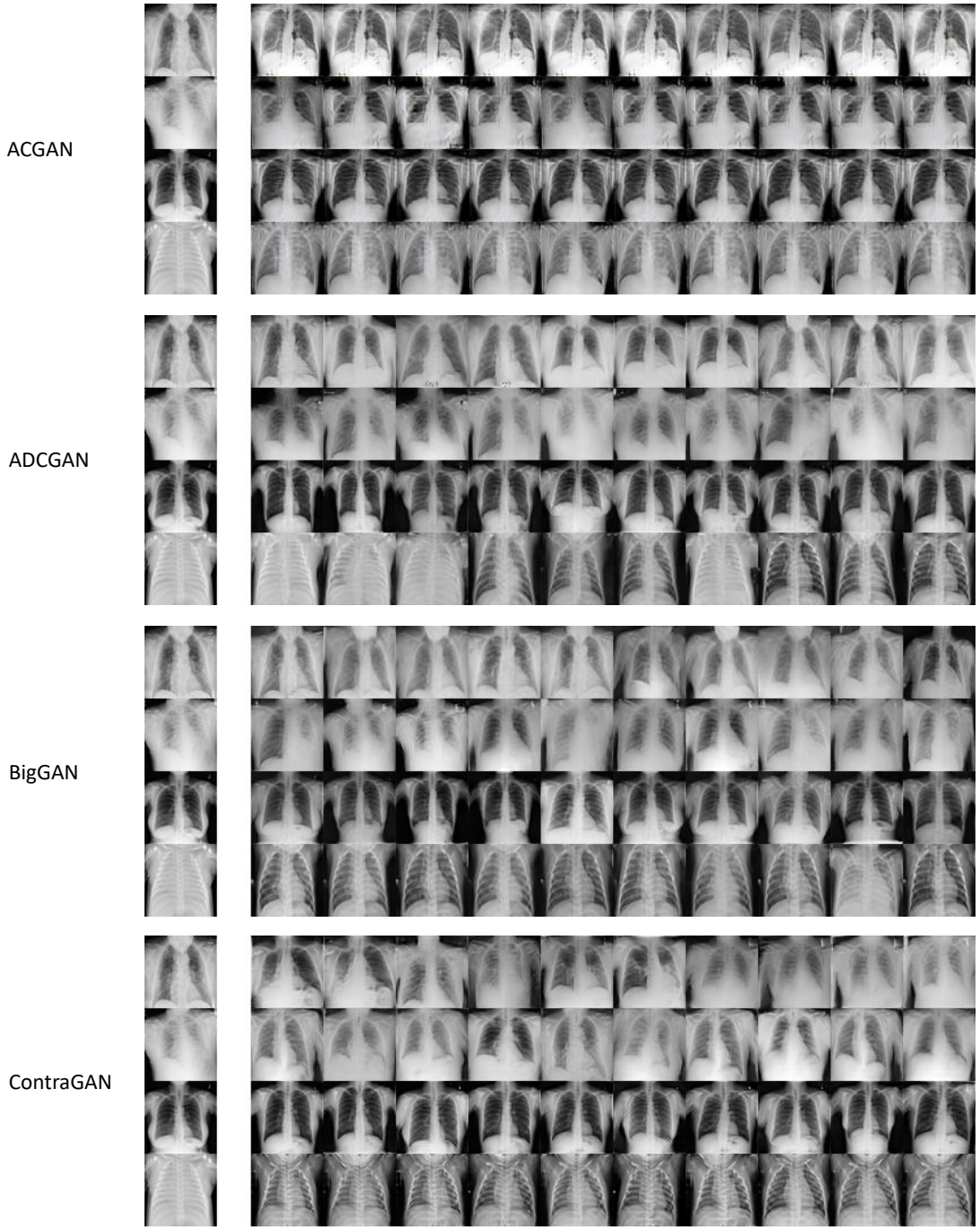


Figure B.3: KNN analysis of ACGAN, ADCGAN, BigGAN, and ContraGAN, with features extracted from the InceptionV3 classifier pretrained on Imagenet.

