In: 2022 7th International Conference on Intelligent Informatics and Biomedical Science (ICIIBMS), 24-26 November 2022, Nara, Japan
DOI: 10.1109/ICIIBMS55689.2022.9971542

An Empirical Evaluation of Generative Adversarial Nets in Synthesizing X-ray Chest Images

Zakariae Belmekki, Jun Li, Karl Jenkins
Center for Computational Engineering Sciences
Cranfield University, School of Aerospace
Bedford, United Kingdom
{zakariae.belmekki} {jun.li} {k.w.jenkins}@cranfield.ac.uk

Patrick Reuter, David Antonio Gómez Jáuregui
University of Bordeaux, ESTIA Institute of Technology,
INRIA
Bidart, France
preuter@labri.fr, d.gomez@net.estia.fr

***Abstract*—In the last decade, Generative Adversarial Nets (GAN) have become a subject of growing interest in multiple research fields. In this paper, we focus on applications in the medical field by attempting to generate realistic X-ray chest images. A heuristic approach is adopted to perform an extensive evaluation of different architecture configurations and optimization algorithms and we propose an optimal configuration of the baseline Deep Convolutional GAN (DCGAN) based on empirical findings. Additionally, we highlight the technical limitations of GAN and provide an analysis of the high memory requirements, which we reduce by a range of 1.2-7 percent by removing unnecessary layers. Finally, we verify that the loss of the discriminator can be used as an assessment metric.**

Keywords—Generative Adversarial Networks; Medical Imaging; X-ray Chest Images

I. Introduction

Deep Learning (DL) has proven to be a powerful class of tools in the last decade. It is divided into two categories: discriminative models and generative models. Among the latter are Generative Adversarial Networks (GAN) [1], which emerged as a powerful generative framework in 2014. Since then, a large number of works presenting impressive improvements to the original framework have been published: state-of-the-art GAN models could successfully generate realistic images [2], [3].

GAN's compelling synthetic capabilities encourage a potential use in different applications. In this work, we focus on medical imaging by generating X-ray chest images. This work is motivated by the following issues that face DL applications in medical imaging:

1) Medical privacy: training a reliable model requires a huge amount of data, which is often not met by models to be safe and ready to use in real-life clinical situations [4], [5]. A major factor that contributes to this lack of data is medical privacy: hospitals are not able to share their local datasets without the explicit ongoing consent of their patients [6]. Furthermore, in a DL project that involves medical imaging, data collection requires approval by institutional review boards to ensure patients' ongoing consent and information, and assess the risks and benefits of the project [6]. Also, patients can withdraw their consent, which leads to the destruction of their data [6]. Therefore, alleviating the medical privacy problem would make available a significant amount of data [7] and potentially make possible a large public clinical database. The latter, not only would it contribute to the reliability of DL models, but would also accelerate clinical DL application pipelines.

2) Models' lack of validation: this issue is directly linked to the previous one, as lack of validation can be mitigated by access to more data. Despite the plethora of papers presenting novel DL models as state-of-the-art exceeding their predecessors in performance and accuracy, there should be a certain degree of skepticism regarding researchers' optimism. The reason being DL studies tend to be retrospective and locally validated [8]. It is undeniable that impressive progress has been made; however, the lack of validation of DL models remains a major problem. Employment of such models in real-life clinical scenarios would not be desirable considering the high risks involved. Most models are single-purpose and were tested and validated using limited and single-institution datasets [5], [8].

3) High cost of data annotation: The tasks of clinical data collection and labeling can prove to be heavy on radiologists [8]. These tasks require human workforce and are not automated. Training reliable DL models requires thousands of images, hence the heavy nature of data annotation. Additionally, it is safe to assume that annotation by expert radiologists requires a significant amount of financial resources and time [5]. Reducing the time and cost of this process would help accelerate the DL solutions development workflow.

In this context, the present work contributes to the line of research aiming to use GAN to alleviate the above-mentioned problems. GAN has the potential of overcoming privacy issues by generating synthetic samples containing features from real datasets that otherwise would be publicly inaccessible. Thus making more data accessible and improving existing and future models' validation process. Additionally, GAN can also function in a semi-supervised way [9], which makes it usable even with small annotated datasets. Our objective is to use GAN in an X-ray imaging scenario: we build a model for generating realistic chest X-ray images. Additionally, we provide an empirical analysis of three aspects of GAN: (1) different

© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

implementation configurations, (2) memory requirements, and (3) GAN assessment metrics. The paper is organized as follows: we start with a general overview of the GAN framework, providing some background and limitations of GAN. Afterward, the model design is described in terms of architectural constraints. We then present a memory requirements analysis and explore the possibility of compressing the conventional DCGAN convolution blocks. Finally, an empirical comparison between different configurations is presented.

II. Methods

A. Generative Adversarial Nets

GAN consists of two models: a generator G and a discriminator D. Both models engage in an adversarial minimax game against one another wherein their weights are updated simultaneously. G implicitly attempts to learn the input data distribution p_{data} by mapping a prior latent noise p_z to data space. G, fed with noise $z \sim p_z$, is updated by feedback from D, which is a classifier trained to distinguish between real images x from p_{data} and synthetic samples G(z) from p_G. Theoretically, G is trained to minimize the probability that D correctly classifies samples i.e. $log\ (1 - D(G(z)))$, while D is trained to maximize the probability of assigning the right classes to its input, i.e. $log\big(D(x)\big) + log\Big(1 - D\big(G(z)\big)\Big)$. The objective function is defined

$$\min_G \max_D E_{x\sim p_{data}(x)}\ [log\ D\ (x)] + E_{z\sim p_z(z)}\Big[log\Big(1 - D\big(G(z)\big)\Big)\Big]. \tag{1}$$

Since GANs emerged, they became of central interest in the generative models' field. A plethora of GAN-based works have been published and impressive progress has been made. Many GAN variants exhibit high synthetic capabilities and have succeeded in outperforming their predecessors (e.g. [10], [2]). However, multiple challenges remain: (1) mode collapse, (2) high memory requirements, and (3) lack of training stability. Mode collapse refers to the scenario where G maps the latent space to the same small number of outputs, thus the samples would all look the same.

