Device-Friendly Privacy-Preserving Generative Adversarial Networks

Sergio Rivera
Department of Computer Science
University of Cambridge
sr2070@cam.ac.uk

Abstract

In an era where computational efficiency and data privacy are prominent among synthetic data generation techniques, this study introduces an end-to-end solution to tackle such complex issues: Device-Friendly Privacy-Preserving Generative Adversarial Networks (GANs). Merging the innovative techniques of GAN Slimming (GS) for model compression and Differential Privacy (DP) for data confidentiality, this research achieves fully converged GANs that are both lightweight and privacy-aware while retaining relatively accurate generations.

1 Introduction

The origin of Generative Adversarial Networks (GANs) has revolutionised the field of deep learning, offering outstanding abilities in generating realistic synthetic data. From augmenting datasets to creating entirely new ones, GANs have found applications in diverse fields such as computer vision, natural language processing, and beyond. However, the deployment of these architecture models, particularly on resource-constrained devices, remains a challenging task due to their computational and memory demands. Moreover, the privacy of data used to train these models is still an area of major concern.

This project explores a stacking approach to address these challenges: Device-Friendly Privacy-Preserving Generative Adversarial Networks. Our methodology exploits three key components: model compression, differential privacy, and GANs. We leverage GAN Slimming (GS) for model compression, embedding popular compression techniques—such as model distillation, channel pruning, and quantisation. This results in a slimmer generator model suitable for deployment in resource-constrained environments.

Privacy concerns are addressed through the integration of Differential Privacy (DP) in the training process. By quantifying and controlling privacy risks, DP ensures that the synthetic data generated by our model does not compromise individual observations from the training set.

2 Background

2.1 Generative Adversarial Networks

First announced in 2014, Generative Adversarial Networks[9] introduced the possibility of learning entire data distributions in a fully unsupervised manner by employing concepts of adversarial training and generative modelling. Over the years, GANs have proven to be useful in a wide variety of applications, ranging from simulating costly experiments in particle physics[4] to turning horses into zebras by applying image translation techniques[21]. Other interesting works have successfully applied GANs to image synthesis[15] and photo-realistic super-resolution[16].

Foundationally, a GAN is set up as a zero-sum game between a generator G and a discriminator D, where both components are typically constrained as multi-layer perceptrons such that D can be implemented as a traditional binary classifier. The overarching goal of this system is twofold: (1) to train D to the point that it becomes able to determine whether any sample x was either drawn from $p_{\text{data}}(x)$ (i.e. the original data distribution we are interested in learning) or drawn from $p_z(z)$ (i.e. a sample generated by G) where z represents the latent space as input of G, and (2) to train G to the point that its generated samples become indistinguishable from those extracted from the real distribution as determined by a fully converged D.

In essence, both models engage in opposite objectives, where G attempts to "fool" D into wrongly classifying generated samples. At the Nash equilibrium, neither player can further improve its loss as $p_z = p_{\text{data}}$ and, hence, the discriminator becomes unable to differentiate between the two distributions, as experienced when $D(x) = \frac{1}{2}$. In practice, both D and G are optimised at simultaneous gradient steps, and it is not required for D to be optimal at each training step[8].

2.2 GAN Slimming

Once a GAN setup has fully converged, the discriminator model is discarded, and the generator can be prepared for deployment. However, as seen in other deep learning architectures, GANs suffer from explosive parameter counts and high parameter complexities. This increases the need for researching compression techniques that can be applied to the generator model, facilitating its deployment into sought-after[17] resource-constrained devices.

Literature has shown that due to some inherent instabilities in the training of GANs, heuristically stacking individual compression techniques does not result in satisfactory results, sometimes even reaching catastrophic failure. Hence, a solution that could circumvent such issues would necessarily have to emanate from embedding some of the most popular compression techniques right into the training step itself rather than form part of a post-processing procedure.