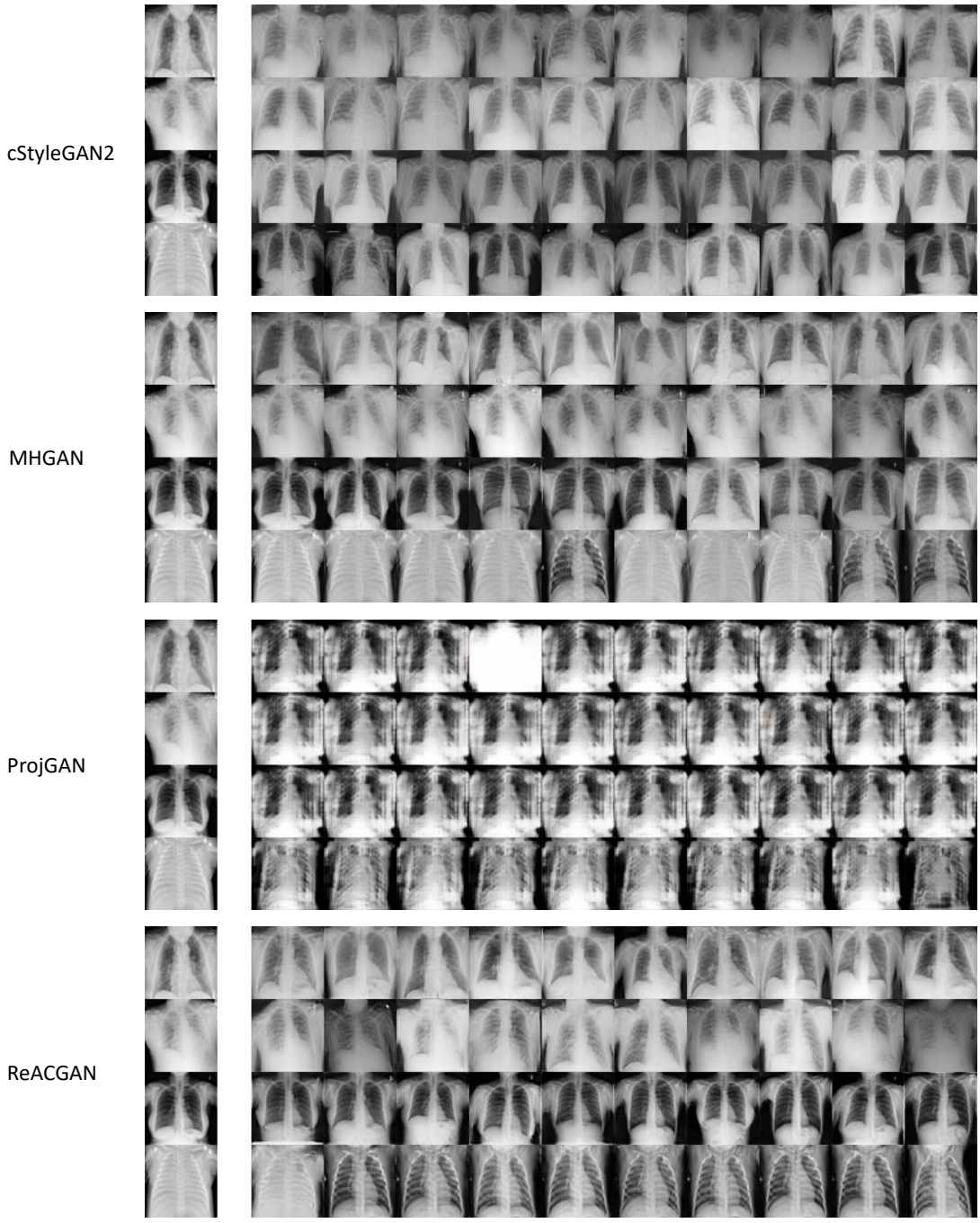


Figure B.4: KNN analysis of cStyleGAN2, MHGAN, ProjGAN, and ReACGAN, with features extracted from the InceptionV3 classifier pretrained on Imagenet.