B. Fréchet Inception Distance

The original GAN paper [1] assessed sample quality based on Parzen window-based log-likelihood estimates. Further work by Salimans et al. [9] proposed the Inception Score, a more reliable metric to assess sample quality and diversity. The present work makes use of the Fréchet Inception Distance ́ (FID), proposed by Heusel et al. [11], which, to the best of our knowledge, is the most reliable quantitative assessment metric for GANs. The method computes the Fréchet Distance ́ between the Gaussian with mean (m, C) from the original data distribution p_{data} and the Gaussian with mean (m_w, C_w) from p_G. It is defined as follows

$$d^2\big((m, C)\big) = \|m - m_w\|_2^2 + Tr\Big(C + C_w - 2(CC_w)^{\frac{1}{2}}\Big). \tag{2}$$

FID computes the distance between feature vectors of both real and synthetic samples. The computed score represents the similarity of two sets of samples based on the Inception V3 model [12]. The lower the score is, the more similar the images are and the better the quality.

C. Model Design

The models presented in this work are all built using fully convolutional layers, similarly to DCGAN (Radford et al. [13]), which is also used as a baseline model. While DCGAN uses LeakyReLU and ReLU nonlinearities, we implement our models using Exponential Linear Units (ELU) nonlinearity, proposed by Clevert et al. [14]. It is defined by

$$f(x) = \begin{cases} x & if\ x > 0 \\ \alpha(exp(x) - 1) & if\ x \leq 0, \end{cases} \tag{3}$$

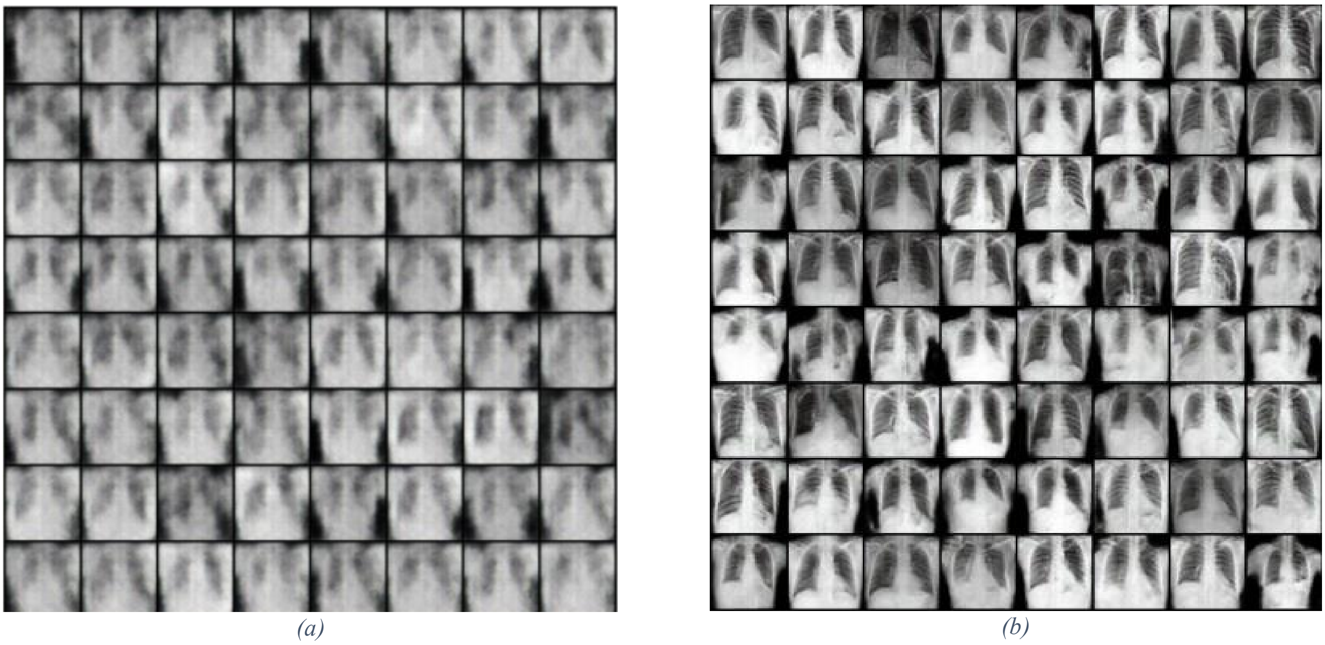
with α > 0. The choice of ELU is justified by its superior performance reported in [14] and its mitigating effect on vanishing gradients, which happens to be a major issue in GANs. It exhibited superior precision and learning speed in fewer epochs of training on the ImageNet dataset [14]. Additionally, ELU fulfills the first convergence assumption for the next implementation choice: the Two Time-Scale Update Rule (TTUR) proposed by Heusel et al. [11], which requires networks with smooth nonlinearities. TTUR consists in choosing two different learning rates for G and D in a way that allows D to train enough while G does not cause high perturbations. Also, Heusel et al. [11] provide proof of convergence to a local Nash equilibrium under some assumptions that can be ensured technically.