An approach that fulfils the previous claim is ***GAN Slimming (GS)***[24], which comprises an end-to-end optimisation framework that combines model distillation (a slim student network learning to mimic the outputs of a larger teacher network), channel pruning (removing redundant feature maps without sacrificing performance) and quantisation (reducing the precision to lower bit-width representations) to form much lighter generators.

GANs are formulated as traditional minimax optimisation problems, following the Binary Cross-Entropy loss function:

$$\min_G \max_D L_{\text{GAN}} \text{ where } L_{\text{GAN}} = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z))] \quad (1)$$

where D and G are jointly trained. However, GS makes use of several compression techniques, one of them being model distillation, which necessitates a further loss term to be defined as a distance metric (typically some form of perceptual loss function):

$$L_{\text{dist}} = \mathbb{E}_{z \sim p_z(z)} [d(G(z), G_0(z))] \quad (2)$$

such that we can enforce the small student generator G to mimic the outputs of the original larger generator G_0. In GS, unlike previous distillation methods[2], the architecture of G is jointly inferred together with the distillation process directly through channel pruning and quantisation. Eventually, the final form of the objective function, including all compression methods, is defined as follows:

$$L(W, \gamma, \theta) = L_{\text{GAN}}(W, \gamma, \theta) + \beta L_{\text{dist}}(W, \gamma) + \rho L_{\text{cp}}(\gamma) \quad (3)$$

where γ denotes all trainable scale parameters, W all other trainable weights in G, θ all parameters in D, and $L_{\text{cp}}(\gamma) = ||\gamma||_1$ (i.e. applying the L1 norm) as recommended by [18].

This kind of inference of the slimmed generator architecture via only pruning and quantisation can be seen as a special case of Neural Architecture Search (NAS)[12], where the student architecture is morphable from the teacher's without the need to introduce any new operators not already present in the larger generator model.

2.3 Differential Privacy

GANs are typically employed to generate synthetic samples, either to enrich or create entire datasets, as a way to avoid data scarcity, with some of the areas that have recently started adopting such methods being biology and medicine for the purposes of generating biomedical data [3][6].

While GANs have proven capable of learning such complex distributions, they do not provide any guarantees on what the synthetic output reveals about individual samples drawn from the original distribution. This implies that, in theory, it may be possible to train a generator that creates synthetic data that reveals actual training samples, hence the need to protect our original dataset by applying differential privacy during training.

Differential Privacy (DP)[5] is a mathematical framework that enables the quantification and control of the privacy risks involved in the release of statistical data. Most importantly, it provides guarantees regarding the protection of individual samples, ensuring that the risk to one's privacy does not increase by participating in a database. Under the context of GANs, including DP can limit the risk of any particular sample being inferred from a fully converged generator, especially when malicious actors employ common attacking techniques such as model inversion.

One can easily apply DP to deep learning processes by employing the following principles:

- **Randomised Mechanisms:** Inject randomness into the data, the model's parameters, or the gradients during backpropagation.
- **Epsilon (ϵ) and Delta (δ):** Measure the *privacy guarantee*, where lower ϵ values indicate stronger privacy. Note that δ is the probability of the privacy guarantee being violated.
- **Clipping Gradients:** Clip the gradients' magnitude to limit the impact of specific samples (sensitivity).
- **Privacy Budget:** Gracefully manage the privacy budget as it gets consumed during training (i.e. as the model learns from the training data).

3 Methodology

To achieve the goal of device-friendly, privacy-preserving adversarial models, we begin by training a large generator model G_0 against a traditional binary classifier (acting as the discriminator D) to mimic samples extracted from an arbitrary data distribution (see Section Dataset). Afterwards, we produce a distillation dataset S by performing multiple forward passes to a fully converged G_0, which will act as a teacher model.

For compression, we employ the GAN Slimming framework to train a distilled, channel-pruned, quantised student generator G_s on S; to better enforce the knowledge transfer between the teacher-student models, we include a perceptual loss [14] term as the distance measure. Then, we extract the sub-network as determined by the non-zero (significant) channels after pruning by copying its quantised weights from the distilled student into a new compressed generator architecture G. Finally, we fine-tune G with Differential Privacy using sensible hyperparameters to create a final fully-compressed privacy-preserving model.