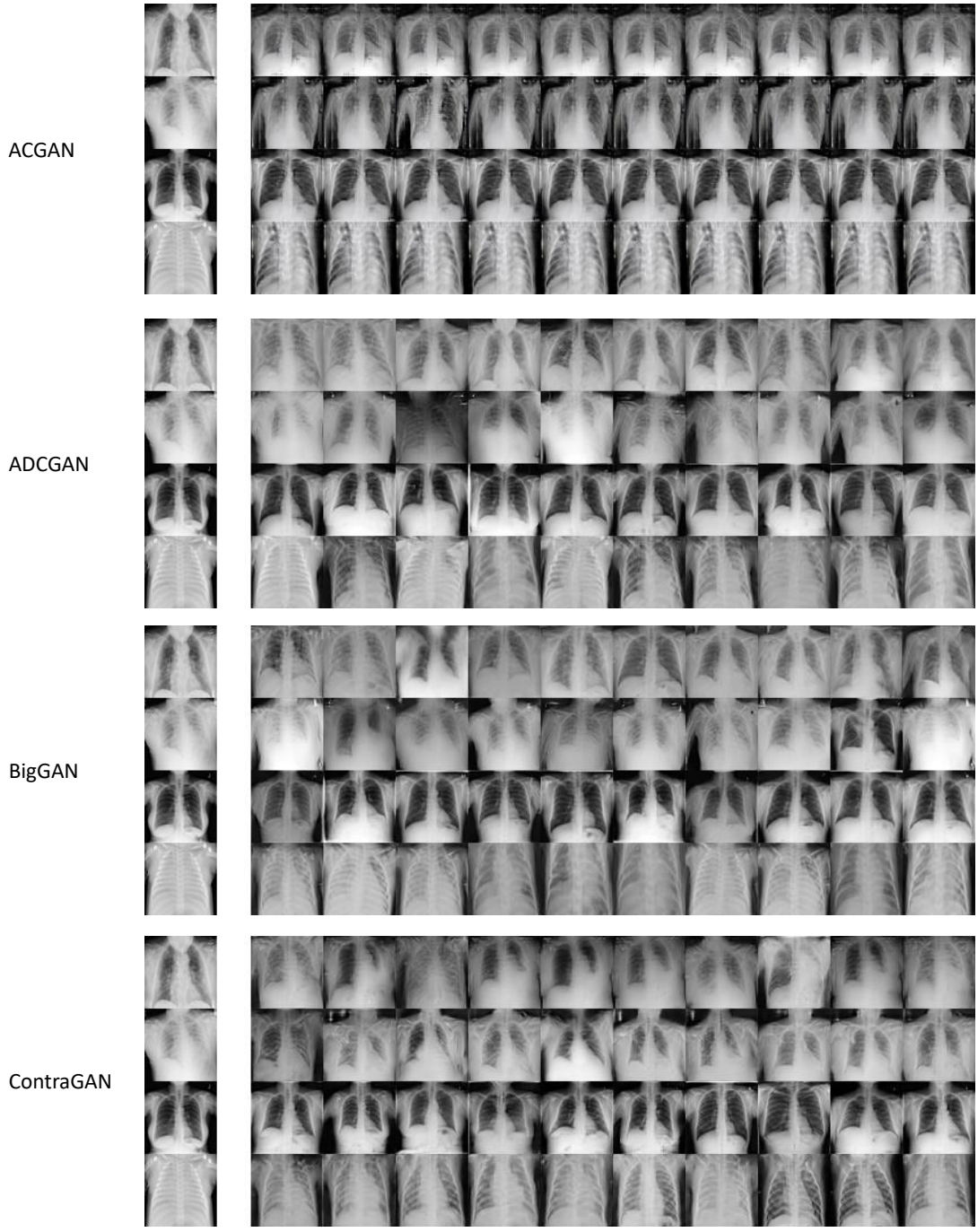


Figure B.5: KNN analysis of ACGAN, ADCGAN, BigGAN, and ContraGAN, with features extracted from the InceptionNet classifier trained on X-rays.