For optimization, the novel Adaptive Step-Size (AdaS) optimizer [15] was used. AdaS relies on fine-tuning the step-size s: given a differentiable convex function $F(x)$, minimizing $F\big(x - s\nabla F(x)\big)$ leads to finding the optimal value for S, leading to the following equation [15]

$$s_{n+1} = s_n + \nabla F(x) \times \nabla F(x + \nabla F(x) \times s_n). \tag{4}$$

The AdaS algorithm replaces the gradient of $F(x)$ with an exponential moving average of the derivative of x and $\nabla F(x + \nabla F(x) \times s_n)$ with the derivative of x [15]. In the author's repository [15], it was reported that, on some datasets, AdaS outperformed AdaM (Kingma & Ba [16]), one of the widely used optimization algorithms. On the MNIST dataset [17], training a shallow network using AdaS led to impressive results compared to AdaM; training was more stable and faster.

Although adaptive optimizers like AdaM are widely used in the DL community, they still suffer from a major problem. As compelling as the training outcomes may be, these adaptive algorithms suffer from poor generalization compared to the standard Stochastic Gradient Descent [18]. On this front, AdaS has an advantage over AdaM when it comes to generalization. Although experiments with this optimizer were not extensive, the official repository provides a comparison between both



(a) *(b)*

Fig. 1. 32×32 samples generated using models trained using AdaM (a) and AdaS (b). (a) are blurry and lack definition while (b) show an impressive level of detail. Both models were trained for the same number of epochs.

optimizers on the CIFAR-100 dataset [19]: the results showed that AdaS surpasses AdaM in generalization [15].

III. Experiments and Results

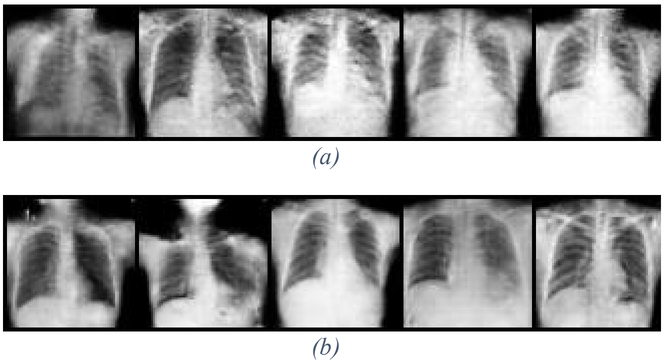
In this work, unless stated otherwise, the models are built to generate 64×64 images. The dataset used is a portion of the VinBigData Chest X-ray Abnormalities Detection dataset [21]. For model assessment, both qualitative and quantitative metrics were used. The latter consist of mere human eye observation of texture details, blur, and apparent similarity to real X-ray chest images. Quantitative measures are more emphasized and include FID scores and the mean loss value of D, which we explore and analyze to conclude whether it can be used as a GAN performance metric.

A. Optimizers Comparison: AdaM vs AdaS

Preliminary results showed that AdaS outperformed AdaM when both are used to train the same model. Using AdaS, the model could generate convincing images in the early stages of training. This supports the claim of the author [15] about the superior performance of AdaS. Also, it converged faster than AdaM in our settings. Fig. 1 shows preliminary samples generated after training two models for a few epochs.

Generating 64×64 samples using the architecture built with ELU non-linearity presented in Table II further confirmed the preliminary results. AdaS exhibited faster convergence characteristics than AdaM. Fig. 2 shows the results after training both models for 150 epochs. The difference in quality is apparent to the human eye.

Fig. 3 shows both models' discriminator's loss during training. For the loss function that we implemented for our GAN models, which is the vanilla one, the higher its value is, the better G captures p_{data}.



(a)

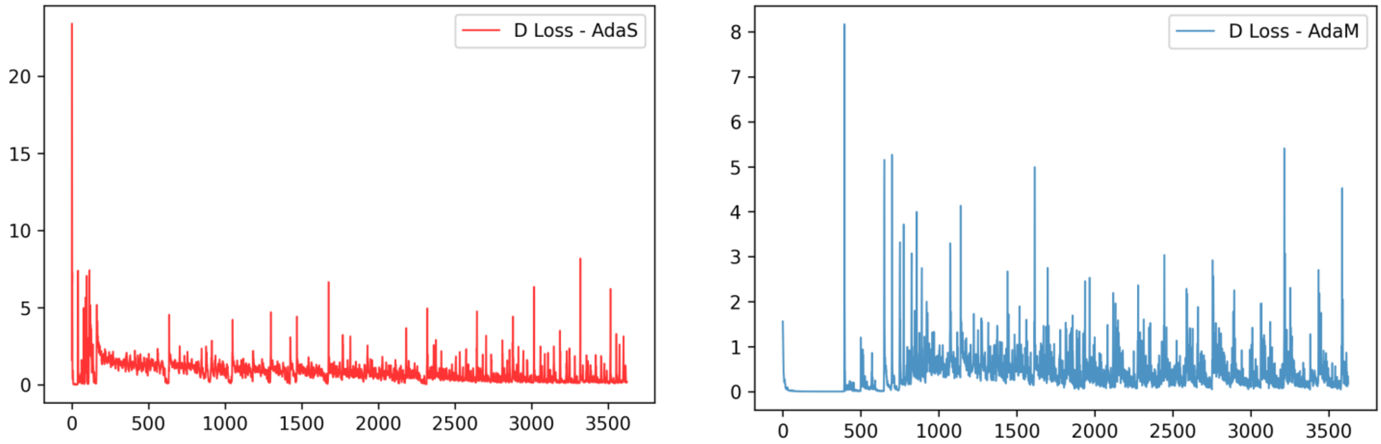
(b)

Fig. 2. 64×64 samples generated by models trained using AdaM (a) and AdaS (b). (b) exhibits more realistic features than (a), which indicates faster training as the model captured more features. Both models were trained for 150 epochs.