To test our setup under sufficient conditions, we train three student networks G_s to evaluate the impact of different quantisation levels and record our quantitative (see Section 4.3) and qualitative (see Section 4.4) results.

4 Experiments and Discussion

4.1 Dataset

Following common benchmarks, we utilise the Large-scale CelebFaces Attributes (**CelebA**) [19] dataset, which contains more than 200k celebrity images with several annotations and is notably popular amongst face recognition (computer vision) algorithms[13][23]. This will serve us as the original distribution that G_0 will attempt to learn by employing the adversarial setup during training.

For the distillation process, we reproduce a dataset of similar size as extracted from a fully trained G_0, containing 200k examples of synthetic generations, which closely resemble those found in CelebA while not exactly being part of the original dataset itself. Note that this step in our methodology demonstrates the real-world utility of Generative Adversarial Networks in extending data distributions, given one is able to fully train a generator model that accurately mimics your existing (presumably scarce) dataset. Examples of successful data augmentation applications using GANs range from enhancing medical datasets for liver lesion classification [7] to generating believable synthetic biosignals [11] like those found in real electrocardiograms and electroencephalograms measured from real patients.

4.2 Implementation and Training

To visually depict the impact of compression in GANs, we make use of a Deep Convolutional Generative Adversarial Network (DCGAN) [20] model for the generator (implemented in PyTorch), which is initially (before compression) formed by five repeating layers composed of two-dimensional transposed convolutions, a group normalisation step and ReLU activation functions. The network's input is a one-dimensional latent vector (random noise) which gets upscaled until it reaches the 64×64 image resolution to then be received by a final $\tanh$ activation layer. After compression, a sequence of extracted significant channels forms a similar architecture with varying numbers of layers and input/output channels.

The discriminator model is composed of a series of two-dimensional convolutional, group normalisation and leaky ReLU layers that downscale the 64×64 input image until reaching a final Sigmoidal activation function, which outputs the binary classification action for any given input.

For GAN Slimming, a Very Deep Convolutional Network (VGGNet)[22] extracts high-level features to compute the perceptual loss between the student and teacher networks, acting as the distance measure as defined in Equation 2. The following figure explains the GS algorithm in further detail:

Algorithm 1 GAN Slimming (GS)[24]

```
1: Input: $\mathcal{X}, \mathcal{Y}, \beta, \rho, T, \{\alpha^{(t)}\}_{t=1}^{T}, \{\eta^{(t)}\}_{t=1}^{T}$
2: Output: $W, \gamma$
3: Random initialization: $W^{(1)}, \gamma^{(1)}, \theta^{(1)}$
4: for $t \leftarrow 1$ to $T$ do
5:     Get a batch of data from $\mathcal{X}$ and $\mathcal{Y}$;
6:     $W^{(t+1)} \leftarrow W^{(t)} - \alpha^{(t)} \nabla_W L_W$;
7:     $\gamma^{(t+1)} \leftarrow \text{prox}_{\rho^{(t)}}(\gamma^{(t)} - \eta^{(t)} \nabla_\gamma L_\gamma)$;
8:     $\theta^{(t+1)} \leftarrow \theta^{(t)} + \alpha^{(t)} \nabla_\theta L_\theta$;
9: end for
10: $W \leftarrow q_W(W^{T+1})$
11: $\gamma \leftarrow \gamma^{T+1}$
```

For channel pruning, we compute the proximal gradients on each training step by applying a soft threshold function to the weights of the group normalisation layers in the generator. This ensures that if the value of any gradient is below a given threshold (as determined by the ρ hyperparameter $\times$ the $\eta^{(t)}$ learning rate), we then consider it non-significant and zero the channel. Quantisation, on the other hand, is embedded right into the model architecture layers by using a two-dimensional transposed convolution that uses a custom linear quantisation estimator operator, which scales and clamps its inputs to the quantisation range to then return its outputs in the original scale but with a different number of bits as its representation.