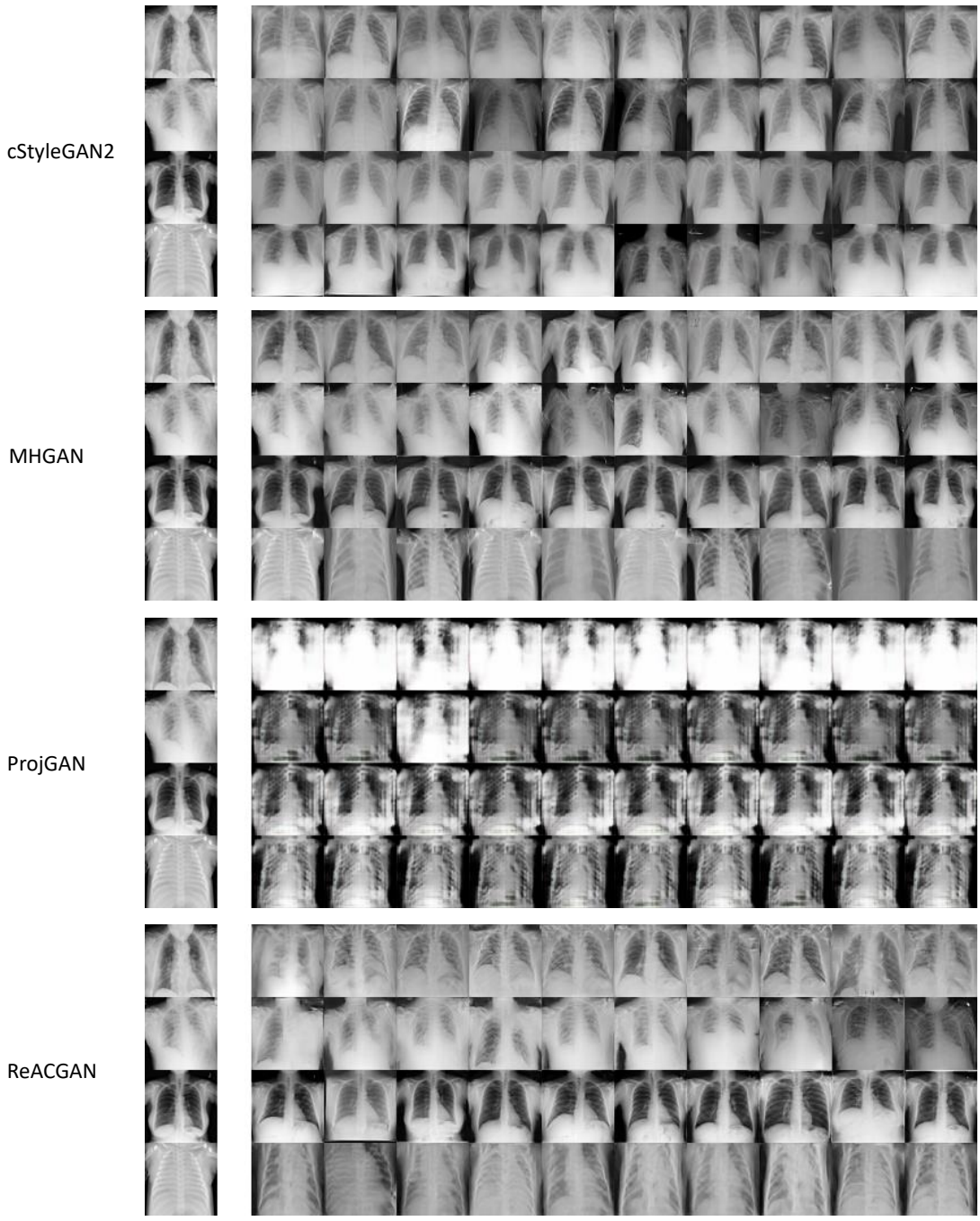


Figure B.6: KNN analysis of cStyleGAN2, MHGAN, ProjGAN, and ReACGAN, with features extracted from the InceptionNet classifier trained on X-rays.