We observed that the loss of D is low early in training. The period is denoted as T_0. It is due to vanishing gradients, already documented in previous GAN literature [1], [20]. As stated in the first GAN paper [1], in the initial stages of training, D distinguishes easily between generated and real samples as G is still not trained enough. The longer it is, the slower G starts learning p_{data}.from samples x. For the model trained using AdaS, T_0 is reduced by half compared to the one trained using AdaM: for the former, it ends before 200 iterations, while for the latter it ends around 400 iterations as shown in Fig. 3. This gives the AdaS model a head start, which explains the higher quality samples in fewer epochs.

TABLE I. STATISTICAL ANALYSIS OF THE LOSS OF D AND FID SCORES FOR THE FOUR CONFIGURATIONS. THREE ASPECTS WERE VARIED: OPTIMIZATION ALGORITHM, NON-LINEARITY, AND TTUR. "P" REFERS TO THE PRESENCE OF TTUR CONSTRAINTS AND "-" ITS ABSENCE. THE CONFIGURATION WITH THE HIGHEST LOSS MEAN VALUE IS THE ONE WITH THE HIGHEST FID SCORE, WHICH IS THE ONE IMPLEMENTED WITH ELU AND TTUR AND TRAINED USING ADAS.

Optimizer	Activation	TTUR	Mean	Variance	Standard Deviation	FID
AdaS	ELU	-	0.845	0.637	0.798	104.552
AdaM	ELU	-	0.418	0.209	0.458	238.254
AdaS	ELU	P	**1.185**	4.066	2.016	**100.581**
AdaS	ReLU, LReLU	-	1.135	0.970	0.985	111.882



(a)

(b)

Fig. 3. The evolution of the loss of D during training for (a) the model trained using AdaS (b) the model trained with AdaM. The initial phase of (b) before 480 iterations is characterized by very low loss, during which G does not learn.

TABLE II. THE TOP-PERFORMING CONFIGURATION. BOTH THE GENERATOR AND DISCRIMINATOR WERE BUILT WITH ELU NON-LINEARITY. BATCH NORMALIZATION LAYERS ARE ONLY INTEGRATED INTO THE HIDDEN LAYERS OF THE DISCRIMINATOR.

Generator	Discriminator

ConvTranspose2d(100, 512, 4)
ELU()
	Conv2D(1, 64, 4)
	ELU()

ConvTranspose2d(512, 256, 4)
ELU()
	Conv2D(64, 128, 4)
	BatchNorm2d(128)
	ELU()

ConvTranspose2d(256, 128, 4)
ELU()
	Conv2D(128, 256, 4)
	BatchNorm2d(256)
	ELU()

ConvTranspose2d(128, 64, 4)
ELU()
	Conv2D(256, 512, 4)
	BatchNorm2d(512)
	ELU()

ConvTranspose2d(64, 3, 4)
Tanh()
	Conv2D(512, 1, 4)
	Sigmoid()

Although initial tests have shown AdaS to be more stable during training, further experiments showed the opposite. The variance and standard deviation of the discriminator's loss curve for AdaS were higher than AdaM's (see Table 1). This led us to dismiss our initial observation about the stability of AdaS. Additionally, models trained using AdaM were more challenging to fine-tune. Some models could not produce any decent samples until AdaS was used. We tested multiple architectures but most failed. Most of them did not achieve any noteworthy results. We varied the learning rate mostly when using AdaM. By setting the learning rate to 0.0001, we observed that the training was stable and we could not see any effect of vanishing gradients in the discriminator. However, the model was too slow to converge and capture p_{data}. Even after 100 epochs, the model failed to produce any distinguishable features.

Following that, attempts to accelerate the convergence of the model were made by increasing the learning rate tenfold. Setting it to 0.001 caused high instability and the effect of vanishing gradients was apparent. In this case, D was too good and did not provide enough information for the generator to train on. Additionally, we observe that increasing the batch size increases T_0 when using AdaM. The model built with ELU could not make any progress in the first 10 epochs for a batch size of 256. However, setting it to 128 reduced T_0 and the model could generate decent samples.

B. *Impact of TTUR on Performance*

In this section, we explore the use of a rough implementation of TTUR. We adopt two different learning rates for G and D in a way to minimize the perturbations of the former on the latter. However, we did not satisfy all the assumptions for convergence in [11]. Fig. 4 shows the loss of D during training with and without TTUR using AdaS. Not only T_0 was minimized, but also the loss of D increased in the initial phases. The stability of the model was also increased after T_0. We calculated the mean, variance, and standard deviation for this new implementation and observed a higher mean value of the loss of D (see Table 1). The variance and standard deviation are higher for the model implemented with TTUR because of the initial phase that has high variance; however, it becomes more stable after approximately 200 iterations. The reason behind the high values of the variance and standard deviation compared to the other two models is the significantly high loss values in T_0. Statistically, we can see that once the training stabilizes after the initial phase, the model that implements TTUR appears to be more stable.

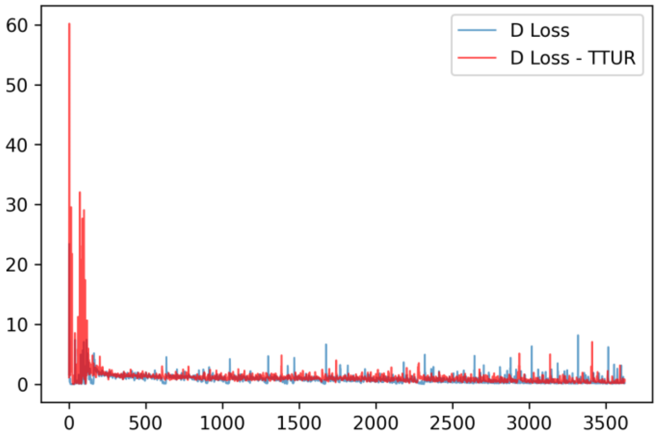


Fig. 4. Loss curves of two models trained using AdaS, the red one is that of the model implemented with TTUR while the blue one is that of the baseline model. TTUR increases the loss of D in the initial stages of training, indicating that G learns more.