After distillation, we copy the quantised weights of the most significant channels as determined by the channel pruning process into a new generator architecture that we programmatically infer to then produce a slimmed model. Then, as a final step, we fine-tune the compressed student using differential privacy by clamping the maximum gradient values, introducing random noise into our weights, and varying the batch size on each training step according to our privacy budget (accountant).

4.3 Quantitative Results

After training our initial GAN setup for 30 epochs ($\approx$ 42 hours in a MacBook Pro with the M1 chip and 1 GPU device), with a consistent learning rate of $2 \cdot 10^{-4}$ and a batch size of 128, we reached convergence with the final generator architecture G_0 consisting of 408.5M FLOPs and a memory size of 14.3 MB; and the trained discriminator D consisting of 99M FLOPs and a memory size of 11.1 MB. However, note that D does not require the application of any compression techniques as we discard it after training, never reaching the deployment stage.

While these numbers are not necessarily something that a resource-constrained device would not be capable of running, they still serve as a good proof of concept where future work could involve the use of larger models if wider computing resources are available.

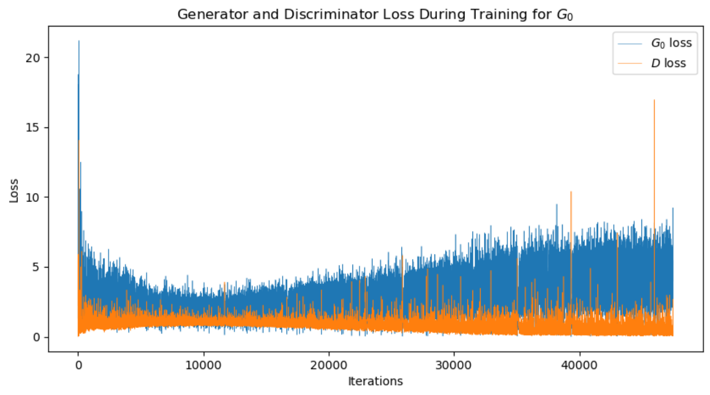

Figure 1: Evaluation loss of G_0 and D throughout training.

Figure 1 shows the progress of the evaluation loss throughout the training session, with G_0's and D's losses rapidly decreasing until the 10,000th iteration, with a slight increase for G_0 as D becomes more accurate. Note the instability of training as depicted by the variability of the loss function; works like [1] constitute an attempt at formalising similar issues of training GANs using rigorous proofs.

Upon creating a distillation dataset of 200k samples (similar in size to CelebA), we train three student models (32-bit, 16-bit, 18-bit) by employing channel pruning and weight quantisation techniques. Following [17], we incorporate a progressive cosine annealing learning rate of 10^{-5} with weight decay 10^{-3} for both Adam optimisers (G_s and D) and a third stochastic gradient descent optimiser for the γ parameters (channel pruning) with an initial learning rate of 10^{-1} and a momentum hyperparameter of 0.5. For channel pruning, we apply $\rho = 10^{-2}$ as the weight for the L1 loss and $\beta = 20$ for the weight of the GAN loss.

All three student models are trained for 20 epochs and compute their corresponding perceptual loss by employing a VGG model, which extracts high-level features to accurately compare the students' generations with the teacher's to incorporate their differences into the error function.

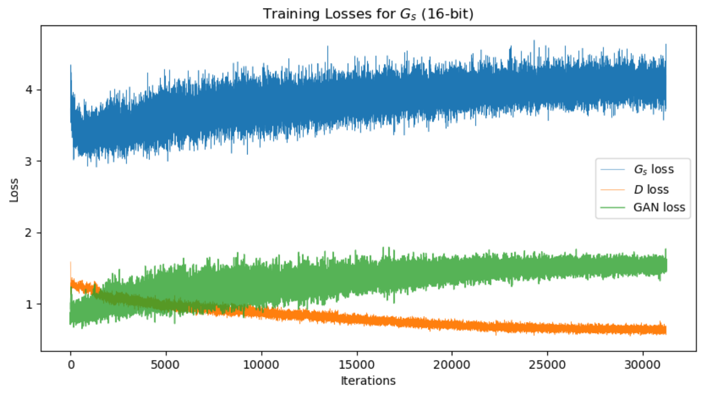

Figure 2: Evaluation loss of G_s (16-bit quantisation) and D throughout training.