C. Non-linearity Comparison: ELU vs LeakyReLU and ReLU

All models trained in the previous sections used ELU. In this section, the performance of the conventional DCGAN nonlinearities, i.e., ReLU and LeakyReLU, is compared to that of ELU. Batch Normalization (BN) was only added to the discriminator of the model with ELU. The statistical analysis of the loss of D showed that the conventional DCGAN using AdaS outperformed the ELU model without TTUR. Stability was improved and T_0 decreased. This is confirmed by the statistical analysis. Results in Table 1 show that AdaS, LeakyReLU, and ReLU non-linearities provide more training to G since the mean value of the discriminator's loss is higher than that of ELU. The latter, on the other hand, provides more stable training as the variance and standard deviation are lower. However, coupling ELU with TTUR bore superior results when trained with AdaS.

D. Sample Quality Assessment

In order to confirm the findings of the previous section, the Fréchet Inception Scores were computed. The results are ′ presented in Table 1. It was further confirmed that the best model was the one built with ELU, TTUR, and trained using AdaS. Its score was nearly twice as better as the one trained using AdaM.

To confirm that our statistical analysis based on the mean value of D is a valid metric to assess GAN, we calculated the Pearson correlation coefficient for all the statistical results. We find that FID and the mean loss value of D are highly inversely correlated. This indicates that the higher the mean loss value of D is, the higher the sample quality. Fig. 5 shows the correlation matrix.

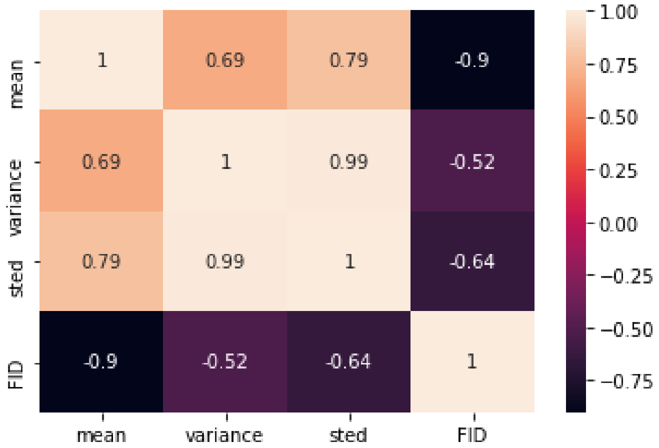


Fig. 5. Pearson correlation matrix between FID, the mean loss value of D, its variance, and its standard deviation. Both FID and the mean loss value are highly inversely correlated.

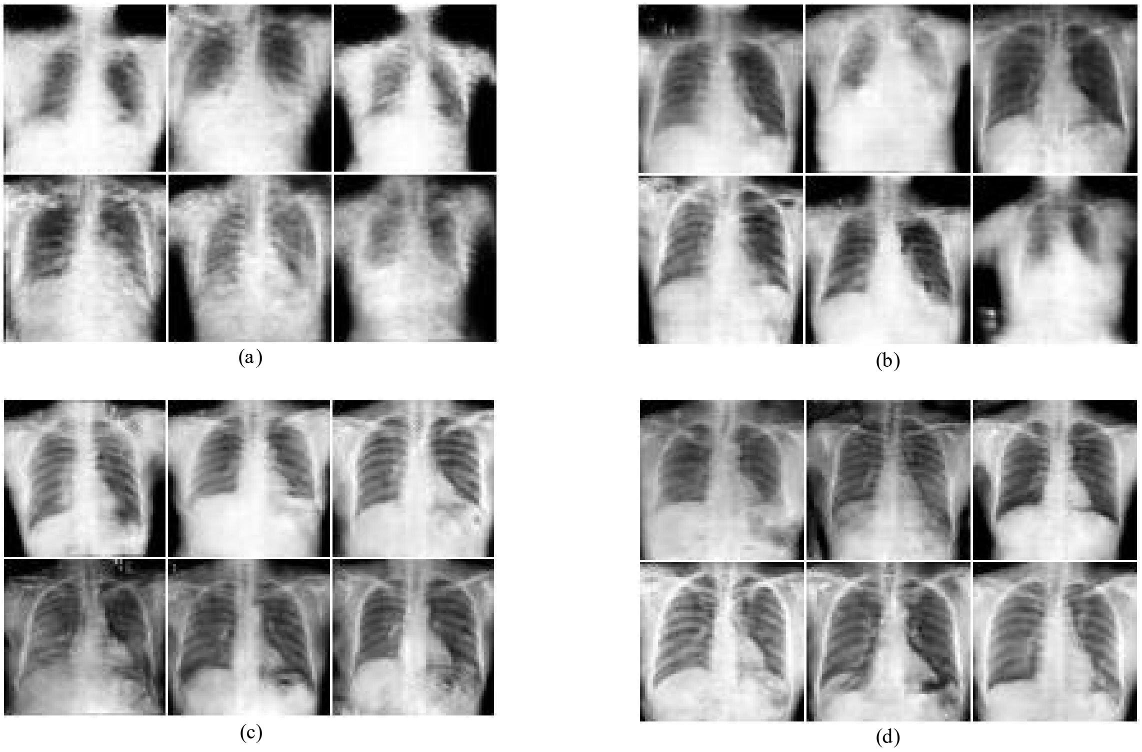
Finally, we present samples from all the four trained models in Fig. 6.

E. Memory Requirements Analysis

DCGAN can be easily trained for low resolutions and achieve impressive results without requiring much memory. Preliminary experiments conducted on 32 × 32 images could run on a GPU with less than 4 Gigabytes of memory; however, once the model had been scaled up for higher resolutions, the GPU was no longer usable. BigGan (Brock et al. [3]), which is a state-of-the-art scaled-up GAN variant, benefited from a large increase of certain hyper-parameters, one of which is the feature map size of convolution layers. We varied the feature map size from 32 to 320 in every convolutional block of the network and reported the memory requirements in the graph in Fig. 7. Our experiments showed that even with low-resolution images of 64 × 64 and 128 × 128, memory requirements are exponentially proportional to feature maps size.

In this analysis, memory requirements for processing one image are provided using the in-built Pytorch function summary(). The total batch memory requirements can be obtained by multiplying the results for one image by the size of the batch. e.g., a 64 batch of 64 × 64 images with a feature map size of 64 would require (53.96 + 46.72) × 64 = 6443, 52 (MB). That is nearly 6.5 GB for low-resolution images, which are of no significant use in a real-life imaging application.

In order to reduce the memory requirements of GAN, a first attempt was to remove BN layers. Fig. 8 is a plot that compares a generator model with and without BN. Removing BN reduces memory consumption by 1.2-7% depending on the feature map size. The higher it is, the lesser the benefit of removing BN. This could be especially useful for scaled-up models, where a trade-off between BN layers and the batch size could be achieved, which is beneficial for GAN [3].


(a)

(b)

(c)

(d)

Fig. 6. Samples generated by the four configurations: (a) ELU + TTUR trained using AdaM, (b) ELU trained using AdaS, (c) ELU + TTUR trained using AdaS, (d) DCGAN (LeakyReLU + ReLU) trained using AdaS.

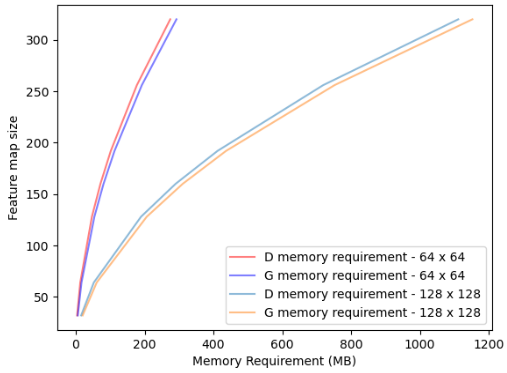


Fig. 7. Evolution of memory requirements of both G and D in terms of feature map size for 64 × 64 and 128 × 128 images.

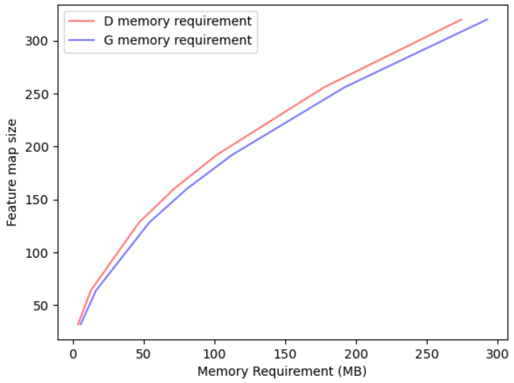


Fig. 8. Memory requirements for processing one image with and without Batch Normalization in the model.

F. *Model Compression*

The conventional DCGAN is built using convolutions with strides for each twofold resolution up-sampling. Inversely, deconvolutions use strides of two for each twofold resolution down-sampling. Following the memory issues, attempts to compress the model were made aiming to find a good balance between performance and memory consumption. All tests failed to produce any successful results. A first attempt was to decrease the depth of the network. The kernel size of the convolution layers had to be adapted according to the desired output dimension. The values were changed according to equation 5 that determines the dimension of the output of a convolution layer

$$Dim_{out} = \frac{Dim_{in}+2\times P-D\times(K-1)-1}{S}+1, \tag{5}$$

with Dim_{out} the output dimension (can be either height or width), Dim_{in} the input dimension. We denote P, D, K, and S respectively the padding, dilation, kernel size, and stride value. Conventional DCGAN models use a stride value of 2 for up-sampling and down-sampling in both convolutions and transpose convolutions. We attempted to increase it in order to have fewer layers. Extensive model building and parameter tuning did not result in any positive results. We consistently obtained unrecognizable features. The major problem with compressing GAN models is the blurry squares with homogeneous colors, presented in Fig. 9. The deeper the network is, the less the phenomenon is observed.

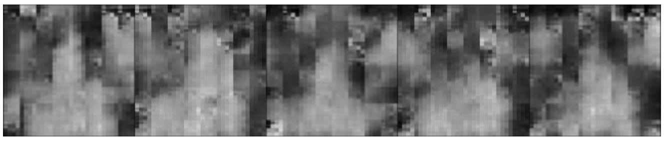


Fig. 9. Blurry squares produced by the compressed GAN.

Attempts to mitigate this problem were made by adding fully-connected layers to the input of D and the output of G. Previous work suggests that combining convolutional and fully-connected layers improves performance (Barua et al. [20]), even though it is not ideal for a scaled-up model considering the scope of our work, which is to use the least amount of memory possible. Results showed that it had no effect on the observed phenomenon. Therefore, a heuristic approach was adopted to identify the source of the problem: configurations and hyper-parameters were varied. Table III summarizes the experiments and results. None of the changes addressed the issue. Increasing the depth of the models was the only solution. Therefore, it was concluded that, for the choice of configurations in this work, a functional compressed GAN was out of reach. Despite failed extensive experimenting, the possibility of a functional compressed GAN is not categorically dismissed. Instead, the observations provide a solid empirical argument for the highly volatile aspect of GAN.