Figure 2 illustrates the training progress for the GAN setup with a 16-bit generator model, where GAN loss is the combined loss of both networks as defined by the Binary Cross-Entropy criterion. Most notably, the training process for students appears to be much more stable than the teacher's; this is expected as we utilise the pre-trained generator (G_0) as the starting point for the student. The following figure depicts the decreasing number of non-zero channels as pruned throughout the training process:

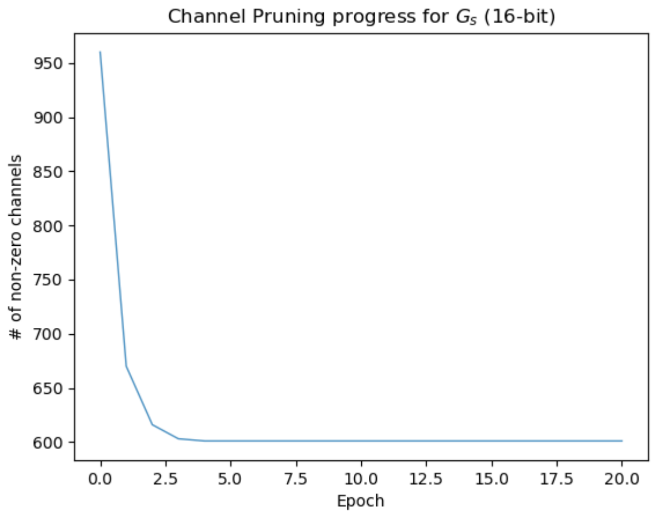

Figure 3: Number of non-zero channels of G_s (16-bit quantisation) throughout training.

Note that, as demonstrated empirically in our experiments, the number of significant channels always reaches the convergence to a minimum value dependent upon the magnitude of the ρ hyperparameter. Selecting a ρ that is too small will negate the effects of channel pruning and leave almost the same number of non-zero channels as initially specified, while a ρ that is too large will eliminate too much information passing across layers and lead to catastrophic imbalances in the adversarial setup. For a fully trained 16-bit student generator, we compute the L0 sum to be 601 channels and the expected L1 to be 0.25, with similar values for all other quantisation levels. In contrast, the non-quantised teacher model has an L0 of 960 (all channels) and an L1 of 0.89.

To measure the compression efficiency, we follow [21] and define the following two metrics:

$$r_s = \frac{FLOPs_{G_0}}{FLOPs_G}, \; r_{c=}\frac{ModelSize_{G_0}}{ModelSize_G} \tag{4}$$

where larger r_s and r_c indicates greater degrees of compression. Table 1 shows the resulting metrics for different quantisation levels (32-bit, 16-bit and 8-bit) after model distillation and channel pruning.

Table 1: Compression results after GS.

Metric	G_0	GS-32	GS-16	GS-8
FLOPS (M)	408.5	254.37	**205**	148
Memory (MB)	14.3	6.1	**5.2**	4.2
r_s	-	1.61	1.99	**2.76**
r_c	-	2.34	2.75	**3.4**

After applying GS, we find that the ratios of compression achieved are very promising, reaching a maximum memory reduction of 70.63% when using 8-bit quantisation and a similar reduction of 63.77% in the number of floating-point operations for the same compression level. More conservative approaches, such as using 16-bit representation, still achieve similarly good results with a memory reduction of 63.64% and 49.82% in the number of operations.

In order to remain faithful to the most popular techniques, which in this case means accepting the most commonly used tensor representations, we select the 16-bit slimmed generator for further fine-tuning in the privacy-preserving concern of this project. While the 8-bit model achieved better results, we acknowledge, as previously shown, that the 16-bit quantisation level is a good enough architecture for the purposes of this next step.

For differential privacy, we employ the open-source Opacus framework[25] to serve as our privacy engine. Previously selecting a δ of 10^{-5}, we choose to apply the DP principles to the discriminator model (since it is the one querying the real data) with a Privacy loss Random Variables (PRVs) accountant[10], which utilises the Fast Fourier transform to calculate tight bounds on the privacy expenditure efficiently. This accountant discretises the PRVs at each step of the computation and convolves the approximations, which can then be used to recover the ϵ value.

After 20 epochs with the 16-bit slimmed generator and a discriminator trained from scratch under similar conditions to the training process for the students, we achieve a final ϵ of 1.76, which indicates that the ratio of the probabilities of a given output occurring with and without a particular individual observation is bounded by $e^{1.76}$. This means that the presence or absence of an individual's data (face picture) can, at most, increase the likelihood of any specific output by a factor of 5.81. This is a value that we deem acceptable for our project, but tighter use cases might require the training of fewer epochs or the tweaking of various hyperparameters such as the accountant, the δ value or the number of examples in the distilled dataset.

4.4 Qualitative Results

Going beyond quantitative analysis, we include this brief section to showcase the visual degradation of synthetic samples as the levels of quantisation increase and, moreover, as we deviate from the original data distribution. The following are some produced samples from the fully trained G_0 model (30 epochs, no GS, no DP):

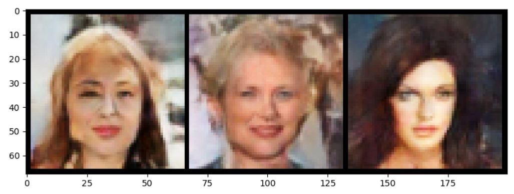

Figure 4: Synthetic samples generated by the teacher model G_0.

As one can appreciate, the 64×64 images are of believable strength and, hence, the result of a successful adversarial run where the generator G_0 was indeed capable of fooling a well-trained discriminator D. These are the kinds of generations that were used to create the distillation dataset S employed to train all students.

A set of examples created by the distilled, channel pruned, 16-bit student generator (20 epochs, GS-16, no DP) are as follows:

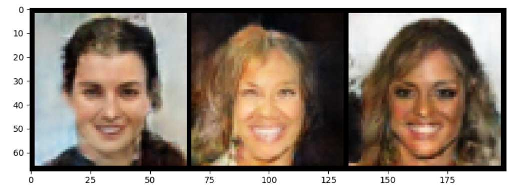

Figure 5: Synthetic samples generated by the student model G_s (16-bit quantisation).

And to showcase the extreme case of 8-bit quantisation (20 epochs, GS-8, no DP):

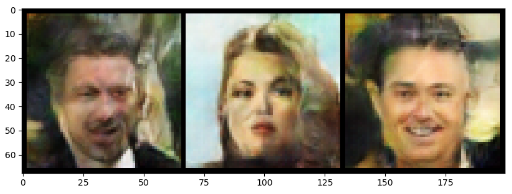

Figure 6: Synthetic samples generated by the student model G_s (8-bit quantisation).

Finally, after applying differential privacy to our discriminator when trained against the 16-bit slimmed generator G (20 epochs, GS-16, DP), we reach the following generation quality, which should serve as the final showcase of our device-friendly, privacy-preserving GAN architecture:

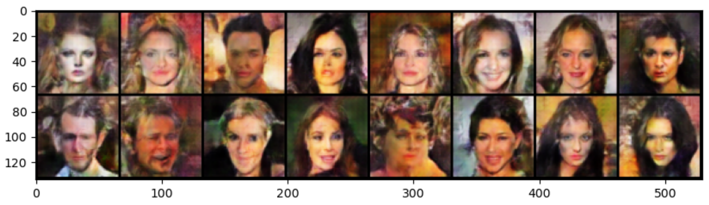

Figure 7: Synthetic samples generated by the final model.