TABLE III. EXPERIMENTS TO IDENTIFY THE SOURCE OF THE BLURRY SQUARES ON IMAGES. "V" STANDS FOR "VARIED" AND "F" STANDS FOR "FAILED". WE VARY ALL THE PARAMETERS IN THE TABLE AND OBSERVE THE EFFECT ON THE PHENOMENON OBSERVED.

	Stride Size			**FC Layers Presence**			**Latent Space Dimension**			**Feature Map Size**		
D	V	-	V	V	-	V	V	-	V	V	-	V
G	-	V	V	-	V	V	-	V	V	-	V	V
Results	F	F	F	F	F	F	F	F	F	F	F	F

G. *Generating High-Resolution Samples*

Prior to implementing models for generating 64 × 64 X-ray chest images, attempts to generate 128×128 images were made. Despite extensive fine-tuning, all attempts to generate any recognizable features failed, except for the configuration presented in Table II that could generate some high-level features. Results after 400 epochs are presented in Fig. 12, which took more than a day to train on Google Colaboratory using an NVIDIA Tesla P100-PCIE-16GB GPU.

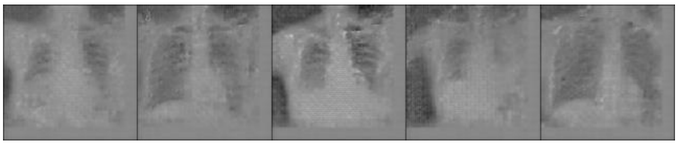


Fig. 11. 128 × 128 samples generated by preliminary a high-resolution model.

After 400 epochs, training had to be stopped due to the occurrence of mode collapse: the model kept producing the same samples for a long range of epochs. In another experiment, it was observed that this phenomenon was also reflected in the loss curve, marked by a sudden exponential increase in both G and D losses (see Fig. 11). Using AdaM in this configuration did not produce any noteworthy samples. DCGAN, in its conventional architecture, is hard to scale up unless the hyper-parameters and networks depth are largely increased. Our effort to reduce the memory costs was merely heuristic, we recommend taking a more theoretical and low-level approach in contrast to ours.

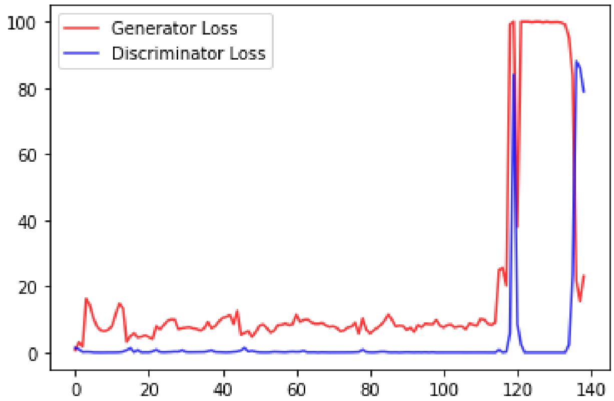


Fig. 11. G and D loss curves during mode collapse. It is marked by a sudden exponential variation in the loss.

IV. Discussions

A. *GAN in Medical Imaging*

GANs have already been used in Medical Imaging in other works. Most relevant to our work is [7], where the authors propose GAN for data augmentation, which is related to the issues we raised in Section I. In contrast to their work, this paper focused on the performance of the GAN model itself, not the consequences it would have on involved subsequent processes (e.g., image segmentation). In the same context of medical imaging, [22] used GAN for 3D images and proposed an algorithmic solution to reduce the significant memory requirements, by operating on a patch-based manner. In our work, we only reduce the number of layers, which we deem a good start before attempting more sophisticated potential solutions.

Despite the apparently realistic samples, a recent study suggested that synthetic samples are not as rich in detail as the real ones [23]. It is plausible that it is due to the recent claim that GANs are biased against high frequencies in the spectral domain [24], meaning low-level details would be missing. For future work, we recommend exploring the bias against high frequencies: alleviating this problem would make GAN-generated synthetic samples more usable in real-life scenarios.

B. *GAN Memory Optimization*

Prior works have attempted to reduce the memory requirements of GAN; however, the amount of works published on the matter is significantly low. The reviewed literature adopted different approaches (e.g., [25] used ternary weights thus achieving compression, [26] altered the GAN framework by removing the generator), all achieving compelling results. In this work, by only removing BN layers, we reduced memory requirements by a range of 1.2-7%.

V. Conclusion

In this paper, we evaluated different configurations and succeeded in improving the baseline DCGAN traditional implementation by experimenting with different configurations. The top-performing configuration, presented in Table II, successfully generated realistic X-ray chest images. Additionally, our results show that the loss value of D can be used as a metric for assessing GAN. The aspect of high memory requirements was also explored; we presented an analysis and adopted a heuristic approach to reduce them by a range of 1.2 − 7%. While GANs exhibit high synthetic capabilities, there is still a significant amount of progress required to make their applications reliable and safe in real-life clinical scenarios.

References

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., Generative adversarial nets, 2014. arXiv: 1406.2661 [stat.ML].

[2] T. Karras, S. Laine, and T. Aila, A style-based generator architecture for generative adversarial networks, 2018. DOI: 10.48550 / ARXIV.1812.04948. [Online]. Available: https://arxiv.org/abs/1812.04948.

[3] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," CoRR, vol. abs/1809.11096, 2018. arXiv: 1809.11096. [Online]. Available: http://arxiv.org/abs/1809.11096.