4.5 Discussion of Limitations

As a final note, we feel important to mention that our project still faces notable limitations. Primarily, the integration of GAN Slimming (GS) and Differential Privacy (DP) significantly increases computational demands. GS's model compression techniques, despite lowering model size, drastically slow down the training process. Additionally, DP's privacy mechanisms (as introduced in Section 2.3) further expand the average duration of a single step time, as well as reduce the final evaluation metrics (primarily due to noisy gradients). Furthermore, typical GAN issues like vanishing gradients still prevail, potentially destabilising training and affecting the quality of the generations. We believe that these limitations emphasise areas for future optimisation in developing compressed, privacy-preserving GANs.

5 Conclusion

In this report, we successfully demonstrate the feasibility of creating a device-friendly, privacy-preserving Generative Adversarial Network. Through the application of GAN Slimming (GS), we significantly reduce the model's computational and memory demands, making it suitable for deployment on resource-constrained devices. The integration of Differential Privacy (DP) into our training ensures that the synthetic data produced by our model maintains individual privacy.

While the experiments conducted using the CelebA dataset and the implementation of a Deep Convolutional Generative Adversarial Network (DCGAN) model have shown promising results, future work could involve the application of this methodology to larger, more complex models and different datasets, potentially broadening the scope of GAN applications while sticking to the constraints of device capabilities and privacy concerns.

References

[1] Martin Arjovsky and Léon Bottou. “Towards Principled Methods for Training Generative Adversarial Networks”. Version 1. In: (2017). DOI: 10.48550/ARXIV.1701.04862. URL: https://arxiv.org/abs/1701.04862 (visited on 12/23/2023).

[2] Hanting Chen et al. “Distilling Portable Generative Adversarial Networks for Image Translation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 3, 2020), pp. 3585–3592. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v34i04.5765. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5765 (visited on 01/02/2024).

[3] Edward Choi et al. “Generating Multi-label Discrete Patient Records Using Generative Adversarial Networks”. Version 3. In: (2017). DOI: 10.48550/ARXIV.1703.06490. URL: https://arxiv.org/abs/1703.06490 (visited on 01/02/2024).

[4] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. “Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis”. Version 2. In: (2017). DOI: 10.48550/ARXIV.1701.05927. URL: https://arxiv.org/abs/1701.05927 (visited on 01/02/2024).

[5] Cynthia Dwork and Aaron Roth. “The Algorithmic Foundations of Differential Privacy”. In: *Foundations and Trends® in Theoretical Computer Science* 9.3-4 (2013), pp. 211–407. ISSN: 1551-305X, 1551-3068. DOI: 10.1561/0400000042. URL: http://www.nowpublishers.com/articles/foundations-and-trends-in-theoretical-computer-science/TCS-042 (visited on 12/26/2023).

[6] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. “Real-Valued (Medical) Time Series Generation with Recurrent Conditional GANs”. Version 2. In: (2017). DOI: 10.48550/ARXIV.1706.02633. URL: https://arxiv.org/abs/1706.02633 (visited on 01/02/2024).

[7] Maayan Frid-Adar et al. “Synthetic Data Augmentation Using GAN for Improved Liver Lesion Classification”. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). Washington, DC: IEEE, Apr. 2018, pp. 289–293. ISBN: 978-1-5386-3636-7. DOI: 10.1109/ISBI.2018.8363576. URL: https://ieeexplore.ieee.org/document/8363576/ (visited on 01/12/2024).

[8] Ian Goodfellow et al. “Generative Adversarial Networks (2018)”. In: *Communications of the ACM* 63.11 (Oct. 22, 2020), pp. 139–144. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3422622. URL: https://dl.acm.org/doi/10.1145/3422622 (visited on 12/23/2023).

[9] Ian J. Goodfellow et al. “Generative Adversarial Networks (2014)”. Version 1. In: (2014). DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661 (visited on 12/23/2023).

[10] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. “Numerical Composition of Differential Privacy”. Version 3. In: (2021). DOI: 10.48550/ARXIV.2106.02848. URL: https://arxiv.org/abs/2106.02848 (visited on 01/12/2024).

[11] Shota Haradal, Hideaki Hayashi, and Seiichi Uchida. “Biosignal Data Augmentation Based on Generative Adversarial Networks”. In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). Honolulu, HI: IEEE, July 2018, pp. 368–371. ISBN: 978-1-5386-3646-6. DOI: 10.1109/EMBC.2018.8512396. URL: https://ieeexplore.ieee.org/document/8512396/ (visited on 01/12/2024).

[12] Yihui He et al. “AMC: AutoML for Model Compression and Acceleration on Mobile Devices”. Version 4. In: (2018). DOI: 10.48550/ARXIV.1802.03494. URL: https://arxiv.org/abs/1802.03494 (visited on 01/02/2024).

[13] Md Imran Hosen and Md Baharul Islam. “HiMFR: A Hybrid Masked Face Recognition Through Face Inpainting”. Version 1. In: (2022). DOI: 10.48550/ARXIV.2209.08930. URL: https://arxiv.org/abs/2209.08930 (visited on 01/12/2024).

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. Version 1. In: (2016). DOI: 10.48550/ARXIV.1603.08155. URL: https://arxiv.org/abs/1603.08155 (visited on 01/12/2024).

[15] Tero Karras, Samuli Laine, and Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, CA, USA: IEEE, June 2019, pp. 4396–4405. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00453. URL: https://ieeexplore.ieee.org/document/8953766/ (visited on 01/02/2024).

[16] Christian Ledig et al. “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”. Version 5. In: (2016). DOI: 10.48550/ARXIV.1609.04802. URL: https://arxiv.org/abs/1609.04802 (visited on 01/02/2024).

[17] Sicong Liu et al. “AdaDeep: A Usage-Driven, Automated Deep Model Compression Framework for Enabling Ubiquitous Intelligent Mobiles”. Version 1. In: (2020). DOI: 10.48550/ARXIV.2006.04432. URL: https://arxiv.org/abs/2006.04432 (visited on 01/02/2024).

[18] Zhuang Liu et al. “Learning Efficient Convolutional Networks through Network Slimming”. Version 1. In: (2017). DOI: 10.48550/ARXIV.1708.06519. URL: https://arxiv.org/abs/1708.06519 (visited on 01/02/2024).

[19] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. Version 3. In: (2014). DOI: 10.48550/ARXIV.1411.7766. URL: https://arxiv.org/abs/1411.7766 (visited on 01/12/2024).

[20] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. Version 2. In: (2015). DOI: 10.48550/ARXIV.1511.06434. URL: https://arxiv.org/abs/1511.06434 (visited on 01/12/2024).

[21] Han Shu et al. “Co-Evolutionary Compression for Unpaired Image Translation”. Version 1. In: (2019). DOI: 10.48550/ARXIV.1907.10804. URL: https://arxiv.org/abs/1907.10804 (visited on 01/09/2024).

[22] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. Version 6. In: (2014). DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556 (visited on 01/12/2024).

[23] Viktor Varkarakis and Peter Corcoran. “Dataset Cleaning — A Cross Validation Methodology for Large Facial Datasets Using Face Recognition”. In: *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. 2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX). Athlone, Ireland: IEEE, May 2020, pp. 1–6. ISBN: 978-1-72815-965-2. DOI: 10.1109/QoMEX48832.2020.9123123. URL: https://ieeexplore.ieee.org/document/9123123/ (visited on 01/12/2024).

[24] Haotao Wang et al. “GAN Slimming: All-in-One GAN Compression by a Unified Optimization Framework”. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi et al. Vol. 12349. Cham: Springer International Publishing, 2020, pp. 54–73. ISBN: 978-3-030-58547-1 978-3-030-58548-8. DOI: 10.1007/978-3-030-58548-8_4. URL: https://link.springer.com/10.1007/978-3-030-58548-8_4 (visited on 12/23/2023).

[25] Ashkan Yousefpour et al. “Opacus: User-Friendly Differential Privacy Library in PyTorch”. Version 4. In: (2021). DOI: 10.48550/ARXIV.2109.12298. URL: https://arxiv.org/abs/2109.12298 (visited on 01/12/2024).