[4] W. Ou, D. Polat, and B. Dogan, "Deep learning in breast radiology: Current progress and future directions," English (US), European Radiology, vol. 31, no. 7, pp. 4872–4885, Jul. 2021, Publisher Copyright: © 2021, European Society of Radiology., ISSN: 0938- 7994. DOI: 10.1007/s00330-020-07640-9.

[5] P. Chea and J. C. Mandell, "Current applications and future directions of deep learning in musculoskeletal radiology," Skeletal Radiology, vol. 49, no. 2, pp. 183– 197, Aug. 2019. DOI: 10.1007/s00256- 019- 03284- z. [Online]. Available: https://doi.org/10.1007/s00256-019-03284-z.

[6] E. Montagnon, M. Cerny, A. Cadrin-Chenevert, ˆ et al., "Deep learning workflow in radiology: A primer," Insights into Imaging, vol. 11, no. 1, Feb. 2020. DOI: 10.1186/s13244-019-0832-5. [Online]. Available: https://doi.org/10.1186/s13244-019-0832-5.

[7] H.-C. Shin, N. A. Tenenholtz, J. K. Rogers, et al., Medical image synthesis for data augmentation and anonymization using generative adversarial networks, 2018. DOI: 10.48550/ARXIV.1807. 10225. [Online]. Available: https://arxiv.org/abs/1807.10225.

[8] W. C. Ou, D. Polat, and B. E. Dogan, "Deep learning in breast radiology: Current progress and future directions," European Radiology, vol. 31, no. 7, pp. 4872– 4885, Jan. 2021. DOI: 10.1007/s00330-020-07640-9. [Online]. Available: https://doi.org/10.1007/s00330-020-07640-9.

[9] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," CoRR, vol. abs/1606.03498, 2016. arXiv: 1606.03498. [Online]. Available: http://arxiv.org/abs/ 1606.03498.

[10] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," CoRR, vol. abs/1606.03657, 2016. arXiv: 1606.03657. [Online]. Available: http://arxiv.org/ abs/1606.03657.

[11] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium," CoRR, vol. abs/1706.08500, 2017. arXiv: 1706.08500. [Online]. Available: http://arxiv.org/abs/1706.08500.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, Rethinking the inception architecture for computer vision, 2015. DOI: 10.48550/ARXIV.1512. 00567. [Online]. Available: https://arxiv.org/abs/1512.00567.

[13] A. Radford, L. Metz, and S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2016. arXiv: 1511 . 06434 [cs.LG].

[14] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," arXiv: Learning, 2016.

[15] Y. Eliyahu, 'AdasOptimizer', 2020. [Online]. Available: https://github.com/YanaiEliyahu/AdasOptimizer. [Accessed: 24- Aug- 2022].

[16] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, 2014. DOI: 10.48550/ARXIV.1412.6980. [Online]. Available: https://arxiv.org/abs/1412.6980.

[17] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun. com/exdb/mnist/.

[18] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to SGD," CoRR, vol. abs/1712.07628, 2017. arXiv: 1712 . 07628. [Online]. Available: http://arxiv.org/abs/1712.07628.

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[20] S. Barua, S. M. Erfani, and J. Bailey, "FCC-GAN: A fully connected and convolutional net architecture for gans," CoRR, vol. abs/1905.02417, 2019. arXiv: 1905.02417. [Online]. Available: http://arxiv.org/abs/1905. 02417.

[21] Kaggle.com, 'VinBigData Chest X-ray Abnormalities Detection', 2021. [Online]. Available: https://www.kaggle.com/c/vinbigdata-chest-xray-abnormalities-detection/overview. [Accessed: 24- Aug- 2022]

[22] H. Uzunova, J. Ehrhardt, and H. Handels, "Memory-efficient GAN-based domain translation of high resolution 3d medical images," Computerized Medical Imaging and Graphics, vol. 86, p. 101 801, Dec. 2020. DOI: 10.1016/j.compmedimag.2020.101801. [Online]. Available: https://doi.org/10.1016%5C%2Fj.compmedimag.2020.101801.

[23] Y. Skandarani, P.-M. Jodoin, and A. Lalande, Gans for medical image synthesis: An empirical study, 2021. DOI: 10.48550/ARXIV.2105.05318. [Online]. Available: https://arxiv.org/abs/2105.05318.

[24] M. Khayatkhoei and A. Elgammal, "Spatial frequency bias in convolutional generative adversarial networks," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 7152–7159.

[25] A. S. Rakin, S. Angizi, Z. He, and D. Fan, "Pim-tgan: A processing-in-memory accelerator for ternary generative adversarial networks," in 2018 IEEE 36th International Conference on Computer Design (ICCD), 2018, pp. 266–273. DOI: 10.1109/ICCD.2018.00048.

[26] S. Tuli, S. Tuli, G. Casale, and N. R. Jennings, Generative optimization networks for memory efficient data generation, 2021. DOI: 10.48550/ARXIV.2110.02912. [Online]. Available: https://arxiv.org/abs/2110.02912.

Cranfield University

CERES https://dspace.lib.cranfield.ac.uk

School of Aerospace, Transport and Manufacturing (SATM) Staff publications (SATM)

2022-12-12

An empirical evaluation of generative adversarial nets in synthesizing X-ray chest images

Belmekki, Zakariae

IEEE

Belmekki Z, Li J, Jenkins K, et al., (2022) An empirical evaluation of generative adversarial nets in synthesizing X-ray chest images. In: 2022 7th International Conference on Intelligent Informatics and Biomedical Science (ICIIBMS), 24-26 November 2022 Nara, Japan

https://doi.org/10.1109/ICIIBMS55689.2022.9971542

Downloaded from Cranfield Library Services E-Repository